Moving Lights and Cameras for Better 3D Perception of Indoor Scenes

Arkadeep Narayan Chaudhury

October 2024

CMU-RI-TR-24-71

The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, Pennsylvania
United States of America

Thesis Committee:

Prof. Christopher G. Atkeson (Chair)
Prof. Srinivasa Narasimhan
Prof. Oliver Kroemer
Dr. Christoph Lassner (World Labs)
Dr. Andy Zeng (Generalist AI)

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Robotics.

Abstract

Decades of research on computer vision have highlighted the importance of active sensing – where an agent controls the parameters of the sensors to improve perception. Research on active perception in the context of robotic manipulation has demonstrated many novel and robust sensing strategies involving a multitude of sensors like RGB and RGBD cameras and a variety of tactile, proximity, and spectroscopic sensors resulting in ever-improving representations and understanding of the world around the agent. In this work we explore sensor configurations, sensor positioning, and robot workspace illumination to improve 3D perception of objects in table-top scaled scenes. We divide the problem into three parts to explore the effects of moving camera-based sensors in a robot's workspace, moving illumination sources around a robotic workspace and controlling both illumination and camera movement.

We show that a robot mounted ensemble of camera-based sensors (namely RGB, RGBD and tactile) can help visually servo a manipulator and accurately localize contacts on objects using vision and touch.

We show that known directional illumination can be very effective for measuring objects in the workspace accurately. We demonstrate this with a robot workspace scaled highly accurate photometric stereo stage.

Finally, we show that multi-view, multi-illumination images captured using a custom multi-flash camera system can be effective in reconstructing, synthesizing novel views of, and relighting table-top scaled scenes. In addition to the capture system, we also demonstrate efficient algorithms and representations for 3D perception of small scenes.

– *In Grateful Remembrance of* –

Dr. Santanu Pramanik

7th November 1988 – 23rd October 2021

Acknowledgments

If you feel like the dumbest person in the room then you're in the correct room.[1]

I'd like to start by acknowledging the unwavering support and freedom I received from my advisor Chris Atkeson in all aspects related to my PhD studies. He never forced me to submit a deadline or work on any particular project. Although my thesis took a diversion from Chris' main research focus (robot learning), my work has been thoroughly influenced by his simple yet profound "I don't get it" type questions, which prompted me to think deeply about technicalities of the projects. Apart from research mentorship, thank you for teaching me how to think, how to discard apparent fluff around a topic in vogue, instilling in me the value of real robot demonstrations of the algorithms we worked on, and teaching me how to think about individual academic contribution as art and find validation for my work from within, rather than looking outward. Although, I am not sure I have mastered the final aspect, this is definitely a valuable tool I now have in my toolbox.

Thank you Srinivas for your advice on my research – especially the final project. Your support was invaluable in framing a work at the intersection of robotics and computer vision for a primarily computer vision audience. Thank you Oliver and Andy, the two other robotics experts on my committee, for your advice on my work. I really appreciated the hour long check-in meetings with Oliver every semester that helped me align my research in a direction that was meaningful to both robotics and computer vision researchers. Thank you Christoph for being so generous with your time – I really appreciated your advice on my research and your strong support in helping me make professional connections.

Thank you Santanu, the dedicatee of this thesis, for suggesting me to pursue a PhD in the first place. I met Santanu in 2018, during my previous graduate studies at the Indian Institute of Science and throughout the years[2], he was a constant source of friendship, advice, and support. Looking back, it is very unlikely that I would have even considered the idea of earning a PhD, and I can almost solely credit him for setting me up for this journey. It is truly unfortunate that we cannot share this moment.

Thank you to the members of Atkeson Lab – Alisa, Leo, and Mrinal for being incredibly valuable sounding boards for my research ideas. Thank you Leo for being an incredible source of literature and technical advice – both recent and classical. Our discussions have profoundly shaped the way I tackled the problems in this thesis. Mrinal suggested the addition of the overhead light source for the photometric stereo rig (Part III), which made the estimation of the shadows much better leading to a significant improvement in our system's performance.

A special thanks goes out to Prof. Wenzhen Yuan and the members of the erstwhile RoboTouch Lab at CMU[3]. Thanks for helping me test the very first versions of my talks. I also gratefully recognize Wenzhen's support in getting me clearance to use her lab equipment at the peak of

[1] Adage. Quoted here from Destin Sandlin of Smarter Every Day (YouTube/X). Tweet from 09/18/2018
[2] Santanu passed away in October of 2021, from complications following a major surgery.
[3] The RoboTouch Lab moved from the RI to UIUC in August 2023

COVID-19 lockdowns (Fall 2020), which led to the timely completion of the first project (Part II).

Thank you Igor Vasiljevic, Sergey Zakharov, Vitor Guizilini, Rares Ambrus, and other members of the Machine Learning Team at the Toyota Research Institute for being fantastic internship mentors and hosting me for the Summer of 2023. I truly appreciate your support that enabled me pursue a project at the intersection of computer vision, machine learning, and robotics (part IV).

Outside my advisors, collaborators and mentors at CMU, I owe debts of gratitude to my friends[4] Ananya, Animesh, Anurag, Arpit, Ashwin, Akhil, Ben E., Ben F., Chase, Dorian, Kate, Md. Suhail, Rishi, Samantha, Sarvesh, Sarah, Simin, Tejus, Tejas, Uksang and Vineet. Thank you for your feedback on almost all of my research output (talks and manuscripts) and the frequent engaging and refreshing conversations about research and life.

Apart from faculty and friends, I've had the opportunity to work with the kindest program manager – Suzanne. Thank you for reminding me of several critically important deadlines, advice on general logistics of the PhD program and answering my questions on the program guidelines with incredible simplicity and detail. Thank you Jean for your help with the logistics RoboOrg, and facilitating lab access during the COVID-19 lockdowns. Thank you Alan[5] for helping with the packages and for explaining why Newell Simon Hall doesn't have a few more floors, how to take amazing pictures of minerals, and the reason for the now decommissioned rail line on the B level of NSH.

This thesis also would not have existed without the support of my parents and my younger brother, who helped make the person I am today. Finally, a sincere Thank You to the Robotics Institute and Carnegie Mellon University for making the search for the "*correct room*" incredibly simple.

[4]alphabetically ordered. Apologies for the unintended omissions.
[5]Alan D. Guisewite passed away in March 2023 after dedicating 40 years of service to the Robotics Institute.

Contents

Contents **vii**

List of Tables **x**

List of Figures **xi**

I Introduction **1**

1 Introduction **2**

1.1 Motivation 2

1.1.1 Thesis statement and research contributions 4

II Moving Cameras **6**

2 Using Collocated Vision and Tactile Sensors for Visual Servoing and Localization **7**

2.1 Introduction 7

2.2 Related work 10

2.3 Methods 12

2.3.1 Sensor platform 12

2.3.2 Visual servoing with hand mounted cameras 12

2.3.3 Pose estimation through vision 17

2.3.4 Precise contact pose estimation through touch 19

2.4 Results 20

2.5 Discussion and future work 25

2.6 Conclusion 26

2.7 Retrospectives 26

III Moving Lights **27**

3 Photometric stereo for manipulation of Lambertian objects **28**

3.1 Introduction 28

3.2 Related Work . . . 30
3.3 Methods . . . 31
3.3.1 Details of the hardware platform . . . 31
3.3.2 Modelling the illumination of the workspace . . . 32
3.3.3 Recovering surface normals from images . . . 33
3.3.4 Recovering object edges and depth . . . 34
3.3.5 Details of our steps for calculating occlusion edges . . . 35
3.4 Experiments and Results . . . 35
3.4.1 Performance of our sensing approach . . . 36
3.4.2 Controlled illumination for pickup tasks . . . 38
3.4.3 Measuring deformation . . . 40
3.4.4 Controlled illumination for localization . . . 41
3.4.5 Application to generic objects . . . 45
3.5 Discussion and Future Work . . . 45
3.6 Conclusions . . . 46
3.7 Retrospectives . . . 46
3.8 Supplementary Materials . . . 46
3.8.1 Description of our algorithm for vacuum grasping . . . 46
3.8.2 Description of our algorithm for measuring deformation . . . 47
3.8.3 Description of our pose estimation pipelines . . . 48
3.8.4 Implementation details and hyperparameters . . . 50
3.8.5 Practical considerations in design . . . 50
3.8.6 Hyperparameters and their effects . . . 51

IV Moving Lights and Cameras **52**

4 Multi-view, multi-illumination capture for learning 3D object representations **53**
4.1 Introduction . . . 54
4.2 Related Work . . . 54
4.3 Methods . . . 55
4.3.1 Primer on neural 3D fields . . . 56
4.3.2 Incorporating dense metric depth . . . 56
4.3.3 Incorporating depth edges in joint optimization of appearance and geometry 57
4.3.4 AdaShell^{++}: Accelerating training with dense depth . . . 59
4.3.5 Baselines augmented with depth . . . 60
4.4 Setup and Dataset . . . 61
4.4.1 A multi-flash stereo camera . . . 61
4.4.2 Captured dataset . . . 61
4.5 Experiments and results . . . 62
4.5.1 Accuracy of incorporating metric depth . . . 62
4.5.2 The effect of depth edges in training . . . 63

4.5.3 View synthesis with dense depth . . . 63
4.5.4 Using noisy depth . . . 65
4.5.5 Relighting . . . 66
4.6 Limitations . . . 66
4.7 Conclusions . . . 68
4.8 Retrospectives . . . 68
4.9 Supplementary Materials . . . 69
4.9.1 Representations and implementation details . . . 69
4.9.2 A multi-flash stereo camera . . . 74

V Conclusion **77**

5 Conclusion **78**
5.1 Improving capture systems . . . 79
5.1.1 Upgrading monocular depth with metric hints . . . 79
5.1.2 Example 1: Super-resolving depth from stereo . . . 80
5.1.3 Example 2: Improving stereo to capture highly non-Lambertian objects . . . 81
5.1.4 Example 3: Convenient camera pose estimation for face capture rigs . . . 81
5.2 Conclusions . . . 82
5.2.1 Relevance of our work in robotics research . . . 83
5.2.2 Frequently asked questions . . . 83

Bibliography **85**

List of Tables

2.1 Relative uncertainties in localization using only vision and tactile sensing augmented by vision. 22
2.2 Best and worst case localization errors when the contacts are selected at random. . . 24

3.1 Qualitative comparison between different normal integration methods 35
3.2 Comparison of our approach with commercial depth sensors 36
3.3 Quantitative measurements of our approach in estimating object shape 37
3.4 Comparison of our method of measuring deformation versus [116, 117] 41
3.5 Performance of pose estimation for known objects 43
3.6 Performance of our tracking pipeline for unknown objects. 44
3.7 Time breakdown of various steps in our pipeline 50

4.1 Some differences between our data and a few established datasets 62
4.2 Accuracy of reconstruction from un-posed RGBD images. 62
4.3 Accuracy of reconstruction from posed RGBD images. 62
4.4 Depth edges help prioritize learning of texture discontinuities over geometric ones . . 63
4.5 Training performance for view synthesis. 63
4.6 Effect of noisy depth and depth edges. 65
4.7 Our performance in relighting of scenes 66

List of Figures

1.1 Accessible systems to collect high-quality 3D data from small scenes 2
1.2 Thesis Statement: Moving lights and cameras can help 4

2.1 Collocating vision and tactile sensors. 8
2.2 Details of our tactile sensor. 11
2.3 Using optic flow for visual servoing 13
2.4 Empirical evidence that averaging across steps refines prediction. 15
2.5 Statistics of POE tracking errors 16
2.6 Our pipeline to estimate pose through vision and touch. 17
2.7 Pose estimation with RGB image and tactile sensor 20
2.8 Steps in localizing small objects 21
2.9 Collocated vision can help us disambiguate between repeated tactile features. In this experiment we touch the middle of the set of teeth and could localize the contact using our pipeline. 21
2.10 Additional demonstrations of our algorithm presented in sections 2.3.3 and 2.3.4 for localizing contact from high-resolution tactile imprints. 23

3.1 We demonstrate the value of controlled illumination approaches for robot manipulation workspaces 29
3.2 Our pipeline for performing photometric stereo at the scale of a robot's workspace . 31
3.3 Our measurements compared to commercial depth sensors 36
3.4 Our pipeline for grasping objects in a robot's workspace. 39
3.5 Our results of measuring the deformation of known objects. 40
3.6 Estimating object pose in the robot workspace. 42
3.7 Our photometric stereo on colored, non-Lambertian objects 44

4.1 An approach for the photo-realistic capture of small scenes 53
4.2 A snapshot of the important supervision signals. 55
4.3 Overview of our sampling process during training 58
4.4 We demonstrate a corner case of jointly refining appearance and geometry. 58
4.5 AdaShell^{++}recovers sampling volumes similar to [186] 59
4.6 Reconstructions with AdaShell^{++} 59

4.7 Schematic of our multi-flash stereo camera 61
4.8 Relative performance of all the methods 64
4.9 AdaShell^{++}can work with noisy depth 65
4.10 Relighting scenes from a few captured views 67
4.11 Hardware limitations: bokeh and decoupled flashes 68
4.12 Our approach to recovering 3D assets from captured data 69
4.13 All the pipelines can be used to extract the "base-color" of the scene 72
4.14 Optimizing for the full Disney BRDF is difficult 73

5.1 Looking forward: Improving 3D capture 78
5.2 Super-resolving stereo with monocular cues 80
5.3 Handling reflective and transparent objects 81
5.4 Estimating camera poses for face capture 82

I

Introduction

1 Introduction

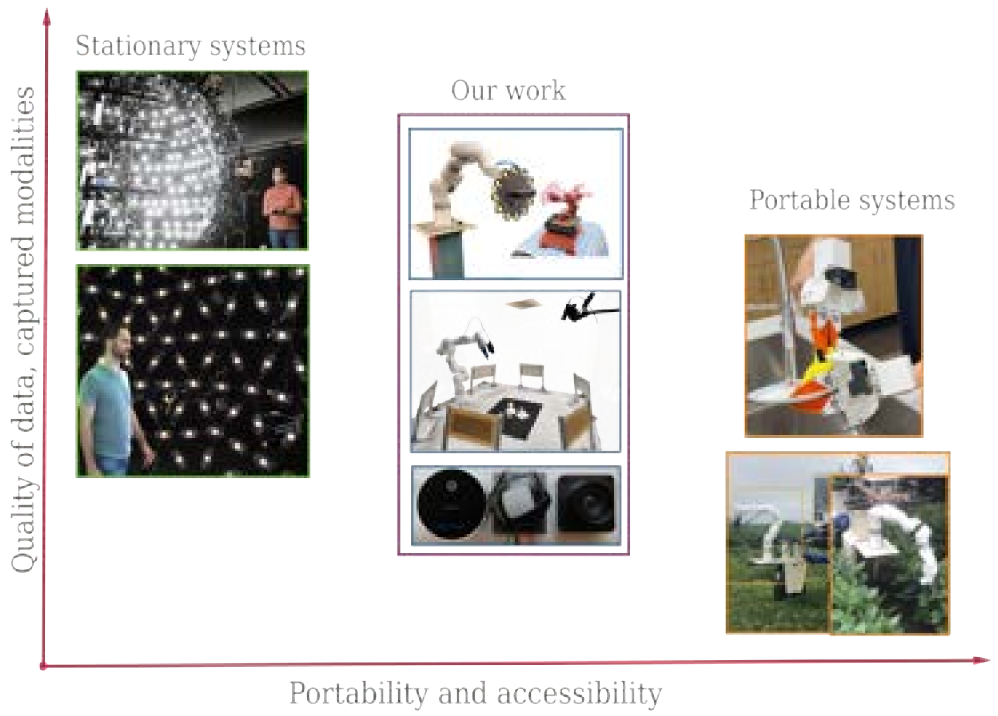

Stationary systems
Our work
Portable systems
Quality of data, captured modalities
Portability and accessibility

Figure 1.1: Capturing high quality data is crucial for understanding small scenes. Systems used to capture data can range from highly complex and specialized capture rigs to simple hand-held systems based around high quality cameras. While the massively multi-view systems (left, [1, 2]) capture very high quality of data at the cost of portability, the more accessible systems (right, [3, 4]) often sacrifice the quality of data and modalities captured in favor of accessibility. In this work (center, [5, 6, 7]), we show that moving cameras, camera based sensors and lights around a robot's workspace can be useful techniques to collect high-quality data from small scenes – effectively identifying an accessible middle ground.

1.1 Motivation

This work was motivated by the comparatively smaller but exciting body of literature on active sensing systems for perception. Active sensing, most notably defined by Bajcsy ([8, 9]) and Aloimonos ([10, 11]) encompasses a variety of problems and techniques that involve actively controlling the sensor for improved perception. Examples include actively modifying camera parameters, moving a camera to look around occluding objects, and obtaining the next-best-view. In the context of robot manipulation, active sensing can additionally encompass placing proximity sensors (e.g. ultra-sonic sensors, spectrometers) near objects or actively touching objects with tactile sensors to estimate their material and surface properties. Research on active sensing for manipulation has resulted in many robust pipelines and sensor configurations which leverage the flexibility in moving the sensors and selecting appropriate sensors for the manipulation task at hand.

Our work is also motivated by the surprising effectiveness of active sensing for small scenes. In particular, we looked at the advantageous aspects of moving sensors, selecting sensor configurations and controlling the illumination of a robot's workspace to improve 3D perception.

As a concrete example, we address some of the challenges in capturing high-quality data from tabletop-scale scenes to enhance their 3D understanding (refer to fig. 1.1). The existing literature in this field can broadly be categorized into two main approaches. The more accessible data capture systems on the left of fig. 1.1 featuring [3, 4] among others, focus on robust and deployable sensors that capture a *reasonable* quality of data. While the quality of the data collected is satisfactory for the downstream tasks (imaging agricultural crops, collecting data for robotic manipulation etc.), these systems often sacrifice the possibility of collecting several modalities of data in favor of portability and accessibility of their platforms.

Conversely, high-resolution capture systems, such as those described in [1, 2] and shown in fig. 1.1, amongst others ([12, 13, 14]), employ stationary, large-scale multi-view, multi-modal capture setups to digitize humans. The data gathered by these systems have been used as benchmarks by the research community, with the CMU Panoptic Studio datasets from [13, 14] remaining the gold standard among open-source datasets for digital human capture almost 10 years after they were published. More recent improved datasets for human performance capture ([1]) and animating human faces ([15, 16]) remain outside the public domain. The systems ([1, 2]) used to capture such data are also out of the reach for the academic researcher.

The role of high quality datasets containing multi-modal information is undeniable in the synthesis of high quality 3D foundation models (e.g. [17]), setting performance benchmarks across several perception tasks. Data from more accessible sensing platforms have also been used to synthesize implicit, task specific world models ([18]) which, cannot be trivially repurposed. Is there a middle ground where an academic researcher can use a sensing paradigm to collect high quality data for the task at hand? This is the third and final motivating question for our research.

1.1.1 Thesis statement and research contributions

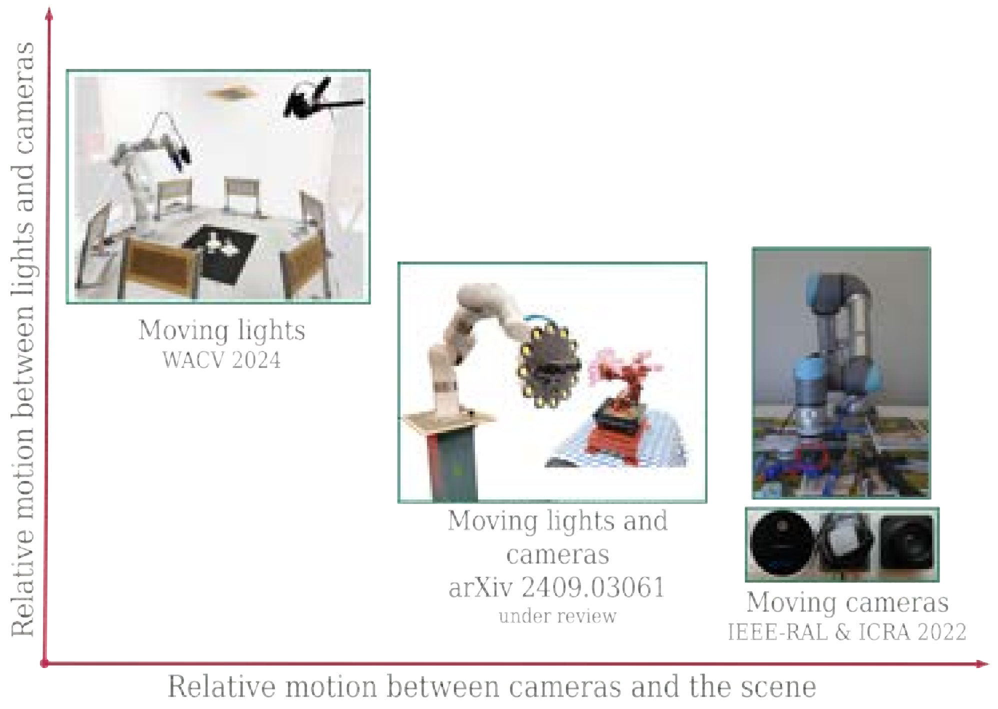

Relative motion between lights and cameras
Moving lights
WACV 2024
Moving lights and cameras
arXiv 2409.03061
under review
Moving cameras
IEEE-RAL & ICRA 2022
Relative motion between cameras and the scene

Figure 1.2: Thesis Statement: Moving lights and cameras can provide an effective middle ground. This thesis argues that, in addition to moving cameras and camera based sensors, switching between sensors, controlling the illumination of the robot's workspace and coordinating robot mounted cameras with lights can capture important information for manipulation of the objects in the workspace. To demonstrate this, we present three independent pieces of research ([5, 6, 7]).

In the search for a middle ground, this thesis argues that, in addition to moving cameras and camera based sensors, switching between sensors, controlling the illumination of the robot's workspace, and coordinating robot mounted cameras with lights can capture important information for manipulation of objects. To demonstrate this, we present three independent pieces of research.

Our first work (part II, [5]) looks at how moving cameras and camera based sensors can help robot manipulation. We show that :

- Robot manipulator mounted cameras can be effectively used for visual servoing.
- Using multiple moving vision based sensors collaboratively yields better performance than using them individually for estimating object pose at contact.

Our second work (part III, [6]) shows that with strong priors on object reflectance:

- Controlling workspace illumination can very effectively capture information of the object's surfaces.
- Obtaining surface normals by reasoning about the observed shading yields better results than obtaining normals as spatial gradients of surface depths.

- Common manipulation tasks of estimating deformation, executing immobilizing grasps, and estimating object poses can benefit from high quality information about surface normals and occlusion edges.

Finally, our third work (part IV, [7]) shows the benefits of moving lights and cameras together by demonstrating a robot-mountable multi-flash stereo camera device. Additionally,

- we present a method to incorporate dense metric depth into the training of neural 3D fields, enabling state-of-the-art methods to use dense metric depth with minor changes.

- We investigate an artifact (fig. 4.4) commonly observed while jointly refining shape and appearance. We identify its cause as existing methods' inability to differentiate between depth and texture discontinuities. We address it by using depth edges as an additional supervision signal.

- We present a new approach to accelerate training of neural fields for photo-realistic scene capture and relighting from multi-view and multi-illumination images by incorporating metric depth.

II

Moving Cameras

2 Using Collocated Vision and Tactile Sensors for Visual Servoing and Localization

Coordinating proximity and tactile imaging by collocating cameras with tactile sensors can 1) provide useful information before contact such as object pose estimates and visually servo a robot to a target with reduced occlusion and higher resolution compared to head-mounted or external depth cameras, 2) simplify the contact point and pose estimation problems and help tactile sensing avoid erroneous matches when a surface does not have significant texture or has repetitive texture with many possible matches, and 3) use tactile imaging to further refine contact point and object pose estimation. We demonstrate our results with objects that have more surface texture than most objects in standard manipulation datasets. We learn that optic flow needs to be integrated over a substantial amount of camera travel to be useful in predicting movement direction. Most importantly, we also learn that state of the art vision algorithms do not do a good job localizing tactile images on object models, unless a reasonable prior can be provided from collocated cameras.

Additional results can be viewed on the paper website.

2.1 Introduction

This work is motivated by FingerVision [19] where the same camera was used for tactile sensing and to view nearby objects through a transparent elastomer. Although FingerVision convinced us of the importance of proximity imaging, it did not provide high resolution images of contact surface texture, and the transparent elastomer blurred proximity imaging, attracted dust, and got scratched and worn so the view of nearby objects was often not as good as we would like. Separating tactile and proximity imaging enables us to get the high resolution of GelSight [20, 21] tactile sensors that produce images of the surface texture (tactile imaging), and better proximity imaging with rigid lenses that don't attract dust as much, are easier to clean and don't scratch or get worn as easily. This paper explores an alternative to using the same camera for tactile and proximity imaging, where a tactile sensor is collocated with a camera for proximity sensing (fig. 2.1). Since the tactile sensor we use, a GelSight variant, is also based on a camera, we are actually collocating multiple cameras to provide tactile and proximity imaging.

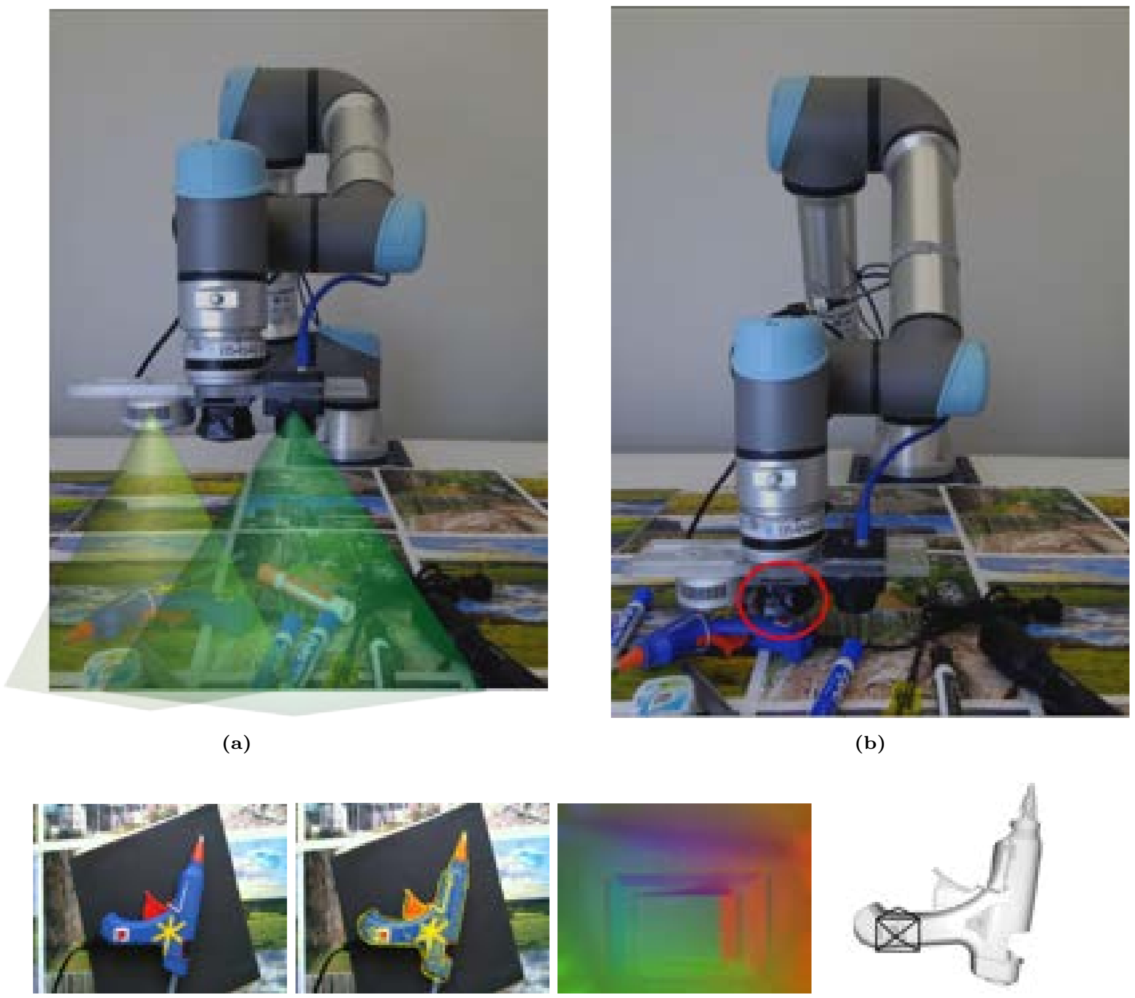


Figure 2.1: We demonstrate an approach to integrate sensors with different fields of view to visually servo the robot arm to a predetermined contact point and estimate the pose of a fixed object relative to the sensors at contact. Figures 2.1a and 2.1b show our sensor platform. We use 2 cameras with 70° and 100° fields of view, which we collocate with a modified GelSight sensor in the middle (fig. 2.1b). The cameras are used to visually servo the robot (fig. 2.1c) and generate a preliminary pose estimate (fig. 2.1d) while the robot is moving towards the target. At contact, the GelSight data is observed (fig. 2.1e) and the preliminary pose is then refined to generate the object pose at contact. Figure 2.1f shows the camera pose superimposed on the mesh model of the object.

Cameras that move with a robot hand can have less occlusion and more resolution since they are closer to manipulated objects. Direct measurement of the direction or bearing to an object and its pose relative to the hand can be used to guide the hand to a particular contact location and center the hand with respect to an object. External and head-mounted cameras are often occluded by the robot itself as well as manipulated objects, and need to use stereo, multi-view, or other forms of depth measurement to locate the hand relative to the approach axis and object, which involves subtracting two noisy estimates (typically large numbers) to estimate a smaller quantity, which is usually less accurate than directly measuring the smaller quantity. We have found depth measurements from stereo or time of flight (TOF) cameras usually have low spatial resolution, so getting the camera close to the hand is useful to improve depth resolution.

We divide the problem of contact pose estimation into two parts – The initial phase before contact, when cameras can be used for vision-based servoing to a contact point target as well as estimating a prior for the tactile sensor contact point and object pose estimation, and the contact phase which refines the prior pose estimates. In this paper we assume that 1) the object is fixed, even during contact, 2) the object is a single rigid body with no articulations, and 3) we have a prior 3D model of the object (potentially provided by our vision of the current object) so we can express the pose of the object with respect to this model. For this paper we put aside the gross object localization and recognition problems in order to focus on fine localization, so we assume a vision system has already located the object, created a bounding box, and recognized the object by creating or selecting an appropriate 3D model that we want to register the actual object to (e.g. [22]). Our experimental pipeline involves selecting a goal and then visually servoing to that goal, recording color and depth data from the vision sensors, generating and maintaining pose estimates of the object, and using the estimates along with tactile information received at contact to localize the contact point on the object. Through this work we show that:

- The optic flow, as observed by the hand mounted cameras, can be used to predict the heading direction of the hand using image-based techniques rather than 3D geometry. In our setup frame-to-frame optic flow was dominated by small changes in orientation of the hand and thus the cameras. Optic flow had to be integrated across about 10cm of hand travel to be useful.
- A few “mid-course” corrections can correct almost all the error in trajectories.
- Pose errors, when measured only with the cameras are about ± 1.5 cm and $\pm 2°$ in translation and rotation respectively about a vertical axis.
- Given these priors, tactile estimation based on a GelSight sensor further improved the pose estimates to an uncertainty of ± 1.5mm and $\pm 0.5°$ in translation and rotation respectively in cases distinct tactile signals were available.
- Collocated vision is particularly useful when an object does not have distinctive tactile surface texture, or has repetitive surface texture. We show that using tactile sensing collocated with vision can help disambiguate tactile signals when used for localization.

We provide additional details of our work, tables of results and reference implementations of our algorithms described in the paper here: https://arkadeepnc.github.io/projects/collocated_vision_touch/index.html

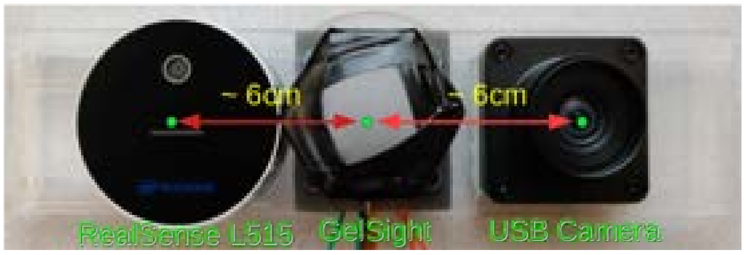
2.2 Related work

In this section we provide a survey of related work on visual servoing and pose estimation using hand-mounted cameras and tactile sensing. **Recent work** on addressing these issues has used hand-mounted cameras to demonstrate superior performance in classical manipulation tasks such as grasping and bin picking[23]. With the availability of a visual perspective complementary to external (or head mounted) cameras, researchers have diversified the moving cameras to serve as tactile devices ([19, 20]) and have implemented delicate manipulation behaviors (see e.g. [21, 24]). Recent research has also developed tactile sensors and algorithms for estimating contact pose and inferring object from contacts [25, 26], tracking object motion by fusing externally mounted cameras and tactile sensors[27], and transferring information between external cameras and hand-mounted cameras (see e.g. [28]). A closely related work by [29] discusses integration of a visual and tactile measurement through a Bayesian filter.

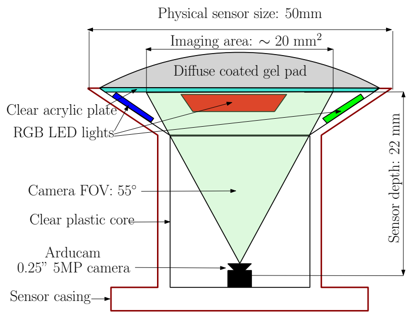
Vision-based localization and contact prediction: Camera-on-hand or more generally camera-on-mobile-agent arrangements have been investigated by several researchers to pursue diverse goals such as visual servoing to a workspace goal (e.g. [30]), collision avoidance systems on miniature aerial vehicles (e.g. [31]), and how flying insects, birds, and rapidly moving animals perceive motion[32]. Literature on quantitative analysis of looming[1] (e.g. [33], [34]) is of particular interest to us as we try to identify an area in the image space corresponding to the direction of heading of the robot at any particular moment. Recent research on optical expansion (e.g. [35]) is focussed on supervised learning, instead of hand crafted functions (e.g. [33, 34]) to compute dense scene flow from optic flow to identify relative motion of objects and the agents, and has been demonstrated to exhibit state of the art performance in identifying objects heading towards the agent. In the current work we build upon research on optic flow for scene understanding to identify an area in the robot's visual field corresponding to the physical point in the workspace where the robot is currently headed.

Tactile localization and contact estimation: Vision sensors can have a wide "field of view" and are good for making large scale models. A tactile sensor has a much smaller measurement area (or field of view) and can potentially capture minute details of the surface it interacts with. Early research on tactile sensing leveraged this capability, even with a seemingly low resolution tactile sensor (Weiss Robotics DSA9205), to demonstrate object recognition using image feature descriptors[36] and, recognize and localize an articulated object through a sequence of touches (see e.g. [37]). With the introduction of camera-based higher resolution tactile sensors, most notably the GelSight (see [20]) and its derivative GelSlim[38], investigations on tactile object recognition and localization have made significant progress in tactile sensing driven perception. [39] described tactile localization using the GelSight sensor using conventional feature based image alignment. [40] integrated the GelSlim with a gripper and interfaced it with a model based controller to successfully perform re-grasps of a cable. More recently, [41] demonstrated tactile localization and shape reconstruction using a GelSlim

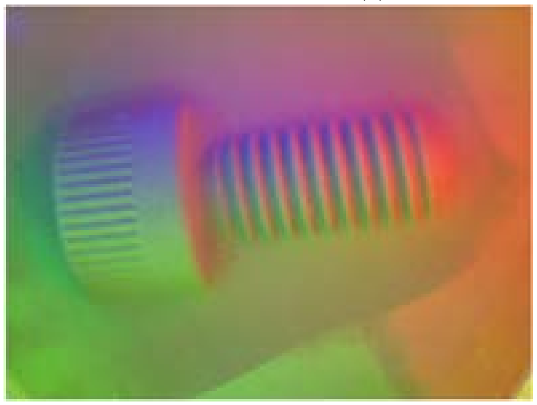
[1]Looming or visual looming is defined as the phenomena of an object getting bigger in the visual field as the relative distance between the observer and the object decreases.



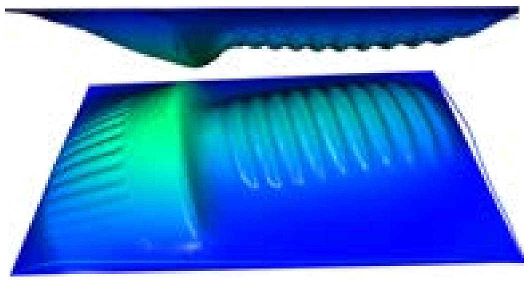
(a)



(b)



(c)



(d)

Figure 2.2: Figure 2.2a shows our sensor platform which is attached to the robot manipulator. The sensors, from left to right, are the Intel RealSense L515 LiDAR camera, our modified version of the GelSight and an RGB camera. The RGB camera and the RealSense are at a distance of 6 cm each from the camera to the left and right respectively. Figure 2.2b shows the schematic of our modified version of the GelSight, fig. 2.2c is an instance of the raw data collected by our GelSight when pressed against an 18mm long 6mm diameter bolt with 1mm pitch. Through the image, we note the physical scope of the sensor is almost entirely covered by the 18mm long bolt. We process the raw GelSight data to yield metric depth and normal maps. The depth of the reconstructed surface is rendered as a shaded point cloud in 2 views in fig. 2.2d.

sensor, where the authors trained neural networks to generate height maps with ground truth data obtained from robot experiments with the sensor and known objects. The trained network was then used to generate height maps of the surface of an object and the height maps were registered to reconstruct the object surface. This work was extended by [42] where the authors used a renderer to generate and cache a large number of possible tactile signals of objects from a data set touching a tactile sensor (GelSlim in this case) in different orientations. An actual tactile signal corresponding to a particular object at a particular pose was then compared with the cache to retrieve candidate object and pose pairs and the network demonstrated in [41] was then used to generate a height map which was registered with the candidate object at the candidate pose to localize contact. In [43] the authors describe a method to integrate a learned representation of a tactile signals (from a GelSight in this case) as a factor in a task of estimating object pose with a stream of subsequent measurements. [44] introduced a new time of flight based tactile sensor, the soft-bubble, and demonstrate object identification using learned embeddings and object localization by registering the tactile signal (obtained as a point cloud by their sensor) with the retrieved geometric model of the recognized object. Recently, and perhaps closest to the current work, [45] introduced a method to reconstruct the shape of an object by refining a coarse prior shape obtained by an RGBD sensor using tactile measurements from a GelSight attached to a robot. Also related to the current work, Dikhale et al.[46] demonstrated a method to fuse off-the-shelf learned pose estimators (PoseCNN [47] in this case) from external RGBD sensors and in-hand tactile sensor outputs to estimate the poses of grasped objects. In the current work we build upon literature on contact localization using high resolution tactile sensors, visual localization and hand mounted cameras to demonstrate a suite of collocated vision based sensors that can be used to visually servo a robot to touch and localize objects in the workspace.

2.3 Methods

In this section we describe our sensor platform (A) and algorithms to estimate where the robot hand will go (B1), visually servo the robot to a target contact point (B2), estimate the pose of the target object (C), and combine vision and tactile information to estimate both the contact point and refine the object pose estimate (D).

2.3.1 Sensor platform

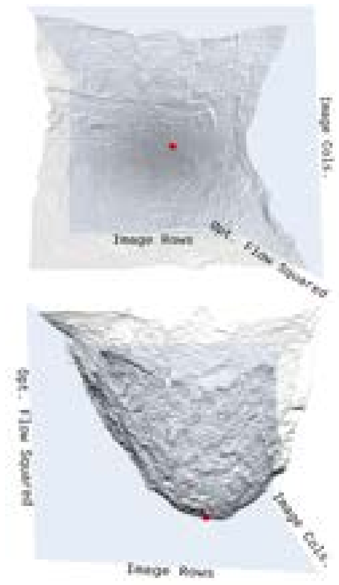
In this work, we collocate cameras with a camera-based tactile sensor by putting the cameras and the tactile sensor in close physical proximity while operating them independently. The sensor platform consists of a GelSight tactile sensor in the middle (co-incident with the robot wrist's axis) (figs. 2.1 and 2.2), with a camera on either side. We modify the GelSight as described by [48] in the physical sensor form factor introduced by [20]. We use a LIDAR-based RGBD sensor (Intel RealSense L515) with a 70° field of view to provide depth (figs. 2.1a and 2.2a left) and color images of the workspace, and a USB camera (a Sony IMX 291 sensor) with a wider 100° field of view lens (figs. 2.1a and 2.2a right). We chose these cameras partly because they were of comparable size to our GelSight sensor, thus ruling out most other popular RGBD cameras. Also, the USB camera yielded high resolution images which were better than the color channel of our 3D sensor and helped us localize small objects. Further details on the sensors can be found in fig. 2.2.

2.3.2 Visual servoing with hand mounted cameras

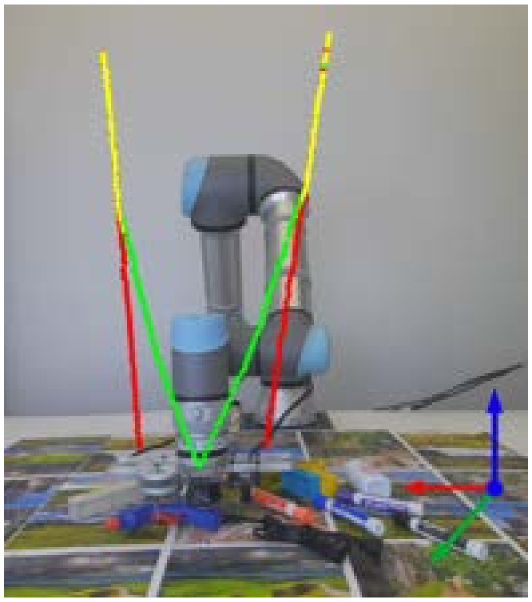
In this section we describe a method to estimate the robot hand's heading direction in the workspace and then we describe how to servo to a goal using that estimate.

Optical point of expansion from in hand cameras

We process the scene to identify the location of a 3D point corresponding to the heading direction of the robot in the image space. To achieve this, we calculate the optical flow between the consecutive frames obtained by the hand mounted cameras and identify the region in the image from which the optic flow seems to be emerging (i.e., we look for a portion of the scene which has zero translation)



(a)



(b)

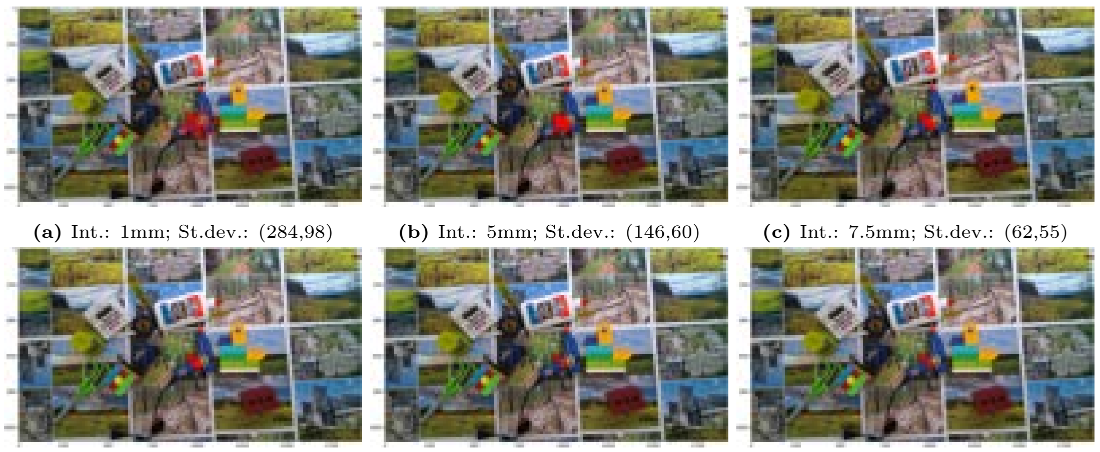
Figure 2.3: Figure 2.3a shows the surface of the squared magnitude of the optical flow between a consecutive frame pair in 2 views. We note that this surface assumes a parabolic shape. The red dot is the minima of the optical flow surface as identified by our algorithm. Figure 2.3b demonstrates the usage of our algorithm to correct trajectory errors using both cameras as shown in fig. 2.2a. The X, Y and Z axes are marked in red, green and blue in fig. 2.3b on the bottom right of the figure

as the camera moves towards the scene. Assuming that the world scene is relatively flat (object depth $\ll$ projection depth), the square of the magnitude of the optic flow at each pixel is roughly distributed as a parabolic surface (see fig. 2.3a). We calculate the motion field (per pixel optic flow magnitude and direction) between two consecutive frames using the OpenCV implementation[49] of the Lucas-Kanade dense optical flow, which solves for per pixel motions (along horizontal and vertical directions) over the full image. We also tested the Farenbäck optical flow [50] and the Brox optical flow [51], and found that the dense Lucas-Kanade optical flow performs slightly better in computation speed and produces smoother optic flow fields. The optical point of expansion (POE) is obtained as the minima of the surface representing the square of the magnitude of the optic flow. We use a robust algorithm described in algorithm 1 to detect the POE.

Algorithm 1 Calculating the point of expansion

```
 1: procedure CALCULATEPOE (image stream $\{I\}$)
 2:     Make pointers list $\{L\}$ for overlapping image tiles
 3:     for $(I_t,\ I_{t-1}) \in I$ do
 4:         $\{d_x, d_y\} \longleftarrow$ denseLKOpticalFlow$(I_t,\ I_{t-1})$
 5:         $d_t \longleftarrow d_x^2 + d_y^2,\ \theta_t \longleftarrow \arctan(\frac{d_y}{d_x})$
 6:         $d_t \longleftarrow$ normalizeAndHistogramEqualize$(d_t)$
 7:         for tile $\in L$ do                              ▷ exec. with n workers
 8:             if Histogram($\theta_t$[tile]) is uniform then
 9:                 tile$\rightarrow L_{ang}$
10:             end if
11:             add one random tile to $L_{ang}$                ▷ if tracking fails
12:         end for
13:         for tile $\in L_{ang}$ do                         ▷ exec. with n workers
14:             fit $\leftarrow$ Least sq. fit paraboloid to $d_t$[tile]
15:         end for
16:         choose $k$ best fits from fit $\rightarrow$ best fits
17:         find pixel $P$ of minima of the fit $\in$ best fits
18:         POE $\leftarrow P$ with the most uniform Histogram around $\theta_t$[P])
19:     end for
20:     return list of POEs per frame pair
21: end procedure
```

The heading direction estimated by the optical flow between consecutive frames was too noisy to yield meaningful heading estimates. To address this, we looked at the trajectory correction estimates for each of the on hand cameras and found their prediction to be very closely correlated (almost equally incorrect or equally correct), which led us to rule out camera noise and incorrect robot kinematics including the camera mounts as the cause of the noise. We concluded that the errors were being caused by small unmeasured rotations of the robot wrists (play or backlash). Numerical modeling showed that the expected amount of play in orientation led to optic flow values comparable to the "noisy" shifts in the POE. We also noted that the noise in the predicted trajectory errors was centered about zero. We averaged POE shifts over a robot travel of at least about 10 mm to get usable POE estimates. For our robot setup. where a maximum of 1m downward travel



(a) Int.: 1mm; St.dev.: (284,98) **(b)** Int.: 5mm; St.dev.: (146,60) **(c)** Int.: 7.5mm; St.dev.: (62,55)

(d) Int.: 10mm; St.dev.: (19,27) **(e)** Int.: 15mm; St.dev.: (21,20) **(f)** Int.: 20mm; St.dev.: (17,15)

Figure 2.4: In this experiment, we move the robot vertically down by 65 cm to a goal location slightly below the yellow cross mark on the handle of the glue gun. There are no errors in the goal location being tracked for this case. This experiment is repeated 10 times. The red dots are the predictions of potential point of contact (calculated as the instantaneous POE). We note that the predictions are centered about the actual point of contact – a point slightly below the yellow cross mark on the glue gun. We report 6 cases where we predict the potential point of contact by looking at various intervals of the trajectory. We report the interval lengths (in mm) and the standard deviation in predicting the point of contact (in pixels) as the labels of the figures. We note that as we increase the length of the interval, the standard deviation of the prediction decreases (as seen through "clumping" of the predicted potential points of contact), but the number of possible predictions decreases (as seen through fewer number of red dots with increasing interval sizes). This leads us to conclude that the averages across larger temporal (and spatial) windows produce smoother and more stable error signals (or correction signals in the case of visual servoing). For our use case, averaging across 10 mm intervals provided us with "enough" number of correction signals while having reasonably low variance. Quantitative results of the convergence of the estimates are shown in fig. 2.5.

was possible, empirically we observed that about 10cm non-overlapping intervals provided useful trajectory corrections and provided the opportunity for several corrections as the robot moved to the target. We describe the experiment that led us to this conclusion in fig. 2.4.

Correcting trajectory errors using pixel space errors

In the previous section we described a method to identify the image coordinates of the point in the workspace to which the robot is headed. In this section we address the problem of correcting trajectory errors using those predictions. This can be useful when the robot's trajectory needs correction and the only information about the updated goal is available in pixel space – possibly from a hand mounted RGB camera which detected a movement of the target. We use the following intuition for visual servoing: to reach a pre-planned goal in the robot's workspace, the robot, almost always, needs to move towards that goal, hence the image of the goal point and the observed POE should be very close. Errors in the pixel space between these two, indicates an error in the trajectory being followed by the robot, which we aim to correct. In more precise terms, let us denote the true workspace goal as $\mathbf{X}_G^W$ and it's estimate $\widehat{\mathbf{X}_G^W}$ in homogeneous 3 space. The robot is initially planned to move to $\widehat{\mathbf{X}_G^W}$ and let the POEs obtained for the subsequent frames be $\mathbf{p}$, in the pixel space. As the camera is registered to the robot, a given point in time, we can calculate the world to camera

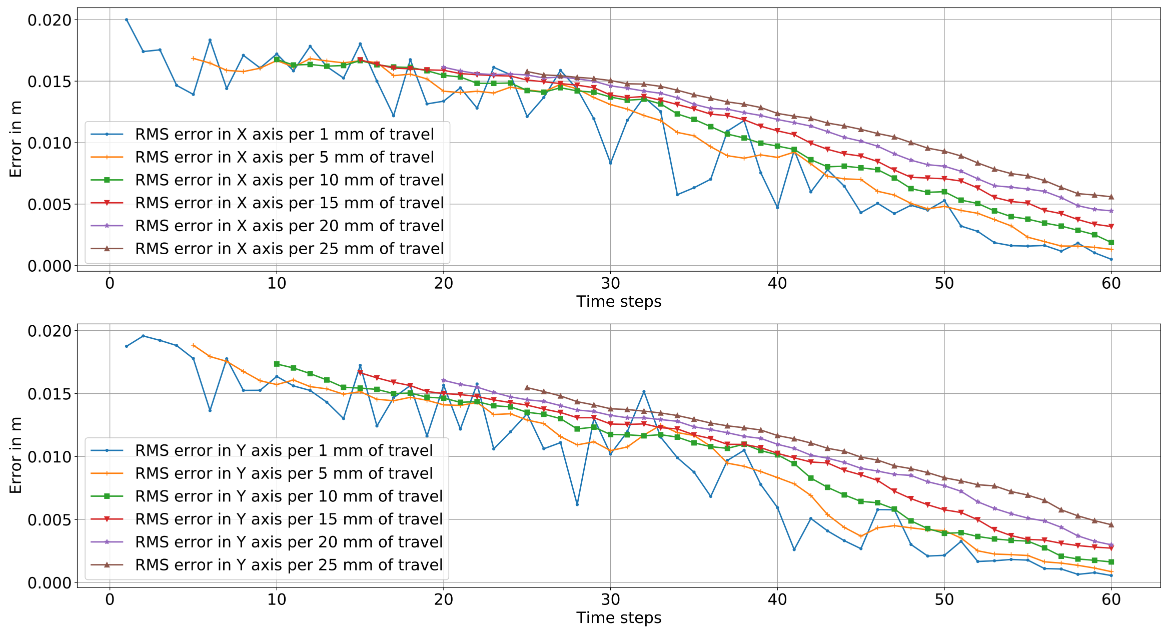


Figure 2.5: We present the quantitative results in estimating the POE with different averaging window width from fig. 2.4. An averaging window of 10 mm returns the lowest end-point error with the least variance.

projection matrix as $\mathbf{P} = \mathbf{K}[\mathbf{R}^{cam}_{world}|\mathbf{t}^{cam}_{world}]$ and can project $\mathbf{X}^W_G$ and it's estimate $\widehat{\mathbf{X}^W_G}$ to $\mathbf{x}_G$ and $\widehat{\mathbf{x}_G}$ in homogeneous 2 space. If the motion between two frames captured by the camera-in-hand is mostly perpendicular to the imaging plane, we can approximate $\widehat{\mathbf{x}_G}$ with the point of expansion $\mathbf{p}$ for each subsequent frames. We also note that, as $\mathbf{p}$ is in the pixel space, with this approximation, we discard the "z-buffer" of the projective transformations associated with $\mathbf{x}_G$ and as a consequence, cannot correct for a error in the camera's projective axis (camera's z axis), unless we reconcile the estimates of the "z-buffer" by triangulating the POE from the two camera in-hand views. We skip that analysis for simplicity here, but we implemented this behavior when performing the experiment in fig. 2.3b. We use the camera projection Jacobian and calculate the task-space error as eq. (2.1)

$$\delta = \begin{bmatrix} \frac{f_x}{Z_W} & 0 & -\frac{X^W_G f_x + Z^W_G c_x}{{Z^W_G}^2} + \frac{c_x}{Z^W_G} \\ 0 & \frac{f_y}{Z^W_G} & -\frac{Y^W_G f_y + Z^W_G c_y}{{Z^W_G}^2} + \frac{y}{Z^W_G} \end{bmatrix}^+ \begin{bmatrix} p^u - \mathbf{x}^u_G \\ p^v - \mathbf{x}^v_G \end{bmatrix} \tag{2.1}$$

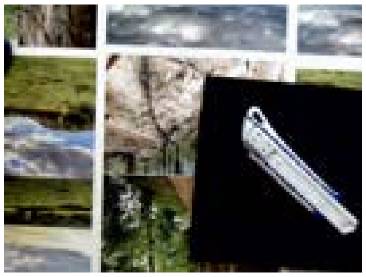
In eq. (2.1), $[f_x, f_y, c_x, c_y]$ are the camera intrinsics, and we denote $\mathbf{X}^W_G = [X^W_G, Y^W_G, Z^W_G]$ and the $[\cdot]^+$ denotes the Moore-Penrose pseudo-inverse. As an example use case, we use the following procedure to correct trajectory goals as the robot moves approximately 1m towards the true workspace goal $\mathbf{X}^W_G$ with a erroneous initial estimates of upto ±10cm in each of X and Y directions.

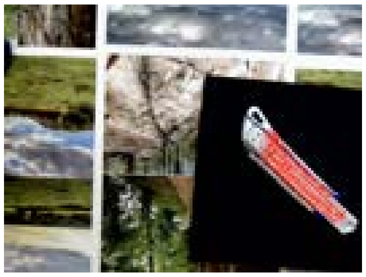
- steps 1–20: Plan a path with 100 waypoints till goal, and execute the first 20 steps.
- steps 21–40: For the next 20 steps, calculate the POE and then use the camera intrinsics and pose of the camera with respect to the robot base to obtain the mean error in trajectory space.

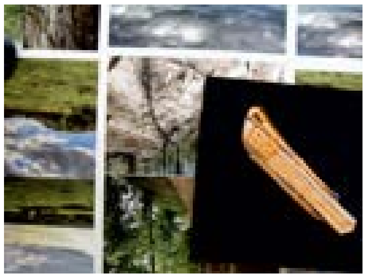
Correct the trajectory error and plan for the remaining 60 steps.

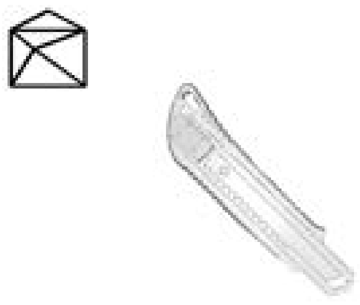
- steps 41–60: Execute the current path re-planned at step 41, do not track the point of expansion or generate new trajectory error estimates.
- steps 61–80: Repeat steps 21–40 while moving through the corrected trajectory, at the end correct trajectory again and re-plan if necessary.
- steps 81–100: Continue with the last planned trajectory, and stop if goal is reached to given tolerance.

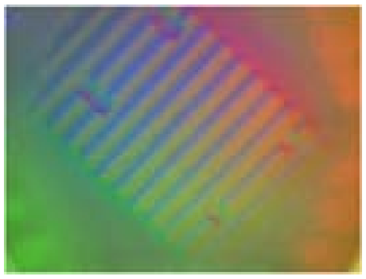
In fig. 2.3b, on the two sides of the robot, we show 2 trajectories where the robot travels about 1m (vertically) from the start to the final contact position, and each of the trajectories need a correction of 10cm errors in the X and Y directions in the robot workspace. We show the initial portions of the trajectories in yellow, the planned portions in red, and the corrected portions in green. The final accuracy achieved was within 5mm of the target.


(a)


(b)


(c)


(d)


(e)

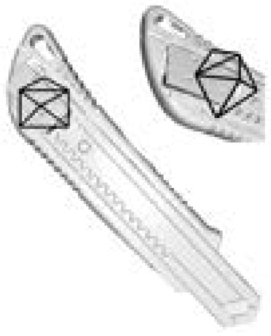

(f)

Figure 2.6: Our pipeline to estimate pose through vision and touch. Figure 2.6a is the color channel of the RGB-D sensor stream captured at a particular point in the trajectory. Figure 2.6b is the pose estimate obtained after solving eq. (2.2), which is refined by solving eq. (2.4) and we obtain a camera pose estimate $\hat{\zeta}$ (shown in fig. 2.6c) which is mostly correct in the camera projection depth and camera yaw. We transfer $\hat{\zeta}$ to the reference frame of the GelSight and obtain fig. 2.6d showing the relative pose of the object and the GelSight. This corresponds to the pose $\hat{\zeta}_{GS}$. This estimate is further refined by minimizing eq. (2.5) using the GelSight data (fig. 2.6e) obtained at contact, and we obtain the final pose shown in fig. 2.6f

2.3.3 Pose estimation through vision

In this section we discuss visual localization of an object with known 3D geometry. At least in the early stages of approach, the hand-mounted cameras can see all or large portions of the target object, and standard image based registration methods ([52, 53]) can be used to estimate the object's pose relative to the robot hand. This is quite different from the situation with tactile sensing where only a small part of the object is visualized. As is common in the whole-object-visible camera-based pose estimation literature (see e.g. [52, 53] and [54]), we decompose pose estimation into two parts – coarse pose estimation by aligning the centroids and edge moments, and finer alignment using a distance-based cost applied densely.

As a prerequisite for this part we need the edge pixels of the object in the image and for this we use either the depth edges on the object if available or else use the image gradient edges of the object – we use a Canny edge finder for this purpose. Let us denote this binary edge image as $\mathbf{I}_S$. Next, we formulate an optimization problem to identify an object pose that produces the most similar edge distribution to $\mathbf{I}_S$.

To do this, we generate an initial guess for θ (the angle of rotation about the camera projection axis) for the object from the principal components of the camera image and if an aligned depth map is available, we use its mean to initialize the z component (distance between the camera and object along the camera projection axis), or else we initialize the z depth arbitrarily. The x and y directions along the image plane are initialized by back-projecting the centroid of the edge pixels using the camera intrinsics and the initial value of z. With the initial guess $\omega = [x, y, z, 0, 0, \theta]$, we use a differentiable renderer (we use a modified version of the DIRT renderer from [55]) to render the mesh model of the object, extract the corresponding edge image $\mathbf{I}_R(\omega)$ and solve the following minimization to obtain a rough pose estimate from the camera image. For each $\mathbf{I}_R$, we identify the image edge pixels $\mathbf{p}_R^i$ and $\mathbf{p}_S$ for $\mathbf{I}_R$ and $\mathbf{I}_S$ respectively and minimize the following sparse edge matching cost $\mathbb{E}_{cs}$ in eq. (2.2)

$$\begin{aligned}\mathbb{E}_{cs}(\omega) = \gamma\,[||\bar{\mathbf{p}_R}(\omega) - \bar{\mathbf{p}_S}||_2] + (1-\gamma) \\ \langle \mathcal{V}\,(\mathbf{p}_R(\omega) - \bar{\mathbf{p}_R}) \cdot \mathcal{V}\,(\mathbf{p}_S - \bar{\mathbf{p}_S})\rangle\end{aligned} \tag{2.2}$$

where, $\mathcal{V}(\mathbf{p})$ is the direction of largest variance of the mean centered point set $\mathbf{p} \in \mathbb{R}^2$, given by the eigenvector corresponding to the maximum eigenvalue, $\langle \cdot\,, \cdot \rangle$ is the dot product between vectors and γ is a weighting factor. Minimizing eq. (2.2) aligns the centroid and approximately recovers the angle of rotation along the camera projection axis.

The expression for $\mathbb{E}_{cs}$ does not admit automatic gradient calculation due to the non-differentiable selection of pixel indices to obtain $\mathbf{p}_R$ from $\mathbf{I}_R$, therefore, we obtain a finite difference gradient using central differences. We minimize $\mathbb{E}_{cs} \forall \theta_i$ and obtain the candidate pose parameters $\hat{\omega} = \{X_{cs}, Y_{cs}, Z_{cs}, \theta_{cs}\}$ in the camera coordinate frame, corresponding to the minimum E_{cs}. From fig. 2.6b we note that minimization of $\mathbb{E}_{cs}$ is not expected to solve for the projection depth, as it only recovers the orientation of the camera and not the projection depth. In the next part, we solve the projection depth by minimizing a modified version of the dense differentiable cost using the directional chamfer matching energy as discussed in [52] and [53] (eq. (2.3)).

$$\mathbb{E}_{cm}(\mathbf{I}_S, \mathbf{I}_R^{\zeta}) = \sum_{^{\xi}\mathbf{p}_s^i \in {^{\xi}\mathbf{I}_S}} \left[\min_{^{\xi}\mathbf{p}_R^j \in {^{\xi}\mathbf{I}_R(\zeta)}} ||{^{\xi}\mathbf{p}_S^i} - {^{\xi}\mathbf{p}_R^j}(\zeta)|| \right] \tag{2.3}$$

In eq. (2.3), the outer sum $\sum_{^{\xi}\mathbf{p}_s^i \in {^{\xi}\mathbf{I}_S}}(\cdot)$ implements the edge awareness by binning the edges according to their orientation (as quantized by $\xi \in [-\pi, \pi]$), and implicitly assigning edge pixel correspondences. We observe that the inner minimization problem $\min_{^{\xi}\mathbf{p}_R^j \in {^{\xi}\mathbf{I}_R(\zeta)}} ||{^{\xi}\mathbf{p}_S^i} - {^{\xi}\mathbf{p}_R^j}(\zeta)||$ for each ${^{\xi}\mathbb{E}_{cm}}$ can be solved by the Euclidean distance transform. As the ${^{\xi}\mathbb{E}_{cm}}$ cost is cumulative over the ${^{\xi}\mathbf{I}_S}$ image, the cost boils down to the pixel-wise sum of absolute differences between the Euclidean distance transforms (EDT) of images ${^{\xi}\mathbf{I}_S}$ and ${^{\xi}\mathbf{I}_R}(\zeta)$. So using the definition of Euclidean distance transform from [56] in eq. (2.3), we simplify our dense edge matching energy as

$$\mathbb{E}_{cm}(\zeta) = \sum_{\xi} \left[\sum_{\mathcal{G}} \left[\left| \mathrm{EDT}({}^{\xi}\mathbf{I}_{\mathrm{S}}) - \mathrm{EDT}({}^{\xi}\mathbf{I}_{\mathrm{R}}(\zeta)) \right| \right] \right] \tag{2.4}$$

We minimize this function with gradient descent to obtain a coarse pose estimate $\hat{\zeta}$ from the camera image and transfer the pose estimate to the GelSight camera frame as $\hat{\zeta}_{GS}$. In contrast to [52, 53], we implemented modified and differentiable versions of the matching costs and thus, our gradient steps are about 80% faster than the reference implementations of [52, 53] for the same size of the image. In practice, to keep the computational cost low, we maintain a coarse pose estimate using eq. (2.2) throughout the major part of the trajectory and switch to eq. (2.4) at a point beyond which the objects are de-focussed. For objects in figs. 2.1, 2.6 and 2.7 this distance was around 15cm and for objects in fig. 2.8 it was 10cm.

2.3.4 Precise contact pose estimation through touch

We found that our vision-based localization had errors on the order of a centimeter, due to our algorithms imperfect camera calibration, local minimums in matching, and sometimes a lack of visual features to match or track. In addition, as the hand comes near the object, the depth and image sensor measurements become unusable – the LiDAR based sensor provides reliable depth estimates at distances greater than 25cm, and the camera image measurements were usable at distances greater than 12-15cm to the object, after which the items of interest often go out of view and the image becomes too blurry to extract high quality edges needed by our 3D pose estimation algorithms. We find that using the tactile image on contact can improve contact point and pose estimation. In fig. 2.2 we noted that we could generate metrically correct depth and normal maps of the deformed GelSight surface at contact. In this section, we use that information, along with the pose estimated from the previous section to localize contact, given that we know the geometry of the object. To achieve this, we use multi-scale dense depth and normal map alignment to obtain the pose of object with respect to the tactile sensor.

We capture the tactile image in our GelSight's native camera resolution of (640×480) pixels and obtain depth and normal maps. We then decompose each of the normal and depth maps into 4 lower pyramid levels. We denote these 5 normal and depth maps of the source image as N_S and D_S respectively. Next, we render the object (using [55]) through the GelSight's viewport using the pose estimated in the previous section and obtain normal and depth maps corresponding to the frame sizes of N_S and D_S respectively.

We note here that this is not an exact simulation of the GelSight sensor through our renderer – the ideal GelSight should only measure objects touching it i.e. ∼25mm from the camera. Anything beyond or closer than that is either not touching the sensor or interfering with it. We relax this requirement, and also neglect the effects of the soft body contact between the gel and the object to get a better basin of convergence in the optimization problem. We denote these sets of rendered depth maps by $N_R(\hat{\zeta})$ and $D_R(\zeta)$. Our alignment cost function $\mathbb{E}_{gs}(\zeta)$ for the GelSight data is

$$\mathbb{E}_{gs}(\zeta) = \sum_{i=1}^{5} \left[\left| D_S^i - D_R^i(\zeta) \right| + \left[1 - \langle N_S^i \cdot N_R^i(\zeta) \rangle \right] \right] \tag{2.5}$$

where $\langle N_S^i \cdot N_R^i \rangle$ denotes the pixel-wise dot products of the normal maps. We solve this with $\hat{\zeta}_{GS}$ as the starting guess using gradient descent. Figure 2.6 describes the steps discussed above.

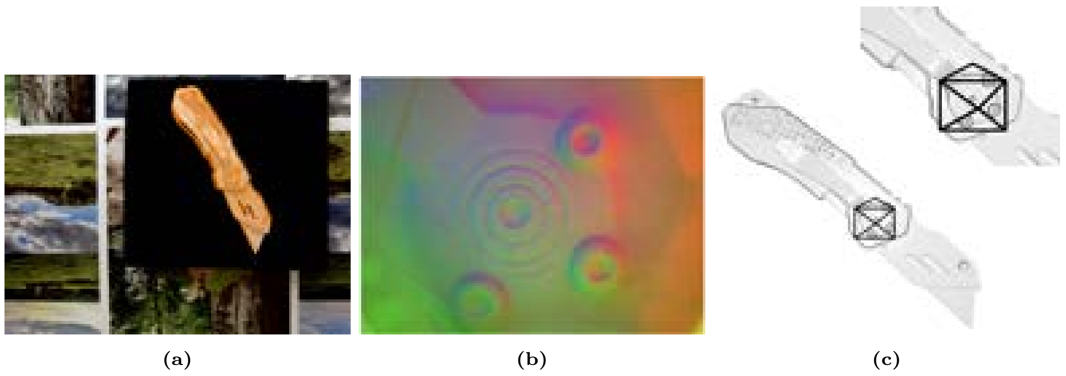


Figure 2.7: Figure 2.7a is the pose estimated using the color channel image. This pose estimate was initialized with coarse pose estimates obtained using depth images. Figure 2.7b is the GelSight data obtained at contact and fig. 2.7c is the camera pose at contact. Figures 2.1 and 2.6 contain similar results.

2.4 Results

In this section we present the results of localizing contacts using the sensor setup and the algorithms discussed in this work through 6 scenarios. For the experiments presented below, we use a black background to simplify the object segmentation and edge detection. Except for the glue gun, all the other objects had reflective parts and were painted matte white to remove specularities.

Localization with color and depth edges

Our RGBD sensor captures the depth of the scene reliably from 30 cm away and can produce aligned depth and color images. As the depth images are unchanged by changes in lighting and texture, extracting edges from depth images and maintaining coarse pose estimates as the robot moves closer to the object is a viable strategy to generate initial pose estimates which are later refined to estimate the pose at contact. For the objects described in figs. 2.1, 2.6 and 2.7, when the depth sensor output degraded (at 20 cm) we transferred the pose estimates obtained (and maintained) using the depth image, to the color image. refined the pose estimates from the final color image using the tactile image to obtain the pose at contact.

Localizing small, flat and thin objects

Depth cameras don't capture much in situations where an object has little depth variation. The limited field of view of most depth cameras means they lose sight of where the GelSight sensor will land as the hand approaches an object. For these reasons we collocated a high resolution large field of view RGB camera with the GelSight sensor, so the system would also work well with small thinner objects with less depth variation. In fig. 2.8 we demonstrate the localization of a slender metallic pin and a 4cm square circuit board.

Disambiguating tactile measurements with vision

With collocated vision, we can disambiguate between repetitive surface features, which would not have been possible with just tactile sensing. To show this, we set up an experiment (fig. 2.9) where the robot approaches the box cutter from above and touches around the middle of the slider. The tactile image recorded is shown in fig. 2.9b. The pose estimates obtained from the camera when

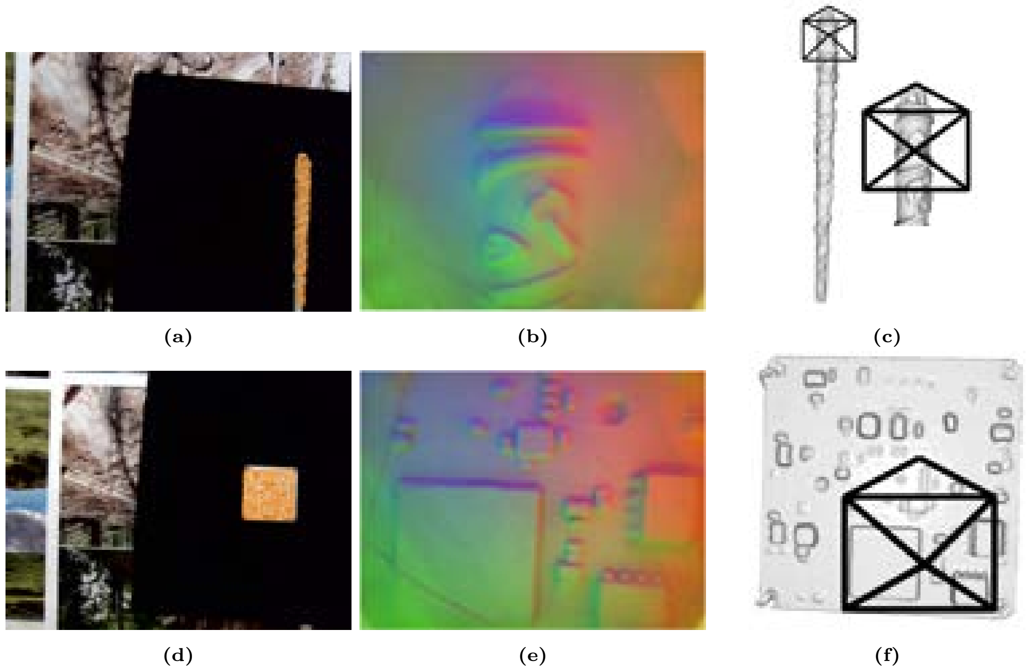


Figure 2.8: Figures 2.8a to 2.8c are the steps in localizing a slender metallic object, figs. 2.8d to 2.8f are the steps in localizing a miniature circuit board. L-R we generate a coarse pose estimate with just the color image, receive the GelSight image on contact and finally obtain the camera pose on contact. The object in fig. 2.8a is 10 cm long and 8.5 mm in diameter at the thickest part. The circuit board in fig. 2.8d is 38mm × 38mm square.

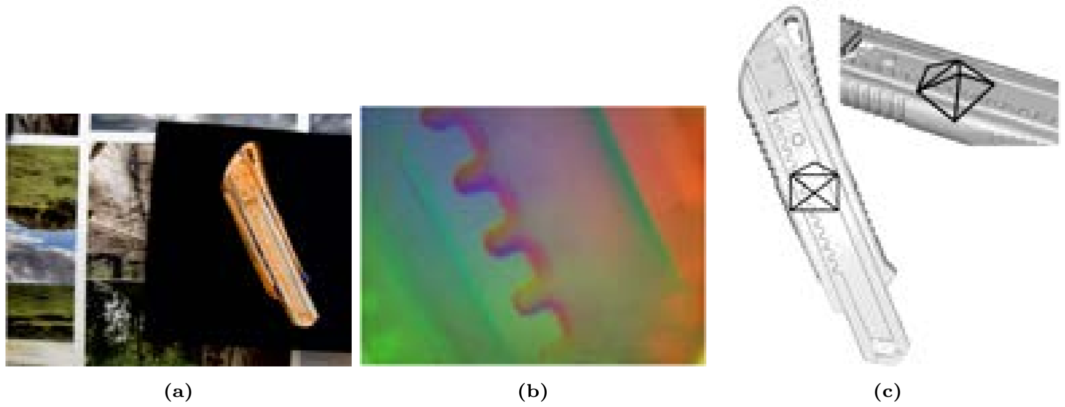


Figure 2.9: Collocated vision can help us disambiguate between repeated tactile features. In this experiment we touch the middle of the set of teeth and could localize the contact using our pipeline.

Object	ΔT^v_x	ΔT^v_y	ΔT^v_z	ΔR^v_x	ΔR^v_y	ΔR^v_x	ΔT^t_x	ΔT^t_y	ΔT^t_z	ΔR^t_x	ΔR^t_y	ΔR^t_x
Figure 2.6	2.56	1.70	5.47	0.52	0.38	0.17	0.71	0.84	0.60	0.50	0.37	0.18
Figure 2.9	0.69	0.57	3.72	0.13	0.10	0.35	6.75↑	1.90↑	0.92	0.14↑	0.06	0.30
Figure 2.7	2.39	3.49	8.45	0.34	0.23	0.27	0.83	1.49	2.45	0.35	0.10	0.23
Figure 2.1f	1.07	1.09	8.32	0.11	0.15	0.22	2.12↑	2.27↑	1.82	0.08	0.11	0.16
Figure 2.8f	0.29	2.25	2.27	0.39	0.24	2.11	0.89	0.33	1.07	0.35	0.27↑	0.51
Figure 2.8c	0.68	0.76	0.62	0.01	0.01	0.04	0.30	1.46↑	0.01	0.002↑	0.0057↑	0.04

Table 2.1: Relative uncertainties in localization using only vision and tactile sensing augmented by vision, in case good tactile signals are guaranteed. ΔT_x is the standard deviation in localization along the x axis of the robot and ΔR_x is the standard deviation in orientation along the x axis of the robot. Same naming convention applies to the y and z axes. The translation uncertainties are in mm. and the rotation uncertainties are in degrees. Superscript $[\cdot]^v$ indicates localization using vision only and $[\cdot]^t$ indicates localization using vision and touch. Lower values are better in all cases. The position uncertainties are in mm and the angle uncertainties are in degrees.

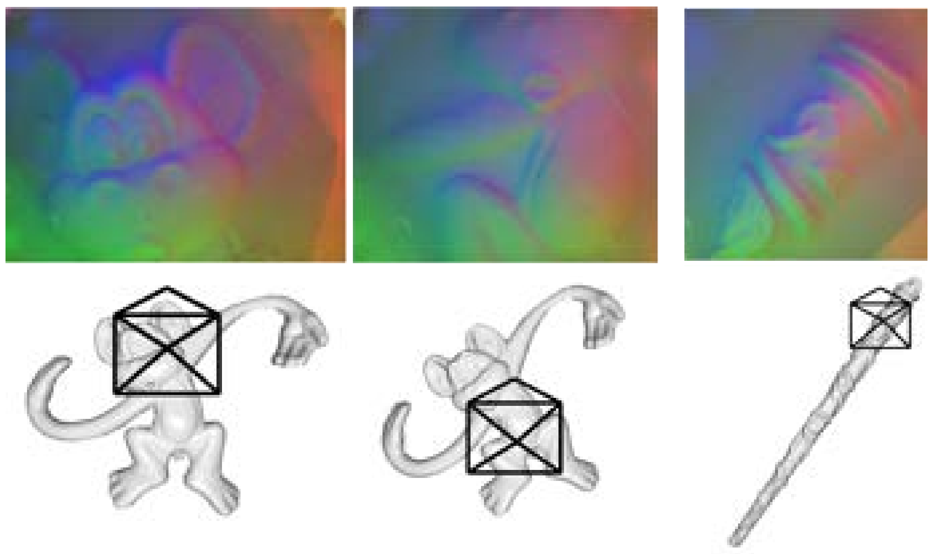
transferred to the GelSight yield useful initial guesses which can then be refined with the tactile data to localize the contact.

Estimating object pose with vision and touch

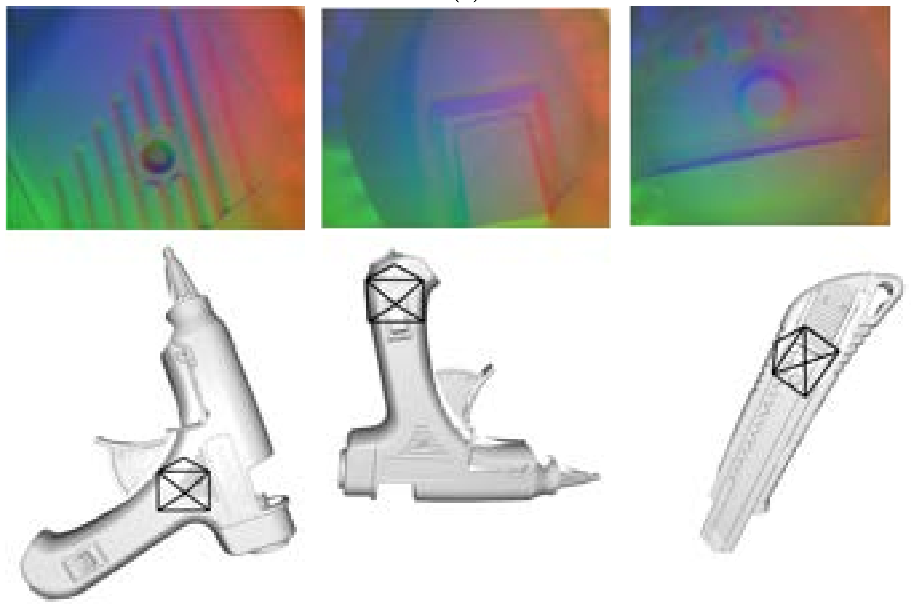
In this section we present our results on localizing objects with vision and touch, and evaluate the performance of our localization pipeline. We focus on the case where the contact point has a unique arrangement of useful tactile features. This is the best case for a combined approach. In practice, if the tactile sensor contacts a featureless area, we can detect and ignore the output of the tactile sensor. Most work in this area specializes for a set of objects used in a training set for a learning approach. We did not find work that combined a camera with a high resolution tactile imaging sensor for localizing objects upon contact, so there isn't an obvious method to compare to. This along with our choice of smaller and highly textured objects preempts easy comparison with recent learned localization approaches (e.g. [41, 42, 47] etc.). Instead, we present overall results of accuracy experiments below.

For each of the 6 objects used in this work (figs. 2.1f, 2.6, 2.7, 2.8c, 2.8f and 2.9), we fix the object to the robot table. Assuming that there is zero error in position control of our robot (we use a Universal Robots UR5E manipulator fixed to a vention.io table of recommended design for the same robot), we register the object with respect to the robot base and treat this pose as our ground truth. Next, we move the robot vertically 1 m above the object and move the robot down to touch the object at a chosen point that will yield good tactile information, and localize the object with respect to the robot. We repeat this 3 times for the same object positions and repeat this experiment for 2 more positions of the object with respect to the robot – i.e. localizing each object 9 times with respect to the robot. Fixing the objects is a restrictive assumption in the context of localizing objects especially with touch, however, to ensure repeatability of the experiments reported in the section, we had to fix the objects to a rigid base. Following [52, 53] we report the repeatibility of our pose estimation pipeline as the measure of its performance.

Using tactile sensing the localization errors were brought down to ± 1.5mm in translation and $\pm 0.5°$ in rotation from about 1.5cm in translation and 2° in rotation using only vision. However, for cases where the tactile features were not unique, e.g. the box cutter teeth (fig. 2.9) and the metallic object (fig. 2.8c), the tactile sensing actually increased the localization errors in the horizontal directions. The order of these errors were equivalent to the scale of the repeated features – 5mm for the experiment described in fig. 2.9c (the box-cutter teeth are about 3mm wide placed in intervals of 5mm) and about 2mm for the experiment described in fig. 2.8c (the embossed features are very similar at intervals of 3.5 mm). This observation is consistent with the fact that the final gradient descent step (eq. (2.5)) to refine the camera based pose estimates will converge to the wrong local minima if the tactile measurements are not distinctive enough. However, for objects with rich tactile features, using tactile sensing assisted with vision provides better localization than exclusively using either as we show in section 2.4. We repeated a subset of the experiments reported above where we



(a)



(b)

Figure 2.10: Additional demonstrations of our algorithm presented in sections 2.3.3 and 2.3.4 for localizing contact from high-resolution tactile imprints.

first corrected the robot trajectory using the procedure described in section 2.3.2 – and observed similar localization performance. We present the results of the experiment reported in this section in table 2.1.

Effect of points of contact on localization accuracy

In this section we present the effect of randomly selected contacts for localization. For this set of experiments, we fix each of the objects and the black background plate used in section 2.4 to a

Object (exp. name)	G.T. (x,y)	Vis. only (x,y)	Vis. & touch (x,y)	Comment
Box cutter best	41 (0,41)	50 (11,49)	45 (5,45)	–
Box cutter worst	78(0, 78)	89 (40, 79)	55 (41,35)	**C1**
Metallic pin best	50 (0,50)	40 (5,40)	53 (6,52)	–
Metallic pin worst	61 (0,60)	65 (8,64)	89 (10,88)	**C2**
Glue gun best	24 (24,0)	15(12,8)	26 (25,8)	–
Glue gun worst	26 (24,10)	21 (20,6)	63 (60,23)	**C3**
Monkey best	13(6,12)	9 (7,6)	15(8,13)	–
Monkey worst	14 (14,4)	18 (17,6)	36 (10,35)	**C4**

Table 2.2: Best and worst case localization errors when the contacts are selected at random. The ground truth measurements are calculated from the starting pose by reading the changes in X and Y translations on the graduated compound slide, the third column reports the magnitude of the translation recovered using vision only, and the fourth column reports the same using vision and touch. The X and Y components of the motion are indicated inside the brackets. We note that, even in the worst case, including the tactile measurements did not predict the contact in the wrong direction. In the last column, we provide possible explanations for the worst case errors as comments – **C1:** Poor tactile signal due to interference, **C2:** Partial contact between object and table, **C1:** Tactile sensor touched flat part, and **C4:** Tactile sensor touched edge of arm and part of table. All measurements have been rounded to mm. The errors in orientation are within 2° - 3° for the cases when the pose estimation was successful and have not been reported. Figure 2.10 presents some of the successful localization results.

graduated compound slide capable of in plane translation and rotation. We then moved the robot vertically down to make contact at the same location on the object used for the experiments reported in section 2.4 to generate a starting pose. Next, we generated 5 random configurations per object in translation and orientation on the plane of the table and moved the robot vertically down to touch the object and attempted to recover the randomly generated pose perturbations we introduced. For each of the objects, as expected, we observed similar errors in localization using only vision as reported in section 2.4. For the box cutter (fig. 2.6) most of the contacts yielded useful tactile signals so the errors in recovering the perturbations in pose were in the range reported in section 2.4 – i.e. $\sim$ 8mm in translation and $\sim$ 1.5° in rotation. This observation was also consistent for the smaller textured objects[3] (fig. 2.8). However, tactile sensing was not always helpful in localizing the objects – for the glue gun (fig. 2.1) and the folding knife (fig. 2.7, significant parts of the object were featureless and the tactile signals obtained when touched at these parts were unusable in localizing the objects as the final gradient descent step (eq. (2.5) in section 2.3.4) re-introduced localization errors of about 3-4 cm and 15° by converging to incorrect poses. We report the results of the experiments described in this section in table 2.2.

Localizing contact using only vision

We explored two approaches for predicting contact with only vision – an image-based approach that tracks the POE similar to visual servoing and a 3D geometric approach that can estimate contact based on a single image captured before the object gets out of focus or gets mostly out of the frame.

In the **first case**, we identify the POE as described in section 2.3.2. Based on the known offset between the camera and the Gelsight, we can identify the predicted GelSight contact point in any camera image. This approach works best when the hand is moving straight to the contact point (no trajectory corrections or going around obstacles), there are a lot of visual features to track, and the hand starts high enough so there is little offset of the object and the background caused by the different distances to the object and the background. We tested performance with the data from the previous section for objects in figs. 2.1f, 2.6 and 2.9. We could predict the location of contact with 1cm accuracy.

In the **second case**, we took the final usable image in these trajectories and used the procedure described in section 2.3.3 to obtain the pose of the object with respect to the RGB camera. Next, we transformed the pose of the object to the frame of the GelSight camera and simulated moving blindly until the vertical distance between the object and the GelSight is 3cm (the gel surface is

2.5 cm away from the GelGight camera), and reported the location of contact. We compared these estimates of the point of contact for all the objects we used. We found that for experiments with larger objects (as shown in figs. 2.1f, 2.6, 2.7 and 2.9) the average error between the estimated true location of contact and the location estimated by blindly moving downwards in the frame of the GelSight camera was 3.5 cm, whereas the same metric for experiments with smaller objects (as shown in fig. 2.8) were about 1.5 cm. The experiments with the smaller objects have lower errors because the distance between the GelSight and the object at the point the pose of the object was measured with only vision was much lower (8-10 cm) than the experiments with the larger object where the distance was about 25-30 cm, so the length of the blind descent was smaller for the smaller objects and hence the accumulated error in predicting the point of contact was smaller.

To localize contact with only single instances of touch, To localize contact with only a single tactile image, we obtained a normal map and point cloud from the tactile image. We rendered a mesh model of the object (e.g. fig. 2.9c) in a stable configuration, with the surface being touched facing the camera. The mesh is colored according to the normals with respect to a simulated camera, which is looking at the object vertically. Then we captured a high resolution image of the colored mesh and used ORB and SIFT features with FLANN based matching (ratio = 90%) to find correspondences between our high resolution image of the rendered mesh and the normal map from the image. We also tried matching the raw images captured by the sensor (e.g. fig. 2.9b) with the corresponding image of the geometry (e.g. fig. 2.9c). Finally, we rendered the mesh as a point cloud and used point cloud feature matching algorithms (B-SHOT[57] and FPFH[58]) to generate correspondences between the template object and the tactile measurement processed as a point cloud, and identify the point of contact on the object. None of the above experiments were able to generate reliable correspondences between the object model and the tactile image. We observe that in the absence of good initial pose estimates from an external sensor or learned embeddings (e.g. [42]), the portion of the object observed by the GelSight is often not large enough for conventional feature matching algorithms to work. Additionally, the GelSight elastomer mechanically operates as a nonlinear low pass spatial filter and does not preserve the true shape of depth discontinuities. Cliffs become gradual slopes, and the inverted surface is filtered quite differently. As a result, point cloud feature matching often fails to match the sharp surface features from the object model to the data captured by the tactile sensor. The GelSight sensor elastomer mechanics need to be simulated to provide a better template to match. To localize contact with only a single tactile image, we attempted to match the processed tactile image obtained from the modified GelSight (fig. 2.2b) to a representation of the object model. First, we treated the tactile signal as an image (fig. 2.2c) and extracted image features (SIFT and ORB) and attempted to match those features with a canonical image of the object mesh model. Next, we processed the raw GelSight data into a point cloud (fig. 2.2d) and used local point cloud feature matching algorithms (B-SHOT[57] and FPFH[58]) between the processed tactile signal and the object mesh (e.g. fig. 2.6f). None of the above experiments were able to generate reliable correspondences between the object model and the tactile image which leads us to conclude that in the absence of good initial pose estimates from an external sensor or learned embeddings (e.g. [42]), the portion of the object observed by the GelSight is often not large enough for conventional feature matching algorithms to work.

2.5 Discussion and future work

We focused on objects with significant surface tactile texture that our small tactile sensor could image. This led us to not use existing sets of objects for manipulation research (YCB[59], McMaster[60]). This different choice is due to the absence of a dataset to benchmark tactile imaging and vision working together. It is unclear what objects should be in the dataset, how approaches trained on such a dataset would generalize to novel objects and how much effort is needed to introduce new objects into the dataset and pipelines based on them. In our approach, we used an off the shelf 3D scanner (EinScan-SE[61]) to generate 3D models of objects and reference poses for our initial estimates (see section 2.3.3). We do expect that versions of the current state-of-the-art edge based localization pipelines (what we use) to be inherently slower than learned pipelines for generating initial pose estimates (e.g. [42] or [47]), but we believe that our localization pipeline would perform well for objects we have not tested here. Also, in the current literature, we did not find explicit or implicit feature transforms invariant across tactile images (or processed tactile data) and visual images. Although there exists work on learning features (see [28, 41, 42]), using them for explicit correspondences to match tactile data to visual data is relatively unexplored and we consider promising future work.

While integrating visual sensors of different capabilities, we noted that having optical sensors with almost co-incident optical axes would trivially solve some of the issues we faced while localizing small objects and disambiguating possible localizations. Collocating an ensemble of cameras of different capabilities (similar to smart phone multi-camera systems) creating a synthetic vision system is a natural next step of this work. Such a system could create a virtual fovea focused on where the tactile sensor could contact the object and bring down the overall footprint of the sensor setup so that it can be efficiently collocated with a standard gripper. Using a gripper with the ensemble of sensors is also future work. The GelSight data can be inexpensively processed to obtain surface normals of objects with respect to a fixed frame. This is valuable for pose estimation and should be emphasized in future object shape representations and manipulation algorithms. Naturally, computing object surface normals by controlling illumination at the scale of the robot workspace is also future work.

2.6 Conclusion

In this work we show that collocating cameras with tactile sensors on a robot hand in many cases enables tactile localization to work where vision only or tactile sensing only localization may perform badly or fail. We demonstrated this with common objects without substantial pre-computation and without using a known and fixed set of object models (learning). Vision can also help disambiguate tactile image matching, especially in cases of limited, repetitive, or otherwise ambiguous tactile features. Tactile sensing almost always improves visual localization as well. We also found that using optic flow from hand mounted cameras had to be integrated across about 10mm of camera travel to provide useful heading estimates, and could be used to correct trajectories.

2.7 Retrospectives

The main body of this research was conducted between 2020 May through 2021 May, with limited access to sensor fabrication and robot experiment facilities. However we should note the following:

1. To demonstrate a simple foveated vision system, we integrated two cameras of varying focal lengths (section 2.3.1). This is counter-intuitive considering that the human foveated vision system consists of two eyes which are largely similar. Robot mounted foveated-stereo systems (see e.g. [62]) also consist of two similar cameras, simplifying the two-view geometry between the cameras. We chose the cameras particularly because of their form factor – they had to be lower profile than the GelSight sensor and the overall profile of the sensor ensemble has to be lower for better maneuverability. Although the scope of our third project is much different, we took the lessons we learnt from designing this device for the stereo system for that project.

2. We observed that for the front facing scenes with objects lying on a table, dense depth was not crucial for our pose estimation pipeline, the precision in pose estimation was directly dependent on the resolution of the image we could process, further indicating that a binocular stereo would have been a better choice.

III

Moving Lights

3 Photometric stereo for manipulation of Lambertian objects

Controlling illumination can generate high quality information about object surface normals and depth discontinuities at a low computational cost. In this work we demonstrate a robot workspace-scaled controlled illumination approach that generates high quality information for table top scale objects for robotic manipulation. With our low angle of incidence directional illumination approach we can precisely capture surface normals and depth discontinuities of Lambertian objects. We demonstrate three use cases of our approach for robotic manipulation. We show that 1) by using the captured information we can perform general purpose grasping with a single point vacuum gripper, 2) we can visually measure the deformation of known objects, and 3) we can estimate pose of known objects and track unknown objects in the robot's workspace. Additional demonstrations of the results presented in the work can be viewed on the paper website.

3.1 Introduction

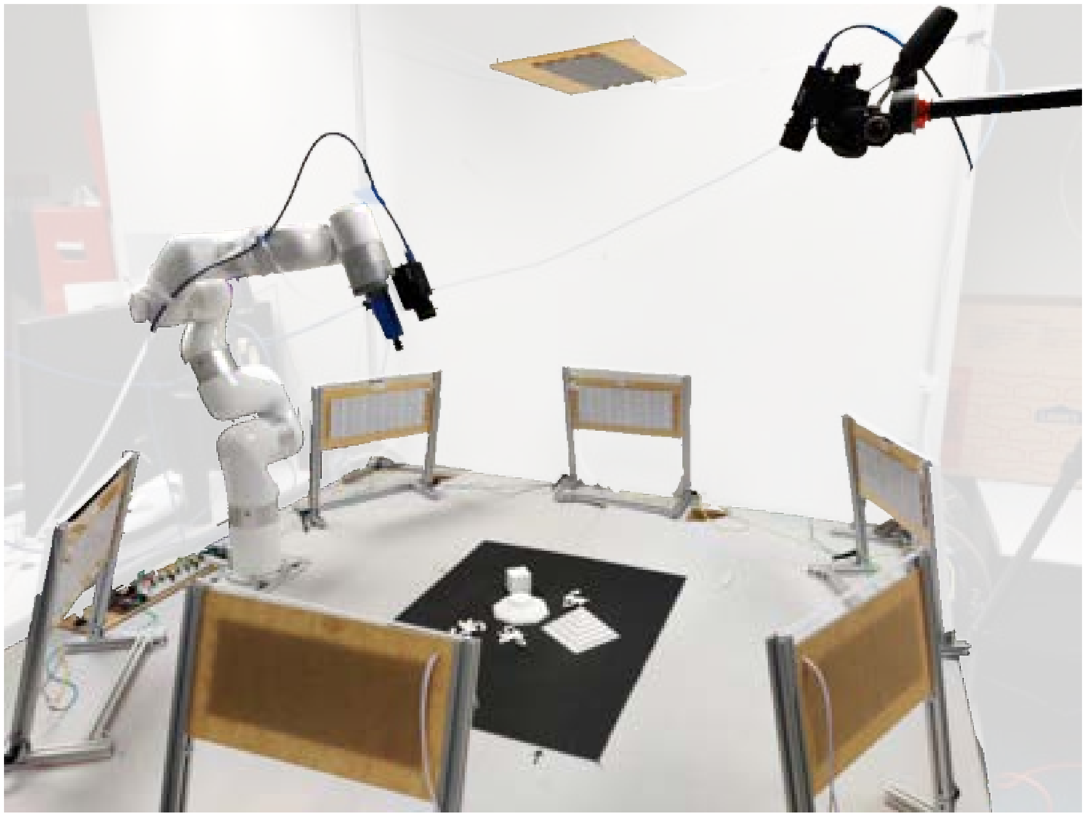
Imagine the following task: you are trying to find a tiny screw on the ground, but the color contrast between the screw and the ground is poor, or the floor has a complex texture. You might turn to a flashlight to aid you in this task, illuminating part of the floor with a low angle of incidence, looking for shadow cues as you hunt for the screw.

Now consider the task of examining the quality of examining the quality of stucco[63] or putty applied to a surface; here, a low angle of incidence light (as in the first task) can highlight high spatial frequency surface discontinuities such as surface roughness and features like creases. A directional source of light (e.g. sunlight throughout the day), meanwhile, can highlight low spatial frequency information such as the curvature of the wall.

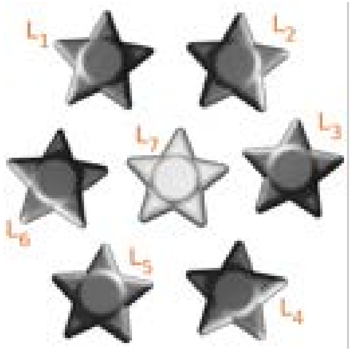
This work is motivated by the observation that changing the direction of illumination can often highlight object surface features and depth discontinuities on the object surface and between the object and its surroundings. These surface features often lead to shadows and illuminated patches when viewed with a camera. Shading and shadows along with an illumination model can aid in reasoning about the surface properties of an object and help us decide how to interact with it.

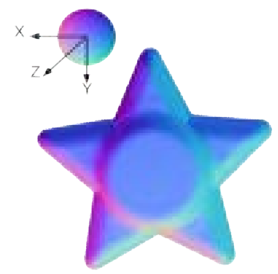
Although classical techniques in multi-view geometry [64], shape from texture [65, 66], high performance area scanners [67, 68] and tactile sensing [20, 38] exist for solving similar problems, multi-modal sensing adds complexity to the problem by requiring the camera poses and object features to be tracked across views and sensors. Coordinating tactile sensing with vision [5] adds the additional complexities of discovering correspondences between cameras and tactile sensor data and accounting for tactile sensor drift. Our experiments focus on precise and effective sensing in a robot manipulation setup [69, 70].

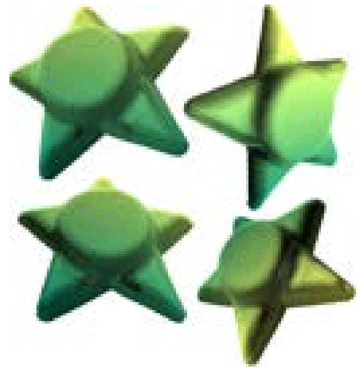
To that end, this paper proposes using actively controlled illumination and tries to address a simplified version of the motivating question: How can controlled illumination be used to estimate



(a)







(b) **(c)** **(d)**

Figure 3.1: We demonstrate the value of controlled illumination approaches for robot manipulation workspaces. Our setup (fig. 3.1a) consists of 2 machine vision cameras (C_1 mounted to the robot and C_2) overlooking a robot's workspace. The illumination of the workspace is controlled using 7 directional lights – six low angle of incidence light sources $L_{1:6}$ placed on the table and one overhead light source L_7. We capture seven images of the object (fig. 3.1b), use standard photometric stereo to calculate object surface normals (color coded in fig. 3.1c), and optionally derive a 3D representation of the object's surface (fig. 3.1d) from calculated normals (fig. 3.1c).

surface properties of objects to make robotic manipulation easier? The simplifications include painting objects with matte paint to remove highlights due to specularities, and using only one color of paint (white) to make the reflectance function uniform over the object.

Active control of sensing parameters is a classical topic in robotics research – [8, 9, 11] define "active perception" as controlling environment parameters (e.g. illumination), extrinsic (e.g. sensor location) and intrinsic (e.g sensor gain, focal length) sensor parameters for better perception. Our work can be classified as a subset of active perception research where the scene illumination, camera poses, and some of the camera intrinsic parameters are actively controlled to solve the perception task at hand.

Although we will handle more general reflectance functions in future work, for this work, we assume that all objects have monochromatic Lambertian surface reflectances, i.e. they do not have specularities and changes in reflectances due to colored patches on the object. We enforce this assumption by coating all objects with matte white paint[1] when necessary. All of our objects are single rigid bodies with no articulations and when needed, we assume that we have 3D models available to us. We show that:

- Controlling illumination of a robot's workspace yields high quality information about surface normals and discontinuities – significantly better than commercially available 3D sensors for table top scale objects ($\leq$20cm^3).
- The computed normals and surface discontinuity information can aid in robot grasping tasks.
- A controlled illumination approach can help us estimate deformation of known objects, and estimate poses of known and unknown objects.

Additional details and results from our work, video demonstrations of robot tasks and summaries of this research can be found on the project web page.

3.2 Related Work

Shape from shading, introduced by Horn[71] and Woodham [72], is a classical problem in computer vision which involves obtaining the shape and surface reflectance of an object by varying either the viewing direction, or the scene illumination or both. Several versions of this problem have been proposed where the researchers have recovered the shape, reflectance and shading of the object from learned priors [73], proposed accurate methods for recovering object shapes as a collection of algebraic surfaces [74], and more recently have demonstrated highly accurate frameworks for recovering object shape, reflectance and geometry by refining multi-view RGB-D data captured by commercial sensors [75].

Although a large portion of recent work [76, 77, 78, 79] advocates learning implicit representations of objects from multiple views and then using them for robotics tasks [80, 81], there has been increasing interest in research on inferring object geometry from its interactions with controlled directional illumination. Recent work has demonstrated the capture of object surface normals and reflectances using multi-view photometric stereo [82], explicitly recovering object shape and reflectance from images taken with cameras equipped with flashes [75, 83] and have demonstrated cameras paired with an ensemble of projectors and flashes to capture scene properties [84, 85, 86]. Other notable efforts include learning implicit scene representations from multiple directionally illuminated images [87], estimating object geometry from shadows cast by the object under directional illumination [88], and reconstructing surface depth and normals from images under directional illumination [89]. Although a large portion of these works demonstrate impressive accuracy for a selected set of objects, it is unclear how appropriate any of these are as a perception system for manipulation tasks. [48] were the first to demonstrate the use of photometric stereo as a metrically accurate sensor and smaller versions of the setup [90, 91] have been shown to be useful for several manipulation tasks. Our

[1]we use RustOleum 7790830 Flat White spray paint

work draws inspiration from [48] as we demonstrate the applicability of object geometry capture for different robotic manipulation tasks using techniques from classical computer vision.

3.3 Methods

It is well known from the shape from shading literature [71, 73] that the measured intensity through an imaging device is a function of three major quantities – the shape and reflectance of the object and the illumination of the environment. In this work, we focus on recovering surface normals as a proxy for the object shape. We use controlled lighting in addition to ambient illumination. Further, we simplify the shape from shading problem by exclusively considering Lambertian objects – painting the objects with matte white paint so that the reflectance is known (Lambertian).

As shown in fig. 3.1a, we illuminate the workspace with a low angle of incidence with approximately parallel light rays $\mathbf{l}$, emulating a light source at infinity. Objects are also indirectly illuminated by the ambient light and we model the illumination as a combination of a linear model [92]:

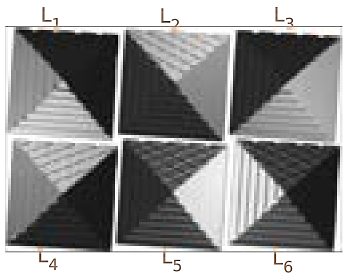
$$I_i^k = \frac{\rho}{\pi}\langle \mathbf{n}_k, \mathbf{l}_i \rangle \quad \forall i \in 1..., 6 \tag{3.1}$$

and a quadratic model $\mathbf{M}$ using second order spherical harmonic functions [48, 93]:

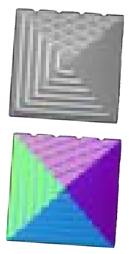
$$I_i^k = \mathbf{n}_k{}^T \mathbf{M}_i \mathbf{n}_k \quad \forall i \in 1..., 6 \tag{3.2}$$

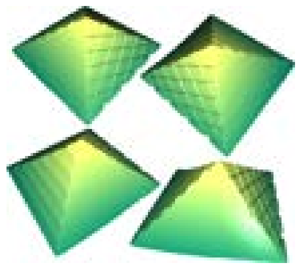
At inference time, given measured pixel intensities I_i^k and albedo ρ, and per-channel linear and quadratic illumination models $\mathbf{l}_i$, $\mathbf{M}_i$, we obtain the surface normals $\mathbf{n}_k$ by inverting the models sequentially for $L_{1:6}$ in fig. 3.1a, while accounting for shadows on the object. Optionally, we spatially integrate the surface normals to get a depth map of the object's surface. Our perception pipeline builds upon traditional techniques from photometric stereo, however, a key contribution of this work is implementing a photometric stereo system as a sensing device for a robot manipulator. In the following three sections, we describe the details of our implementation. However, for a robotics practitioner, the details may not be very relevant and the material can be skipped till without any loss in continuity.

3.3.1 Details of the hardware platform



(a)


(b)


(c)

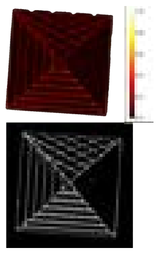

(d)

Figure 3.2: Our pipeline for performing photometric stereo at the scale of a robot's workspace. Figure 3.2a displays the images captured by the camera C_2 using each illumination source – L_1 through L_6 with the background automatically removed. The top image of fig. 3.2b is the image captured using L_7 and the bottom image represents the normals obtained after solving eq. (3.6). The normal slopes about world's X,Y and Z axes (see fig. 3.1a for reference) have been mapped to red, green and blue channels respectively. Figure 3.2c presents the heightmaps of the object constructed by integrating the normals in fig. 3.2b, and, fig. 3.2d from top to bottom presents our setup's confidence about a pixel being a depth edge and the recovered edge map.

Our controlled illumination experimental setup (fig. 3.1a)[2] consists of seven 25cm×10cm light panels fabricated from commercially available LED strips[94] fixed in the robot workspace as shown in fig. 3.1a. Six light panels are placed around the robot's base to illuminate the robot's workspace with low angles of incidence. One panel is placed over the workspace to provide baseline illumination to calculate shadows (more details in section 3.3.3). Each of the panels has a rated power of 45W, and is powered by a 500W switching mode power supply. The light panels are driven by a high-current switching transistor controlled with an Arduino Uno$^{\text{TM}}$. For this work, the illumination of the room due to the ceiling lights, interreflections of all the lights around the ceiling and walls etc. are assumed to be constant. The Arduino relays the control commands from our algorithm running on a workstation to the light panels through a serial port. To capture images, we use two FLIR machine vision cameras (C_1 and C_2 in fig. 3.1a) [95]. The camera C_1 is fitted with a 12mm focal length lens and camera C_2 is fitted with a 16mm focal length lens[96]. The cameras are configured to respond linearly to the amount of light captured by the lenses (gamma = 1) and output 1536×1536 pixels 16 bit monochrome images. To position the cameras and perform manipulation tasks we use an xArm7 manipulator[97] and our vacuum gripper is manufactured from standard 1/4″ vacuum fittings. Robot demonstrations are recorded with two HD webcams placed around the workspace. The data captured is processed on a Linux workstation with an Intel Corei9 processor, 64GB RAM, and an Nvidia RTX3090Ti graphics card with 25GB of vRAM.

3.3.2 Modelling the illumination of the workspace

We choose to illuminate the robot workspace with low angle of incidence lighting, with approximately parallel light rays, emulating a light source at infinity. This arrangement is also known as grazing illumination in the literature (see e.g. [48]). To achieve grazing illumination in practice, we use rectangular shaped light panels larger than our objects and mount the robot so the center of the robot's workspace is approximately equidistant from all our light sources. In this section, we describe our approach for mathematically modelling the illumination in the robot's coordinate frame. To recover our illumination model, we capture images from the two cameras (C_1 and C_2) with only one of each light (($L_{1:6}$ in fig. 3.1a) on. The calibration object is a matte white hemishperical target with a 4cm radius.

Following [48], we choose to model the illumination seen by a camera using a linear and a quadratic model. The linear illumination model, following [92], is the Lambertian reflectance equation

$$I_i^k = \frac{\rho}{\pi}\langle \mathbf{n}_k, \mathbf{l}_i \rangle \quad \forall i \in 1..., 6 \tag{3.3}$$

where I_i^k is the intensity of the k^{th} pixel, given the i^{th} light of $L_{1,...,6}$ is switched on, $\mathbf{n}_k$ is the normal vector at the world point corresponding to the k^{th} pixel in the manipulator's coordinate frame (see fig. 3.1a for reference), $\mathbf{l}_i$ is the direction of the illumination of the i^{th} light source and ρ is the albedo of the surface of our target. By design, no three illumination vectors of our approach are co planar – they are better approximated by vectors on the surface of a frustum, so for each pixel we follow [92], and use the Moore-Penrose operator to invert eq. (3.3) to get the illumination vectors $\mathbf{l}_i, i \in (1, ..., 6)$ at the k^{th} image pixel. As the albedo is constant on the reflective surface, we include it inside the recovered illumination vectors.

The linear shading model, however, does not capture the effects of ambient illumination (room lights), the effects of lights bouncing off of walls to re-illuminate the scene (e.g. light from illumination channels $L_1,\ L_2,\ L_3$ bouncing off the wall (see fig. 3.1a) and re-illuminating the scene), and the errors due to our assumption that we have directional light sources at infinity. To approximately address these artifacts we consider a quadratic shading model derived from a truncated spherical harmonic shading model, which has been shown to be a good approximation of Lambertian reflectance under arbitrary illumination conditions. Researchers have used this model for general purpose rendering (see e.g. [93, 98]), for small scale micro-geometry capture systems ([48]) and for recovering shape

[2]figure numbers have been carried over from the main document

and illumination from shading of images (e.g. [73]). We follow the formulation in [93] and can write the quadratic shading model as

$$I_i^k = \tilde{\mathbf{n}}_k^T \mathbf{M}_i \tilde{\mathbf{n}}_k \quad \forall i \in 1...,6 \tag{3.4}$$

where, $\tilde{\mathbf{n}}_k = [\mathbf{n}_k, 1]^T$ and $\mathbf{M}_i$ is a 4×4 symmetric matrix with 9 independent quantities (spherical harmonic lighting coefficients) per illumination source. For each light source, we expand eq. (3.4) and solve a system of linear equations to obtain the lighting coefficients.

Finally, we observe that due to the limited power of the lights and further errors in our assumptions on modelling the lights and the sensitivity of the cameras, the illumination coefficients vary at different parts of the workspace and to take this into account, we take several images of our calibration target (20-25 placements of the target in a workspace of 460mm $\times$ 610mm) and use a bi-quadratic spline interpolation to model the spatial variation of the linear and quadratic illumination coefficients in the robot's workspace. Higher order illumination models were unnecessary given that our objects were Lambertian.

3.3.3 Recovering surface normals from images

With the linear and quadratic illumination coefficients recovered, and the object reflectances made uniform and known due to the white paint, we expect to be able to obtain the *shape* (world coordinate normals at image pixels in a camera) of an object as seen by the cameras. Although, this should be as simple as evaluating eq. (3.3) at every pixel to get a initial estimate of the shape and then refining it using eq. (3.4), cast shadows and unequal influences of the light sources at every pixel due to the relative location of the object and the lights slightly complicate the shape recovery. In this section we describe our approach for recovering the normals at each of the imaged pixels by reasoning about their illumination or shading. We note that it is known from literature (see e.g. [99, 100]) that recovering shadow contours yields better performance in shape from shading problems but we choose to reason locally at every pixel to recover shape. With our choice of large image sizes, narrow field of view lenses, and freedom of locating the camera in the workspace, we can typically get a resolution upwards of 50 pixels/mm^2 and can achieve reasonable shape estimates through local reasoning.

To get an initial estimate of whether a pixel is illuminated or shadowed, we compare the intensity of the pixel for the images captured using $L_{1:6}$ with the image captured with L_7 (see fig. 3.1b). We generate an initial binary mask for shadowed and illuminated pixels by observing that the pixels illuminated due to a directional light source ($L_{1:6}$) would *almost always* be brighter than the pixels illuminated with an overhead source (L_7). With this, we get a binary shadow-illumination vector $\mathbf{w}_k \in \{0,1\}^6$ for all the directional sources ($L_{1:6}$) and augment it to get a binary diagonal shadow-illumination matrix $\mathbf{W}_k = \mathrm{diag}(\mathbf{w}_\mathrm{k})$. Using $\mathbf{W}$ and following [92] we can invert a weighted version of eq. (3.3) per-pixel to get the initial estimates of *shape* at each pixel as:

$$\hat{\mathbf{n}}_k = (\mathbf{W}_k \mathbf{L}_{lin}^k)^\dagger \mathbf{W}_k \mathbf{i}_k \tag{3.5}$$

where $\mathbf{L}_{lin}^k \in \mathbb{R}^{6\times3}$ is the concatenated illumination matrix at the world location of the k$^{\text{th}}$ pixel for each of the illumination sources, obtained by concatenating $\mathbf{l}_i \forall i \in 1,...,6$, and $\mathbf{i}_k$ is the vector of all the intensities observed at the k$^{\text{th}}$ pixel due to $L_{1:6}$. $(\cdot)^\dagger$ is the Moore-Penrose inverse operator.

However, in practice, the effect of the shadows are not binary and if a pixel is shadowed across more than 3 light channels, which happens often for undercut surface features, eq. (3.5) is not solvable. We frequently encountered this scenario for objects like the textured pyramid, the Buddha, the bunny, and while imaging dough balls (high resolution images on paper website). In these cases, the estimates from eq. (3.5) were incorrect and we assigned shadowed pixels the normals of their nearest valid neighboring pixel. This introduces high local errors in the normal estimates and to address this, along with our motivation for recovering the quadratic shading model leads us to jointly refine the normal estimates $\hat{\mathbf{n}}_k$ and shadow contributions $\mathbf{w}_k$ from eq. (3.5). To do this, we use the previously estimated $\hat{\mathbf{n}}_k$ in eq. (3.4) to estimate the intensities $\hat{I}_i^k, i \in (1,...,6)$ of each pixel conditioned on which light source ($L_{1:6}$) was switched on. We then weigh the intensities with their corresponding shadow weights $\mathbf{w}_k$ and compare the predicted intensities to the observed intensities $I_i^k,\ i \in 1,...,6, \forall k$ to get a per pixel loss $\ell_k = ||I_i^k - \hat{I}_i||_2$. Finally, we iteratively solve a regularized photometric loss (eq. (3.6)) across all pixels and channels while updating our hypotheses of per-pixel shadow weights using eq. (3.7) at every step.

$$\sum_{k,i}(||I_i^k - w_k^i \hat{I}_i^k||_2) + \beta\mathcal{L}_d(\mathbf{n}_k)\ i \in (1,...,6), \forall k \tag{3.6}$$

$$\mathbf{w}_k^{t+1} = \left\{ \begin{array}{c} \mathbf{w}_k^t,\ \text{if}\ \ell_\mathrm{k}^{\mathrm{t+1}} \leq \ell_\mathrm{k}^\mathrm{t} \\ \epsilon \max\{\epsilon, \ell_k^{t+1}\}\ \ \text{if}\ \ell_\mathrm{k}^{\mathrm{t+1}} > \ell_\mathrm{k}^\mathrm{t} \end{array} \right\} \tag{3.7}$$

As we treat every pixel individually, we incorporate the intuition that neighboring pixels (or world points) should have similar normals (also known as an integrability constraint in [48, 71] or smoothness priors in [73]) through a Laplacian cost $\beta\mathcal{L}_d(\cdot)$ in eq. (3.6). The influence of this regularizer can be controlled through the width of the Laplacian filter d and the weight β. We minimize eq. (3.6) using gradient descent using backtracking line search ([101]) – variants of stochastic gradient descent were too unstable for our use case. We also note that $\mathbf{n}_k$ is always calculated in the robot's coordinate frame, as was the case with the illumination model and the calculated normals are independent of the camera's orientation in the manipulator's coordinate frame. Our two step refinement is somewhat similar to the one described in [48], however, our per pixel inference model along with the differentiability incorporated in the computational structure affords large accelerations with modern tensor libraries and GPGPUs. We discuss further implementation details in section 4.9.1.

3.3.4 Recovering object edges and depth

Recovering edges: We observe that occlusion of a light source at any point on an object surface depends on the relative location of the light source with respect to the object, so occlusion edges change when the light source location is changed. Consider the image due to light source L_1 in fig. 3.2a – one can intuitively conclude that the light source is at the bottom left of the object and as we move along the light's path (from bottom left to top right), the sudden change in brightness of the pixels denotes a sharp change of object's visibility along the direction. Indeed, the illumination-shadow ridge is along the image of the ridge where two sloped faces of the tetrahedron meet, and the resulting surface patch falls out of the "view" of L_1. Similar reasoning applies to all the images captured using the illumination channels $L_{2:6}$ in fig. 3.2a. This observation was used by [102] to perform non-photorealistic stylized rendering of images and by [52] for detecting edges to localize fabricated parts. In this work, we modified Raskar and colleagues'([102]) pipeline to accommodate illumination sources not co-incident with the camera and use the relative orientation between the camera and illumination sources to calculate edges in an object due to depth changes. Figure 3.2d (top to bottom) shows our confidence (following [102]) about whether a pixel is at a depth edge and the edge map obtained by hysteresis thresholding (see e.g. Canny [103]) the confidence map. We provide more details in section 3.3.5.

Recovering depth: With the surface normals from the camera's viewpoint recovered, we can spatially integrate the normals to get a representation of the surface in 3D, as imaged by the camera. This is a classically studied problem in vision known as "shape from shading" and there are several solutions to this problem in the literature. We looked at four classical solutions given along the first column of table 3.1. We benchmarked them in four aspects: computation speed (speed), robustness to local errors in calculated normals due to shadows, accuracy of surface reconstruction (or absence of strong global smoothing priors) and admissibility of non-axis-aligned arbitrary quadrilaterals image patches. Table 3.1 summarizes our qualitative findings and for the results in this work, we used the perspective corrected Poisson integration ([104]). Resulting integrated depth maps can be seen in figs. 3.1d and 3.2c, obtained by integrating the normals in figs. 3.1c and 3.2b respectively.

However, we should note that, unlike other true depth sensing systems (see table 3.2) which

Method	Speed	Robust	Accurate	Shape
Variational calc. ([105, 106])	✓	×	×	✓
Fourier ([107])	✓	✓	×	×
Least sq. (IRLS) ([48, 108])	×	✓	✓	✓
Poisson Int. ([104])	✓	✓	✓	✓

Table 3.1: Qualitative comparison between different normal integration methods. For the results presented in this work, we used the perspective corrected Poisson integration technique. The techniques which had suitable behavior in our test criteria have been marked with a ✓ sign, and unsuitable behavior has been marked with a × sign.

directly measure depth at every pixel of the imaged scene using stereo or time-of-flight sensing, the integrated heightmaps are implicit surfaces that locally have the same normals as the calculated normal map. They will be metrically incorrect unless the whole object is visible from the camera's view port, and the projection scale and the boundary conditions for the integration are not exactly known. We ensured this for the objects presented in figs. 3.1d and 3.2c. An object like a cuboid would not work. This limits our approach's applicability as a true depth sensor for manipulation tasks when only one camera view is being used. Counter-intuitively, this is not a limitation while obtaining depth maps from tactile sensor images (see e.g. Chaudhury et al.[5] and Johnson et al.[48]) where the aim is to only reconstruct the deformed sensor surface - not the object beyond the deformed gel membrane. However, apart from the small sensing footprint, elastomeric tactile sensors fail to capture sharp surface details as they drape over the surface discontinuities.

3.3.5 Details of our steps for calculating occlusion edges

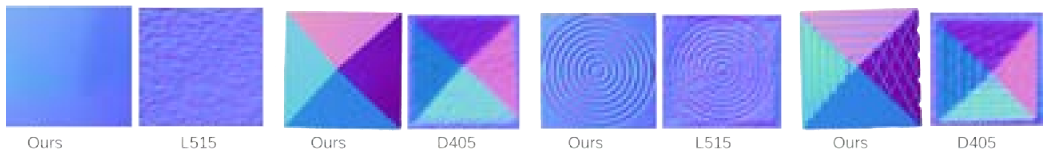
For detecting occlusion edges we adapt the algorithm described in [102] to suite our approach with lights far away from the camera. We first collect the directionally illuminated edges $\mathbf{I}_{1:6}$ corresponding to L_1 through L_6 and the image $\mathbf{I}_7$ with the overhead light L_7 (see fig. 3.1a). We use the intuition that, given the viewpoint remains fixed, a portion of the scene illuminated with a directional source ($\mathbf{I}_{1:6}$ due to $L_{1:6}$) would be highlighted in contrast to an image due to the overhead light ($\mathbf{I}_7$ due to L_7). Following [102] we calculate the ratio images in eq. (4.11), making adjustments to avoid numerical errors.

$$\mathbf{R}_{1:6} = \frac{\mathbf{I}_{1:6}}{\mathbf{I}_7} \tag{3.8}$$

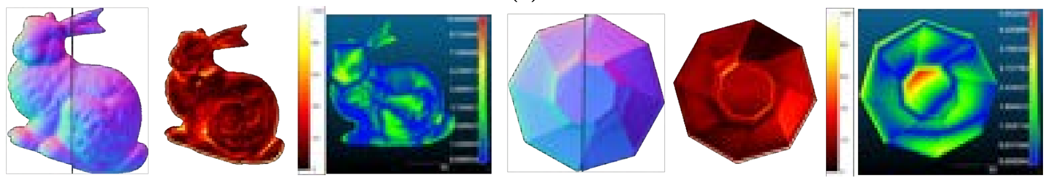
The ratio values in $\mathbf{R}_{1:6}$ will be higher for the areas illuminated with the directional sources ($L_{1:6}$) and lower for areas in shadows, with a distinct transition (from low to high values) along the occlusion edges. To further highlight the individual contributions of the directional illumination, we jointly reason about the transitions in $\mathbf{R}_{1:6}$ conditioned on the direction of the illuminating source $L_{1:6}$ with respect to the camera. For each light source $L_{1:6}$ making an angle $\theta_{1:6}$ with the image axis (camera axes X and Y correspond to the vertical and horizontal image axes respectively), we extract the pixel values in $\mathbf{R}_{1:6}$ with strong transitions along a direction of $\theta_{1:6}$. This gives us a measure of our confidence that a pixel is on an occlusion edge (see top image of fig. 3.2d). Finally following [103], we apply hysteresis thresholding to the confidence map and obtain a binary map of occlusion edges (bottom image of fig. 3.2d).

3.4 Experiments and Results

In this section we describe our experiments on using our sensing system for common manipulation tasks. We start by quantifying the performance of our approach and then demonstrate the use of our approach in three sub-tasks related to manipulation – general purpose object picking, estimating object deformation with vision, and pose estimation. We focus the paper on the results of our experiments, deferring details of the methods to the supplementary material.



(a)



(b)

Figure 3.3: Our measurements compared to commercial depth sensors. In fig. 3.3a we compare the quality of normals computed by our sensor with the processed data from the best performing commercial depth sensor (table 3.3). Figure 3.3b shows the normals of the bunny and octagon II from table 3.3 as sets of 3 images each. In each set from left to right, we overlay the normals calculated by our approach (right of black line) on the ground truth normals (left of black line), the per pixel deviations of the normals from ground truth (in degrees) and the deviations of the measured surface from the ground truth mesh in mm.

3.4.1 Performance of our sensing approach

Sensor	Flat 0°	Flat 45°	Texture 0°	Texture 45°
D455	0.12	0.12	0.27	0.12
D405	0.09	0.04	0.21	0.15
D435	0.09	0.23	0.22	0.22
L515	0.06	0.05	0.20	0.15
Kinect II	0.24	0.14	0.19	0.16
Ours	**0.02**	**0.01**	**0.18**	**0.10**

Table 3.2: Comparison of our approach with commercial depth sensors (lower is better). The metric value indicates the *dissimilarity of the measured normals with ground truth* which in the case of flat surfaces is related to the standard deviation of the angles of the normals with respect to the mean. Representative normal maps corresponding to the next best performing sensor have been provided in fig. 3.3a along with the data captured by our sensor in those categories.

Quality of normals versus true depth sensors

The commercial depth sensors widely used for robot manipulation tasks are fundamentally different from our sensor because the primary measurement in those sensors is the depth of the visible surface from the sensor calculated either through stereo matching (first 3 rows of table 3.2), or through time-of-flight (rows 4, 5 of table 3.2). For these depth sensors, we calculate the normals as spatial derivatives of the captured depth. Our sensing approach infers object surface normals from shaded images and optionally calculates a depth map that best explains the observed normals when the whole object is visible from the cameras' viewpoint. We also note that the commercial depth sensors we used (except for the D405) are designed to operate in room scale environments and are not necessarily suited for measuring surfaces of the objects we used. To focus on the quality of the estimated surface normals we measure the statistical similarity of the normals measured across all the sensors on the same object patches. We image flat and textured surfaces at two orientations – facing the camera's projection axis and at an inclination of 45° to the projection axis at the closest possible distance. We use the earth movers distance [109, 110] to compare the normals captured by our approach and the commercial sensors to the ground truth.

	Normals ($\triangle°$)		Surface ($\triangle$mm)	
Object name	$\mu(\sigma)$	$< 20°$ (%)	$\mu(\sigma)$	max
Pyramid	13.31 (12.89)	93.06	1.31 (0.93)	4.17
Star	18.85 (14.04)	86.73	0.80 (0.60)	3.65
Bent Cyl.	18.81 (15.08)	86.19	0.50 (0.36)	2.03
Octagon I	16.23 (12.53)	83.45	0.87 (0.85)	4.50
Octagon II	17.17 (12.09)	82.52	1.20 (0.90)	5.05
Spot	17.38 (10.42)	83.55	0.54 (0.48)	2.82
Bas-relief	18.24 (10.01)	76.24	0.87 (0.94)	4.26
Bunny	20.07 (14.80)	75.88	1.60 (1.27)	8.12
Text. Pyramid	19.19 (17.52)	61.75	1.74 (1.31)	7.30
Happy Budd.	29.19 (20.52)	54.29	1.54 (1.22)	7.66

Table 3.3: Quantitative measurements of our approach in estimating object shape. The measurements in the first set of columns are the deviations of measured normals from ground truth normals in degrees. The measurements in the second set of columns are the deviation of the reconstructed surface from the ground truth mesh in millimeters. Please visit the project website for a qualitative visualization of the results.

We also note that the stereo-based depth sensors rely on visual textures which our objects lack. To address this, we follow the recommendations for imaging textureless surfaces from [111] by projecting visible (D405) or infrared (D435, D455) patterns on the objects as applicable. Additionally, to reduce the noise in the measurements, for the stereo sensors (rows 1 through 3 in table 3.2), we calculate the measured depth as a trimmed mean (we remove 10% of smallest and largest outliers) of all the depths at each pixel for 50 consecutive frames (acquired over 1.5 - 2 seconds). For the time of flight sensors (rows 4 and 5 in table 3.2) we use a 3×3 pixel window median filter to smooth the captured depths at each pixel after temporally filtering the data using trimmed means. We present our quantitative results in table 3.2 and our qualitative results in fig. 3.3a. We outperform all the commercial sensors in imaging surfaces as normals in each category, often by significant margins. Our approach is much better at detecting that a surface is actually smooth and flat, and in measuring local surface orientation. Unsurprisingly, we also note that the D405 and L515 sensors outperform the other sensors because they were designed for (or support) close range imaging which is relevant to the task we tested the sensors on.

Accuracy of our approach

To quantify the accuracy of our sensor, we 3D printed a set of objects of footprint less than 12cm^2 using a standard 3D printer. The objects were imaged with our setup in fig. 4.1. The surface normal quality measurement is performed by finding the object pose that aligns the measured surface normals to the ground truth surface normals generated with a renderer imaging the ground truth mesh. We used photometric stereo to calculate the normals and the method described in section 3.4.4 to align the captured object to the ground truth mesh model. We then calculated the angle between the measured normal and the rendered ground truth normals at every pixel and reported the statistics of the angles as a quantitative measure of the quality of the normals estimated by our approach. Our pose estimation pipeline (section 3.4.4) aligned the simulated and the real data up to a maximum error of 5 pixels, making the per pixel error calculation meaningful. In table 3.3 under the label 'normal quality' we report the mean and the standard deviation of the per pixel angle error in degrees. We also report the percentage of pixels with angle error less than 20°. This metric was influenced by our observation during vacuum picking experiments (see section 3.4.2) that with the choice of a more compliant gripper, our gripping pipeline was tolerant of surface normal errors up to 20°. Notwithstanding the errors introduced due to warps on some 3D printed models (see e.g. high error pixels around the edges of insets 2 and 5 in fig. 3.3b), the quality of our normal and depth estimates are similar to classical methods [73, 74] and are marginally worse than recent deep learning based methods [82, 87, 89].

To measure the surface quality, we generated a surface depth map by spatially integrating the normals of the scene calculated using photometric stereo (more details in the supplementary materials). We then registered the integrated depth map to the ground truth mesh using point-to-plane ICP [112] and calculated the Hausdorff distance [113] between the recovered depth map and the ground truth mesh of the object. The mean, standard deviation and the maximum point-wise distance of the recovered depth map from the ground truth meshes are reported under the surface quality column of table 3.3. For both the experiments, we observe that the objects with large planar faces or smoothly varying curvatures worked the best while, objects with undercut surfaces performed poorly – especially the happy Buddha object which had several undercut faces (see fig. 3.6a). We present a qualitative result in fig. 3.3b. More qualitative results can be viewed on our project website.

3.4.2 Controlled illumination for pickup tasks

To perform immobilizing grasps with a single suction cup vacuum gripper, we need to identify a portion of the object that is large enough for the suction cup to fit and flat enough for the suction cup to be effective. Additionally, we have to localize the point of grasp with respect to the gripper and identify the local surface orientation so that the gripper can approach along the surface normal with the face of the suction cup perpendicular to the surface to best execute the grasp. We have described pipelines to identify surface normals, demonstrated our system's performance in measuring surface discontinuities, and we have a stereo system (C_1 and C_2 in fig. 4.1) for triangulating a world point in the robot's frame. In this section, we describe our pipeline to detect arbitrarily oriented flat objects that can be grasped with a suction cup gripper of a given size. We also demonstrate how low angle of incidence directional illumination can help identify and pick up thin flat objects when segmentation is difficult using color or depth contrasts with conventional sensors.

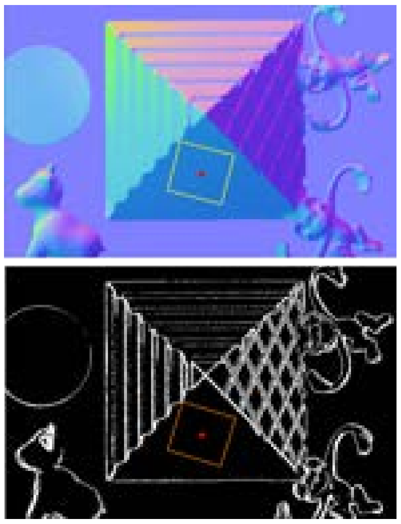
We divide the problem of picking up 3D objects into two major steps – identifying the largest geometrically flat patch in the workspace across two views and executing the robot motion to grasp the identified face. To identify the largest corresponding flat patch across the camera views, we modified the well-known CAMShift algorithm[114] to filter out image patches not graspable with a vacuum gripper.

With geometrically corresponding grasping locations identified across two views, we use camera poses and intrinsics of the two views C_1 and C_2 to triangulate the position of the gripper in the robot's coordinate frame. For the orientation of the gripper – we calculate the normal at the grasp location by averaging the normals at the identified grasp location in the two views. We then generate and execute a minimum jerk trajectory [115] which moves the robot to make contact with the selected object, picks the object up and, places it in a bin in the robot's workspace.

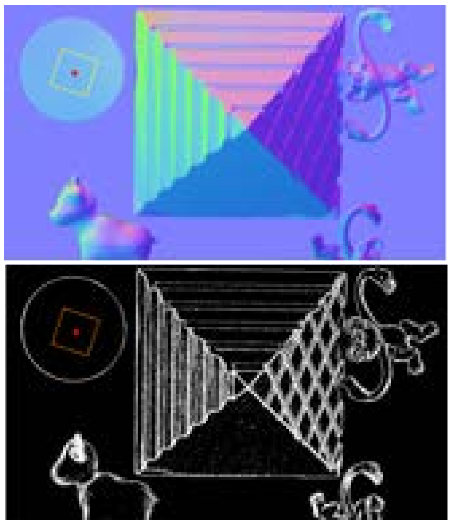
For picking up thin objects without color contrast following Raskar et al. [102], we observe that image brightness variations due to occlusion are more prominent than brightness differences due to color texture. We use this intuition to identify thin and flat objects from the background. For this experiment we pick a plastic disc of 50mm diameter, and 3.15mm thickness with a random pattern (pixel values sampled uniformly between 0 to 255) printed on the surface of the disc from a flat plane with the same visual texture. As is obvious from figs. 3.4c and 3.4d, there isn't enough intensity or color texture to segment the object from the background. However, looking along the light direction (see fig. 3.4c), we note that the shadows due to occlusion are more prominent than the visual textures on the object and the background – as imaged in figs. 3.4c and 3.4d.

This shadow cue was sufficient for our edge detection procedure (see supplementary materials for details) to identify occlusion edge pixels in the scene. We then estimated the center of the disc by fitting a minimum enclosing circle [49] to the occlusion edge pixels across the two views C_1 and C_2. We visualize this in fig. 3.4d – the centers of the circles have been projected onto the images captured by the two views with the ambient and overhead light (L_7 in fig. 3.1a). To pick up the disc, we approached the center of the disc along the surface normal of the tabletop by triangulating the object from the two views in fig. 3.4d. None of the commercial depth sensors we used (see table 3.2) detected the disc reliably.

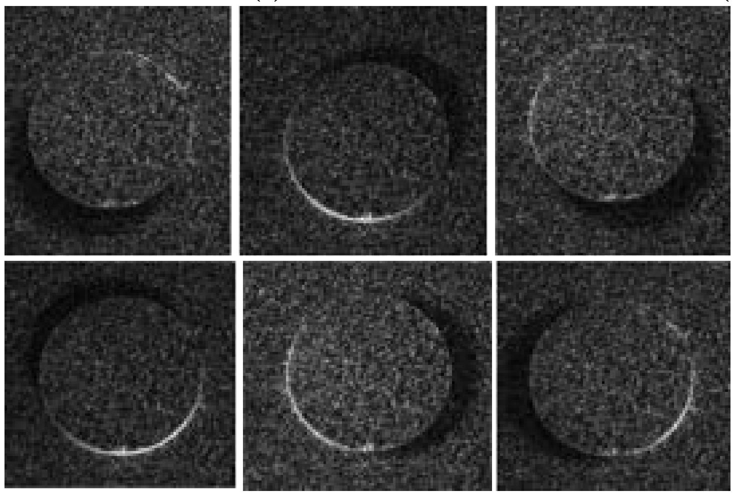
We performed 35 grasp experiments with 3D objects and ten experiments with flat textured



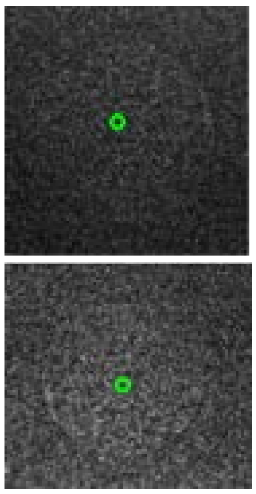
(a)



(b)

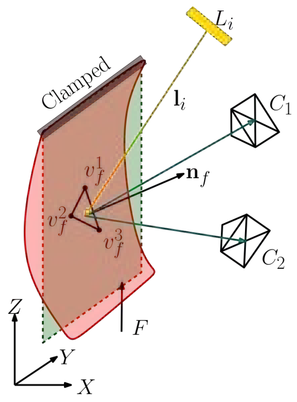


(c)

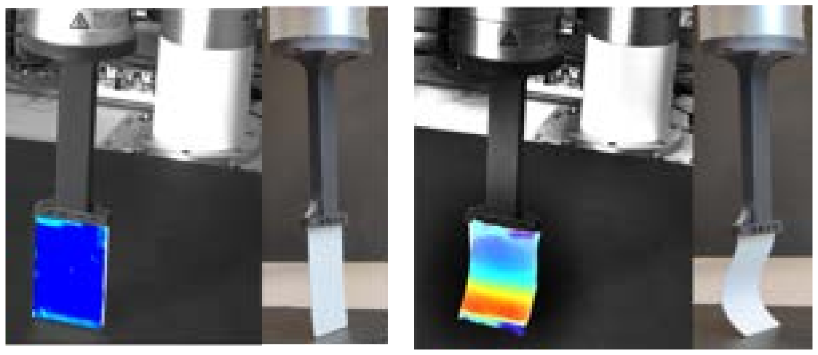


(d)

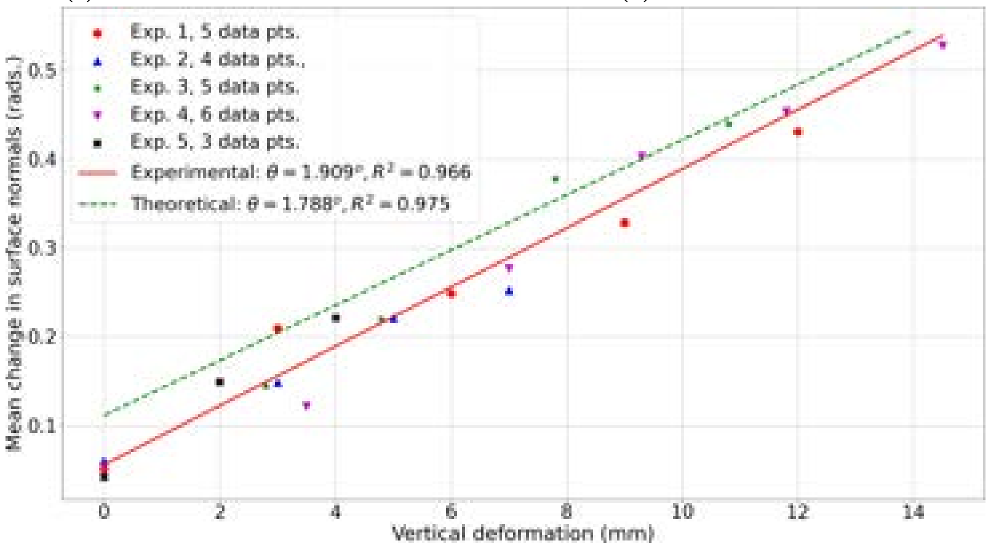
Figure 3.4: Our pipeline for grasping objects in a robot's workspace. Figures 3.4a and 3.4b respectively show two distinct grasps selected by the camera C_1 along X and Y directions respectively. On the top rows of figs. 3.4a and 3.4b we show the detected grasps projected on the scene normals and, on the bottom row, we show the detected grasps projected onto the occlusion edge pixels of the scene. We note that our selected grasps do not have occlusion edges. Figures 3.4c and 3.4d shows our pipeline for picking up thin objects in absence of obvious depth or texture segmentation cues. Figure 3.4c shows the images of a plastic disc (50mm diam., 3.15mm thick) captured with C_1 placed vertically above the disc, using the directional lights $L_{1:6}$. Figure 3.4d shows our detection of the center of the discs overlaid on the image captured with cameras C_1 (top) and C_2 (bottom) with L_7 and ambient light illuminating the scene.



(a)



(b)



(c)

Figure 3.5: Our results of measuring the deformation of known objects. Figure 3.5a shows a schematic diagram of our procedure, fig. 3.5b shows the estimated shape of the card overlaid on a camera image. The insets of the images in fig. 3.5b provide an external view of the state of the cards. Figure 3.5c summarizes the quantitative results of result of our experiment to show we were able to estimate the physical process behind the deformation of the card.

objects (5 with the disc, and 5 with an irregular octagon inscribed by a 50mm diameter circle and of 3.15mm thickness). Movies of our experiments can be viewed on the project website. Across all the experiments we failed four times while picking up the textured pyramid (fig. 3.4a) due to the gripper failing to attach or colliding with the object due to triangulation errors.

3.4.3 Measuring deformation

Our approach can be used to perceive the deformation of objects by tracking the change of local normals on the object's surface. In this section, we describe our method for measuring the buckling deformation of a 0.76 mm thick rectangular PVC card of size 86×54mm. We measure the deformation in a typical analysis-by-synthesis fashion – we explain the change in the surface normals of the deformed card by calculating the deformation of the card geometry.

Figure 3.5b shows a qualitative result of our pipeline – the first image shows the reconstructed

Method	Scale	Error $\mu(\sigma)$	Max	Sample size
VIRDO [116]	89×56	1.12 (- - -)	–	5.6k
VIRDO++ [117]	89×56	1.04 (0.35)	–	5.6k
Ours	86×54	0.78 (0.52)	4.57	100k

Table 3.4: Comparison of our method of measuring deformation (chamfer distance) against baselines [116, 117]. Our experiment (single view) was carried out in simulation because we do not have access to a sensor accurate enough to measure ground truth deformations, and we are comparing the performance of Wi and colleagues' results during inference on simulated data with a single camera view. Our experiment captures the deformed card at a simulated endpoint deflection of 12 mm corresponding to a mean change in surface normals of 0.41 radians. All length measurements are in mm. and all the experiments assume full visibility of the object.

shape before the onset of buckling and the second image shows the reconstructed mesh overlaid on the deformed object. In both cases, we show the local change in surface normals of the reconstructed shape as the color of the mesh. An external view of the objects is also provided as the inset to the images. During our experiments we observed that skewed viewing directions of the cameras C_1 or C_2, e.g. the inset views in fig. 3.5b, significantly reduced the performance of our pipeline due to the decrease in the effective number of pixels imaging the object across the two views. This led us to generally use C_1 and C_2 to capture frontal views. More qualitative results of our experiments can be viewed on the project website. Detailed explanation of our pipeline can be found in the supplementary materials.

To verify that we indeed captured the physical process behind the card buckling under vertical loads, we compared our predictions with theoretical values calculated using solid mechanics. The theoretical data showed a linear trend in the mean change in surface normals and the vertical deformation of the card. A line with a slope of $1.78°$ and offset of 0.114 fits the theoretical data with a r^2 score of 0.975. We repeated the experiment of deforming the card five times with five different PVC cards and different camera positions (C_1 and C_2 in fig. 3.5a) and obtained 23 data points as shown in fig. 3.5c with five different markers corresponding to the experiments. The experimental data also showed a linear trend – a line with slope of $1.9°$, offset of 0.055 fit the data with a r^2 score of 0.966. Disregarding the difference in the offset due to the baseline noise in our measurements and the slight violation of a geometric boundary condition due to slipping of the card at the base, from fig. 3.5c we can conclude that our pipeline is repeatable and can capture the physical process of the buckling deformation of the card due to vertical loads.

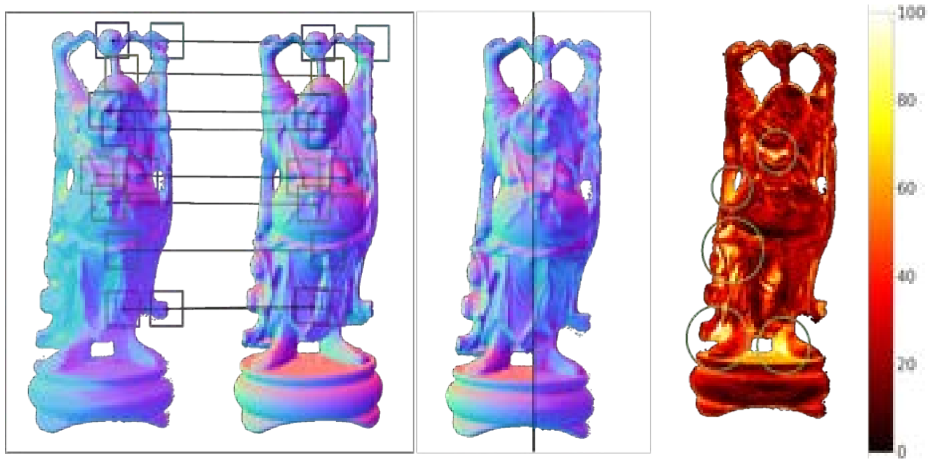
We note that in this work we exclusively use vision. We do not factor in the force that is causing the deformation in the object. In related work by Wi et al. [116, 117] the authors combine visual measurements with forces measured by a force torque sensor coupled to the deforming object to infer its deformation. For objects of similar scale, we compare our performance with the works of Wi and colleagues in table 3.4. Although we perform better in reconstructing shapes, we require nearly full visibility of the object in both views which is not a limitation for Wi et al. in [117].

We also note that our "analysis-by-synthesis" procedure works better than integrating the captured normal map (see supplementary for details) to reconstruct the surface of the card. This is due to the strong view dependence of the shape reconstruction, which prevents us from trivially incorporating multiple views to reconstruct the bent object. Using two camera views and an initial mesh of the object to predict the shape of the deforming card makes our pipeline more robust to shadows and slight occlusions in camera views.

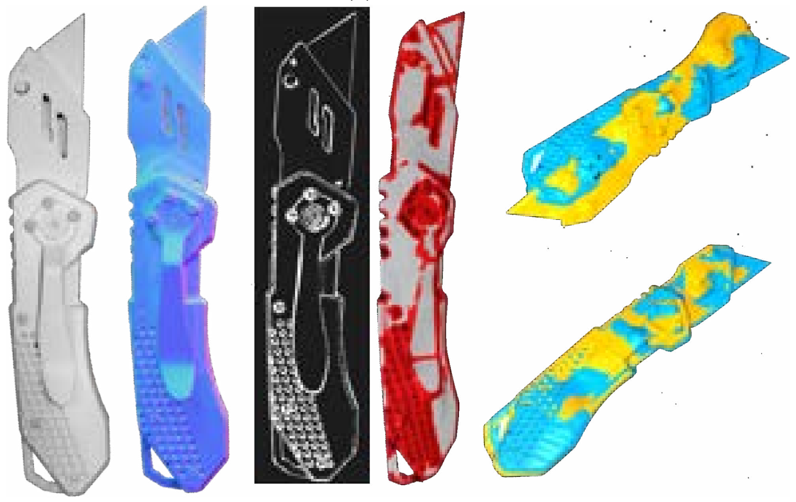
3.4.4 Controlled illumination for localization

Estimating poses of known objects

When an object model is available, we estimate its pose by aligning the measured surface normals, depth edges, and object silhouettes of the object with their simulated counterparts. We improve on



(a)



(b)

Figure 3.6: Estimating object pose in the robot workspace. Figure 3.6a shows our steps in estimating object poses when a 3D model of the object is available apriori. From left to right in fig. 3.6a we calculate the pixel-wise correspondences between captured data and the available geometry, overlay the captured surface normals (right of black line) onto the ground truth object normals (left of black line) and show our pixel-wise pose estimation costs about an arbitrary scale. The green circles indicate areas with high local costs due to errors in estimating normals. Figure 3.6b shows our pipeline for tracking object pose when a 3D model is not available a priori. From left to right, we show our captured data, calculated normals, calculated depth edges, calculated 3D representation of the surface with salient point-features overlaid on the points in red and, initial and final stages of our estimation of the object's pose.

Object	Bounding Box (mm)	Px. Errors $\mu(\sigma)$
Pyramid	$100 \times 100 \times 55$	2.50 (1.70)
Star	$110 \times 110 \times 15$	3.62 (1.03)
Bent Cylinder	$116 \times 48 \times 13$	5.50 (2.12)
Octagon I	$100 \times 100 \times 29$	3.35 (0.88)
Octagon II	$100 \times 100 \times 29$	2.85 (1.83)
Spot	$52 \times 15 \times 53$	1.50 (0.23)
Bas-relief	$83 \times 110 \times 7$	0.80 (0.20)
Bunny	$93 \times 94 \times 43$	4.06 (1.19)
Text Pyramid	$100 \times 100 \times 55$	4.02 (0.96)
Happy Buddha	$115 \times 45 \times 37$	3.10 (0.69)

Table 3.5: Performance of pose estimation for known objects. All the experiments were done with a resolution of ~7 pixels/mm. Qualitative results for the objects happy Buddha, bunny and octagon II can be found in fig. 3.6a and the first and fourth insets of fig. 3.3b respectively. More qualitative results can be found on the project website.

commonly employed pipelines in the pose estimation literature [5, 52, 53] by generating good initial pose estimates through discovery of object patch correspondences between the observed and simulated surface normals (inset 1 of fig. 3.6a). We describe our method in detail in the supplementary material.

We present a qualitative result in fig. 3.6a – the second inset overlays the normal images with measured normals $\mathbf{N}_R$ on the left of the black line and rendered normals $\mathbf{N}_S$ on the right. In the third inset of fig. 3.6a, we visualize our per-pixel pose estimation costs overlaid on the observed image silhouette $\mathbf{M}_S$ on an arbitrary scale between 1 to 100. We note that the under-cut parts of the geometry, highlighted by green circles, have large local costs, but we do not incur large costs due to silhouette misalignment or scaling as seen by the absence of high cost patches at the object edges or background. For all the objects tested, our multi-scale and multi-modal pose estimation pipeline reliably converges to the correct pose within a 5.5 pixel error (7 px/mm) even with high local errors in the measured data – the model in fig. 3.6a has ~50% of pixels with erroneous normals ($> 20°$ deviation from ground truth).

We present quantitative measurements of our pose estimation approach for all the objects in table 3.3 and table 3.5. Our experiments were done with C_1 imaging a 227mm × 129mm area at a resolution of 7 pixels/mm. We placed each of the objects at a random position and orientation and used the pipeline described in this section to align the 3D model to the observed data. After the alignment, we overlaid the simulated and measured data and manually measured the pixel misalignment between the real and simulated data around the object edges. We repeated this experiment five times for each object and report the mean and standard deviations of the pixel misalignment in table 3.5.

Estimating the pose of unknown objects

If a 3D model of an object is not available, the pose estimation problem, defined classically, is ill-posed. In those cases, we can estimate the object's change in pose through rigid registration of the 3D representations of the same object as it moves. For our 3D representation, we choose the point cloud obtained by integrating the surface normal maps, factoring in the camera intrinsics (see supplementary for details). As noted before, this representation is not metrically correct for parts of the object that are not fully visible by the camera (e.g. full profile of the belt clip of the knife in fig. 3.6b), however, the relative surface depth changes and the overall scale of the object are captured accurately which lets us estimate the change in pose between two measurements of the same object. We pictorially describe our pipeline for tracking unknown objects in fig. 3.6b, which involves capturing the image of the object, measuring its surface normals and depth edges, generating a point cloud of the observed surface and registering two instances of the point clouds to obtain the change in pose. We describe our pipeline in detail in the supplementary material.

To evaluate our pipeline for estimating pose changes of unknown objects, we imaged four objects

Object	Bounding Box (mm)	$\triangle X$(mm)	$\triangle Y$(mm)	$\triangle\theta$(°)
Knife	$172 \times 30 \times 20$	1.02	1.66	0.36
Monkey	$46 \times 64 \times 8$	1.67	0.97	0.22
Circuit	$13 \times 21 \times 33$	3.28	2.40	1.28
IO shield	$120 \times 39 \times 21$	1.48	1.86	0.25

Table 3.6: Performance of our tracking pipeline for unknown objects. All the experiments were done with a resolution of approximately 7 pixels/mm. Qualitative results for the knife is shown in fig. 3.6b, to view results of the other objects, please visit the project website.

of different scales twice, while introducing a known pose perturbation between two measurements and recovered the pose perturbation using the method described in this section. We repeated this experiment six times for each of the objects in table 3.6 and report our pipeline's uncertainty in recovering the pose perturbations. We present our quantitative results in table 3.6 – our approach has a tracking uncertainty of about 2° in planar rotation and about 4 mm in translation.

Counter-intuitively, we also noted that generating a mesh of the object from the captured 3D representation and then using the mesh in the pipeline discussed in section 3.4.4 actually led to poorer pose estimation because the successive processing steps (normal calculation, integration, and meshing) reduced the quality of the mesh input. Since the 3D representation generated is strongly dependent on the viewpoint of the camera during the experiment meant that the 3D model was metrically incorrect for novel views.

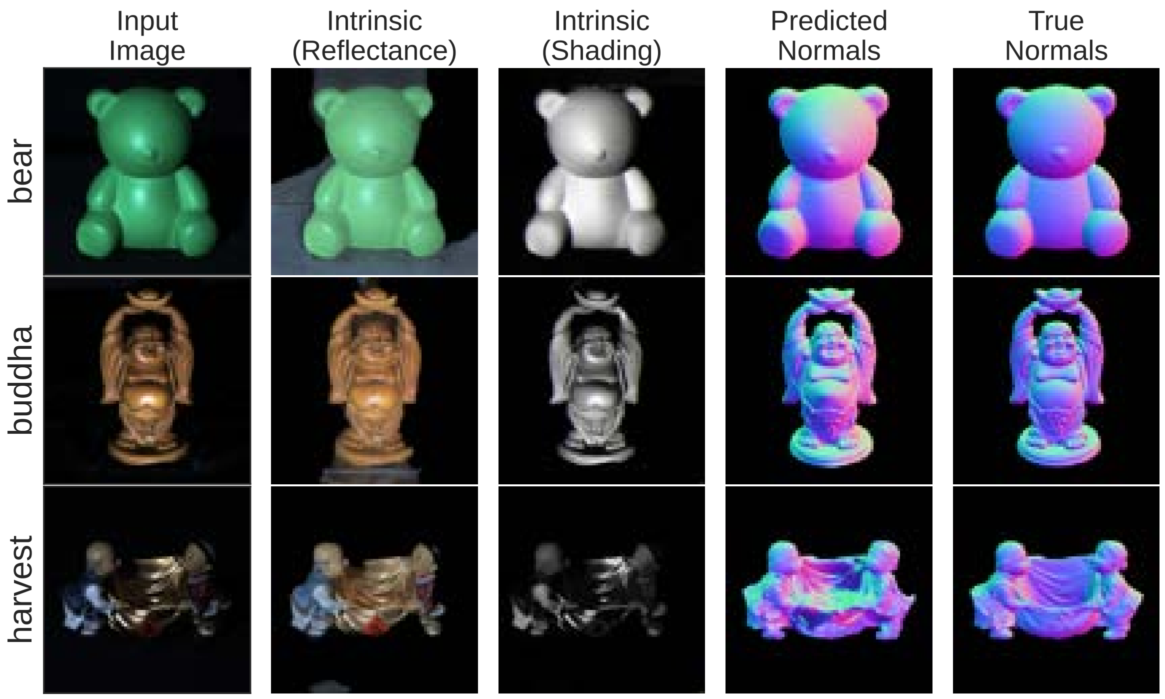


Figure 3.7: Our photometric stereo on colored, non-Lambertian objects using intrinsic image decomposition [118] on a dataset [119]. section 3.4.5.

3.4.5 Application to generic objects

Our previous experiments focused on Lambertian objects with diffuse, white paint in calibrated environments. Extending these techniques to more general objects is possible using off-the-shelf methods for intrinsic image decomposition. Intrinsic image decomposition separates natural color images into a reflectance image and a shading image. These shading images can be used as input to any typical photometric stereo method, as they estimate the isolated impact of lighting across the surface. There are many approaches to intrinsic image decomposition, including classic priors [73, 120] and data-driven learning methods [121, 122]. In our experiments, we use a recent method called PIE-Net[118] to perform the intrinsic image decomposition.

We show the results of this approach on three examples from the DiLiGenT[119] dataset, which contains several objects that include not only color but also general non-Lambertian materials. We show the results of a simple pipeline, using the pretrained PIE-Net model to obtain shading images. Photometric stereo is solved using the global, linear model that assumes distant point light sources and Lambertian shading. For each scene, DiLiGenT provides a single view of the scene with 95 different illumination conditions. We used 30-40 lighting directions for our reconstructions, each at PIE-Net's 256^2 image resolution.

Our results are shown in fig. 3.7. While this pipeline produced good results for some objects (bear, buddha), it struggled with the glossy, dark, and self-occluding harvest object. Nonetheless, considering that our pipeline used an off-the-shelf network and classic photometric stereo (without explicit handling of shadows or specularities), the results are promising. While this approach is far from the metric quality of our robotics setup in sections 3.4.1 and 3.4.4, we believe this quality of normals is potentially promising for vaccum gripper tasks such as those in section 3.4.2 or the deformation estimates in section 3.4.3. While we focused on the high end of precision in photometric stereo, this type of off-the-shelf pipeline enables generalization to generic objects, and produces potential utility in less demanding tasks.

3.5 Discussion and Future Work

Having a Lambertian reflectance requirement for our objects is restrictive and can be a barrier for our methods to apply to many manipulation tasks – future work can build on section 3.4.5 to enroll general objects into our pipeline. During this work, we observed that although we do well in inferring the cumulative shape of the object we often have high local errors – see e.g. the second inset of fig. 3.3b, and third inset of fig. 3.6a. We believe that these local errors are due to shadows that could not be resolved by our proposed method of inferring shape at a single pixel level with lights that are not co-incident with the camera. Further research is required on multiplexing the illumination sources to reduce the effect of shadows and to determine a better combination of light and camera locations. Lights collocated with cameras and on the robot workspace will possibly address some of the limitations of our work. Further research is also required for selecting alternative object representations. Current literature indicates that a triangle-based representation [75] or locally smooth patch-based representations [74] may work but will need a plethora of hand-tuned regularizers, hyper-parameters and a significant amount of computational effort to converge to a meaningful representation. Volumetric representations [76, 79] have certain advantages over patch-based representations in the context of robotic manipulation. Integrating volumetric representations with a robot workspace scaled controlled illumination approach is future work. In this work, we achieve a higher fidelity of measurement than some commercial depth sensors by imposing priors on object reflectances and controlling illumination. A natural extension of the current work would be to augment the performance of a commercial depth sensor by using it in conjunction with our approach.

3.6 Conclusions

In this work we demonstrated the application of classical techniques from photometric stereo to robotic manipulation through a robot workspace scaled controlled illumination system. We showed that, by enforcing a reflectance prior on the objects, reasoning about observed object intensities conditioned on the direction of illumination can yield accurate surface normals and identify surface depth discontinuities with very little computation. We also showed that the normals captured by our approach are significantly better than the ones derived from measurements with commercial depth sensors and we also evaluated the accuracy of our approach in capturing surface depth and normals. With the surface representations generated using our approach, we demonstrated three common manipulation tasks – picking up objects of arbitrary shape with a single point vacuum gripper, estimating bending deformation of a known object and estimating poses of Lambertian objects.

3.7 Retrospectives

The main body of this research was conducted between June 2021 through November 2022, largely coinciding with the global chip shortage, which made it difficult to procure machine vision cameras, Raspberry Pi computer boards, LED chips and reels. This global event somewhat hampered our planned extensive prototyping experiments. However we should note the following:

1. Using two different cameras and lenses (see section 3.3.1) made the process of geometric calibration process more involved. The two cameras with different fields-of-view were also not dramatically beneficial for any of our experiments. In hindsight, having two identical cameras in a binocular stereo configuration would have been more helpful, at the added cost of a larger robot manipulator payload. This observation, in addition to section 2.7, prompted us to select a robot-mounted binocular vision system.

2. The requirement for monochromatic Lambertian objects was primarily driven by the availability of only monochromatic cameras. In hindsight, our monochromatic camera could have been against a lookup table to enroll multi-color objects. [73, 118] demonstrate methods to decompose images captured under varied illumination into the intrinsic components and we show (section 3.4.5) that our methods also performs competitively in those datasets, but the resolution of those decompositions are unfortunately too low to be useful for a robotics task.

3. Finally, we did not experiment with polarized lights and cameras to incorporate reflective objects. Ideas from [123] may have been a good starting point for this. Our initial experiments with a near-infra-red (NIR) camera imaging a scene with transparent objects (borosilicate glass, transparent poly-carbonate plastic utensils) illuminated with directional 850 nm IR lights yielded some interesting data. However, incorporating specialized cameras, optics and illuminants into a robotics workspace posed some challenges beyond the scope of the project, so we elected to focus on a photometric stereo based sensing system for manipulation of Lambertian objects.

3.8 Supplementary Materials

3.8.1 Description of our algorithm for vacuum grasping

Our pipeline for detecting a flat face along an axis can be summarized with the following steps:

1. We first obtain the normals and the depth edges of the objects in the robot's workspace. The top insets of figs. 3.4a and 3.4b denote the normals of the scene and the bottom insets denote the depth edges of the scene.

2. For a given picking direction, we randomly generate a set of feasible vacuum gripper orientations centered along the picking direction. For example, if the picking direction is along the $+Z$ axis (see fig. 3.1a), our samples resemble picking directions that are normally distributed around the vertical axis. For picking along $\pm X$ or $\pm Y$ we adjust the sampling to only admit candidate orientations that avoid collision between the gripper and the table.

3. Next, we calculate the probability of each of the pixels in the imaged workspace to be approachable by the set of sampled vacuum gripper configurations. If the normal at a pixel is aligned with the picking direction, it is assigned a high probability of success. This is identical to the histogram back-projection step of the CAMShift algorithm.

4. Following this, we identify the largest cluster of feasible pixels using adaptive mean shift clustering. We adapt the scale and the orientation of the kernel as prescribed in [114]. This step identifies a candidate zone for our grasp. The top rows of figs. 3.4a and 3.4b demonstrate the output of this step, projected onto the normals of the scene calculated using the method described in section 3.3.3.

5. Finally, we score the suitability of executing a vacuum grasp on the selected area by noting that any surface patch with depth edges or surface textures would not be suitable for a vacuum grasp. To do this, we project the selected grasp areas on the scene's edges calculated using the method described in section 3.3.4 and generate a score based on the 0^{th} and 1^{st} pixel moments that indicate the number of edge pixels and their spread inside the chosen grasp area. Lower scores indicate that the selected patch does not have depth edges. The bottom rows of figs. 3.4a and 3.4b identify the selected grasp in areas without surface textures projected on the object depth edges calculated by our method in section 3.3.4.

We iteratively apply the above steps along all the directions reachable by our robot, namely $\pm X$, $\pm Y$, and Z for both cameras C_1 and C_2 (see fig. 3.1a for reference) and score the detected patches. We identify the highest-scoring patches as flat and "pickable". Figures 3.4a and 3.4b denote the corresponding "pickable" surfaces oriented along X and Z directions of the workspace, imaged with the robot-mounted camera C_1. The grasp areas geometrically correspond across two views imaged by C_1 and C_2 because the scene normals are calculated with respect to the robot's coordinate frame.

3.8.2 Description of our algorithm for measuring deformation

We assume that we have a uniformly dense high-quality mesh $\mathcal{M}$ of the object consisting of triangles with aspect ratio close to 1. We also assume that we have knowledge of the pose of $\mathcal{M}$ in the robot's coordinate frame and have access to a differentiable renderer $\mathcal{R}$ (we use PyTorch3D[124]) that takes the mesh $\mathcal{M}$, its pose and renders its silhouette and its surface normals in the viewport of two cameras C_1 and C_2. We clamp the card at one end and push it against a horizontal surface ($X - Y$ plane in fig. 3.5a) to induce buckling, and with our vision system, we measure the change in curvature on the object and capture its deformed geometry. From fig. 3.5a, we note that the normal $\mathbf{n}_f$ measured by the cameras at the face f of the object mesh $\mathcal{M}$ is dependent on the vertex positions $v_f^i,\ i = 1, 2, 3$. Given the initial states of all the vertices in $\mathcal{M}$ we calculate changes in the vertex positions such that the calculated normals at the face f match closely to the imaged normals at the same geometric location. We achieve that through the following steps:

1. We capture images of the scene using all the lights and calculate the scene normals using the method described in section 3.3.3. We also capture the silhouettes of the deforming object using our knowledge of the object's pose, geometry and the background. We denote the calculated surface normals and silhouettes of the *scene* as $(\mathbf{N}_S^{C_1}, \mathbf{N}_S^{C_2})$ and $(\mathbf{M}_S^{C_1}, \mathbf{M}_S^{C_2})$ respectively. Equivalent quantities are also *rendered* by the differentiable renderer $\mathcal{R}$ as $(\mathbf{N}_S^{C_1}, \mathbf{N}_R^{C_2})$ and $(\mathbf{M}_R^{C_1}, \mathbf{M}_R^{C_2})$ respectively.

2. If the states of all the vertices of $\mathcal{M}$ are exactly known, the measured and rendered images should be equivalent. We minimize the measured difference between the rendered and measured quantities by updating the vertex positions of $\mathcal{M}$.

3. Next, we calculate the difference between the rendered and measured quantities for the data corresponding to C_1 as:

$$\ell_{C_1}(\mathcal{M}) = \sum_k \left[1 - \langle \mathbf{N}_S^{C_1}, \mathbf{N}_R^{C_1} \rangle\right] + |\mathbf{M}_S^{C_1} - \mathbf{M}_R^{C_1}|^2 \tag{3.9}$$

   for all the k pixels in the camera C_1's viewport. We also obtain a similar difference ℓ_{C_2} for C_2. We compute a cumulative loss for both the views as

$$\ell(\mathcal{M}) = \alpha\ell_{C_1}(\mathcal{M}) + (1-\alpha)\ell_{C_2}(\mathcal{M}) \tag{3.10}$$

   where α is the fraction of the number of pixels imaging the deforming object in view C_1 with the total number of pixels imaging the object across both views C_1 and C_2.

4. Minimizing eq. (3.10) should, in theory, be enough for finding the new locations of the vertices of $\mathcal{M}$, but in practice, local measurements and the nature of gradient descent do not generate a smooth and physically plausible mesh through the gradient updates. To address this, we follow [75] and add two regularizers based on physics – $L_{Lap.}$: the mesh Laplacian regularizer which encourages geometrically smooth updates to the mesh vertices, L_{edge}: the mesh edge length regularizer that promotes mesh vertex updates that keep the aspect ratios of the mesh elements close to 1, to augment our loss in eq. (3.10).

5. Finally, we formulate our objective function as:

$$\min_{v_i} \ell(\mathcal{M}) + L_{Lap.}(\mathcal{M}) + L_{edge}(\mathcal{M}) \; \forall v_i \in \mathcal{M} \tag{3.11}$$

   We use gradient descent with backtracking line search to minimize eq. (3.11) above.

3.8.3 Description of our pose estimation pipelines

If a 3D model of an object is available, we define the pose estimation problem as finding the rigid transform $\mathbf{T}_{base}^{obj}$ which aligns the captured image of the object to a rendered equivalent given the camera parameters $\mathbf{K}_i$ and camera poses $\mathbf{T}_{base}^{C_i}$ for cameras C_1 *or* C_2. In addition to the 3D model $\mathcal{M}$ of the object being available to us, we also assume that we have access to a differentiable renderer $\mathcal{R}$ that can render the 3D model $\mathcal{M}$ given $\mathbf{K}_i$, $\mathbf{T}_{base}^{C_i}$ and $\mathbf{T}_{base}^{C_i}$. We use PyTorch3D[124] for our work. Using the method described in sections 3.3.3 and 3.3.4 we process our images captured by a camera – say C_1 of a view-port of $w \times h$ pixels, to obtain the *scene* normal map $\mathbf{N}_S \in \mathbb{R}^{w\times h\times 3}$, the *scene* depth edges $\mathbf{E}_S \in \mathbb{R}^{w\times h\times 1}$ and the object silhouette from the *scene* $\mathbf{M}_S \in \mathbb{R}^{w\times h\times 1}$.

With $\mathcal{R}(\mathcal{M}, \mathbf{T}_{base}^{obj}, \mathbf{T}_{base}^{C_1}, \mathbf{K}_1)$ we also *render* equivalent normal, depth edge and silhouette images $\mathbf{N}_R$, $\mathbf{E}_R$ and $\mathbf{M}_R$. Given that the object pose $\mathbf{T}_{base}^{obj}$ has been correctly estimated, the rendered and the scene images for normals, depth edges and object silhouettes should exactly match. In the following steps, we describe our steps for aligning the scene and rendered data. For our case, $\mathbf{T}_{base}^{obj}$ is parameterized by the position of the object along the X and Y axes and a rotation θ about Z axis.

1. To generate initial estimates of poses, we find correspondences between the measured surface normals $\mathbf{N}_S$ and rendered surface normals $\mathbf{N}_R$ for a given set of $\theta \in [0°, 180°]$. To do this, we identify object depth corners in $\mathbf{E}_S$ and look for patches in $\mathbf{N}_R$. The first inset of fig. 3.6a shows some corresponding depth edge corners overlaid on $\mathbf{N}_S$ and $\mathbf{N}_R$ for a particular θ. We find the matches by looking at 80×80 windows around the depth corners in $\mathbf{N}_S$ and match them to $\mathbf{N}_R$ for a particular θ using 3D cross-correlation. These matches, along with $\mathbf{K}_1$ can be used to compute the essential matrix between $\mathbf{N}_S$ and $\mathbf{N}_R$. We then decompose the essential matrices for all θ and identify the decomposition which produces a rigid transform closest to identity rotation. The corresponding θ_i is our initial guess for the object's orientation.

2. Next, we align the silhouettes $\mathbf{M}_S$ and $\mathbf{M}_R|(x, y, \theta_i)$ using a pixel-wise squared L_2 cost summed over all the k pixels in the $w \times h$ simulated and real camera view ports

$$\min_{x,y,\theta} \sum_k \left|\mathbf{M}_S^k - \mathbf{M}_R^k|_{(x,y,\theta)}\right|_2^2 \tag{3.12}$$

3. Following which, we align the rendered and measured surface normals $\mathbf{N}_R$ and $\mathbf{N}_S$ respectively about 4 pyramid levels. To do this we generate the normal images ${}^i\mathbf{N}_S, {}^i\mathbf{N}_R,\ i \in (1, 2, 3, 4)$ across different scales and calculate the cosine similarity between the k corresponding pixels inside the corresponding scaled silhouettes ${}^i\mathbf{M}_S$:

$$\min_{x,y,\theta} \sum_{k \in {}^i\mathbf{M}_S} \left[1 - \langle {}^i\mathbf{N}_S^k, {}^i\mathbf{N}_R^k|_{(x,y,\theta)}\rangle\right] \forall i \tag{3.13}$$

   where $i = 1$ is the lowest pyramid level.

4. Finally, following Chaudhury et al.[5] we align the rendered and measured depth edge images $\mathbf{E}_S$ and $\mathbf{E}_R$ by minimizing the mean squared error between the Euclidean distance transforms (EDT) of images $\mathbf{E}_S$ and $\mathbf{E}_R$. Using the definition of Euclidean distance transform from [56], we formulate our dense edge alignment cost as

$$\min_{x,y,\theta} \sum_k \left|\mathrm{EDT}(\mathbf{E}_\mathrm{S}) - \mathrm{EDT}(\mathbf{E}_\mathrm{R}|_{(\mathrm{x},\mathrm{y},\theta)})\right|_2^2 \tag{3.14}$$

   for all the k pixels in the $w \times h$ simulated and real camera view ports.

If a **3D model of the object is unavailable**, we define the pose estimation problem as finding the rigid transform $\mathbf{T}$ that aligns the two 3D representations captured by our system as the object moves.

1. We first image the object with our camera – the leftmost inset of fig. 3.6b shows the image of a folding knife captured by C_1 with L_7 illuminating the scene.

2. Next, we calculate the normals of the scene and identify the object depth edges – these are shown as the second and third insets of fig. 3.6b. These normals are only used to generate a 3D representation of the object.

3. Following that we calculate the point cloud of the surface of the object and also calculate per point features – we use Fast Point Feature Histograms (FPFH) [58]. The fourth inset of fig. 3.6b shows the point cloud of the object with the salient points colored. We note that these points roughly indicate the depth discontinuities by referring between the third and fourth insets of fig. 3.6b.

4. We repeat the steps above for the data obtained at the new pose of the object.

5. Finally, we use the FPFH in a robust point feature based pose estimation pipeline (we use FGR [125]) to generate initial pose estimates and then use point-to-plane ICP [112] to solve for the change in pose of the object between the two measurements. The last two insets of fig. 3.6b shows the two initial measurements of the object on top and the latter measurement registered to the former measurement on the bottom. Point cloud normals need to be re-calculated in the object's frame for the ICP based refinement step.

3.8.4 Implementation details and hyperparameters

Step	Processing + [overhead]
Capture (fig. 3.1b)	0.20 (per cam.) + [5]
Normal (eq. 3.3,3.4)	3.5 (1000^2 px)
Depth (section 3.3.4)	0.15 (1000^2 px)
Edges (section 3.3.4)	0.05 (1000^2 px)
Pickup (fig. 3.4a)	0.5 (1000^2 px)
Pickup (robot)	20 (pick and drop)
Bending (eq. (3.11))	15 (1500 vert, 300 iter)) + [5]
pose est. (eq. 3.12, 3.13, 3.14)	12.5 - 15.5 (150 iter/eq.) + [5]

Table 3.7: Approximate time breakdown of various steps in seconds for processing a 1000^2 px image

In this work, we attempt to capture and process all the captured data online for the demonstrated perception tasks. To do this, we implemented almost all of the pipelines with strong GPU support. The algorithms for solving the optimization problems associated with normal estimation (eq. (3.6)), bending estimation (eq. (3.11)) and all the alignment costs (eqs. (3.12) to (3.14)) have been implemented with custom GPU back-ends, which led to massive speedup even with a resolution of 50px/mm^2. This lets us perform all the tasks in less than 30 seconds time per task per object at full resolution. A breakdown of the time take by each step is shown in table 3.7. We also discuss our major design decisions and the observed effects of important hyperparameters of the system we discovered during implementing the system below.

3.8.5 Practical considerations in design

We implemented the system demonstrated in the work on a robot table of 1.5m $\times$ 1.5m, The black mat of size (460mm$\times$610mm) in fig. 3.1a denotes the dexterous workspace of the robot. These dimensions governed the placement of the light sources, their power, and some of the camera hyperparameters (exposure, f-stops) for our experiments. To achieve good illumination of the workspace (see section 3.8.6, item 1), we observed that a 45W white light source was sufficient when placed 500mm away from the objects. So we placed six light sources in an approximately hexagonal pattern around the center of the workspace with a radius of approximately 500mm. Theoretically, as long as there is a measurable effect of a directional light source on the image of the object, our approach would work – however the more pronounced the effect, the less is the noise in measuring it with a camera. With the constraints above, and our choice of hyperparameters discussed in section 3.8.6, we could obtain a reasonable performance with our system.
As the lights L_1 through L_6 (figs. 3.1a and 3.2) were used as grazing directional light sources they were oriented towards the center of the workspace. The exact orientation of the light panels were not measured as those were implicitly recovered by our illumination models discussed in section 3.3.2. However, through our fixtures, we ensured that the light sources stationary during the experiment. Light L_7 was solely used to estimate the shadows and highlights when the scene was illuminated by L_1 through L_6, and an illumination model was not recovered from it. We experimented with using L_7 as another source (by using seven light sources instead of six in eqs. (3.3) and (3.4)) and the quality of the normal measurements degraded significantly because of the poorer initial estimates of the shadow-illumination matrix $\mathbf{W}_k$ in eq. (3.5) and a resulting poorer performance of eqs. (3.6) and (3.7). Our two step illumination model requires at least three light sources to illuminate every portion of the scene for eq. (3.5) to be valid. Except simpler shapes (like a hemisphere or a pyramid) a minimum of three light sources are not guaranteed to illuminate all the portions on the surface, unless they are at a higher angle of incidence to the surface. However, a higher angle of incidence does not accentuate the depth discontinuities as much as low angle of incidence illumination, which was a requirement for our case. These constraints led us to choose six illumination sources on the table. We observed from initial experiments that fewer sources performed poorer – especially for more complex shapes and, inspired by small-scale photometric stereo sensors ([5, 20, 90, 91]) we decided to have six illumination sources.

3.8.6 Hyperparameters and their effects

The performance of our approach is dependent on some crucial hyper-parameters. We note the important hyper-parameters below:

- Gains, exposure times and lens f-stop numbers are critical hyperparameters for capturing the effects of directional illumination on the scene consistently. For all our experiments, in addition to setting the sensor gamma to 1, we set the sensor gains and black levels to zero and turn off automatic exposure and gain settings. We adjust the lens f-stops and exposure times so that the brightest spot in the image under all illumination conditions is between 75% to 85% of the maximum brightness (`uint16_max` 65535). For our setup, an f-stop of 1.6 on the 12mm focal length lens and 1.2 on the 16mm focal length lens and an exposure time of 2.5 milliseconds worked well. These parameters are also dependent on the color and power of the ambient illumination of the room, distance between the light source and the scene, reflectivity of the ceiling and walls, and the color of the wall.

- We used backtracking line search for all the gradient descent steps (eqs. (3.4) and (3.11) to (3.14)) in this work. The initial learning rates were 0.1, with a c value of 10^{-5} and an annealing factor of 0.9. We terminated the gradient descent when the step size was lower than 10^{-7} or 150 gradient descent steps have been completed.

- For the pickup tasks, we selected the edge pixel score threshold by setting it to a low value as we only grasped geometrically flat patches. The hyperparameter value selected patches with less than 3–5% of the pixels labeled as edges.

- For estimating bending deformation, we found that locating the cameras (C_1 and C_2 in fig. 3.5a) such that both yield similar sized images gave us the best results while estimating the deformation of the card. For all our experiments in fig. 3.5c, we made sure that the value of α in eq. (3.10) was in between 0.45 to 0.55.

- Finally, for the global registration step in fig. 3.6b, we used a FPFH size of 35, and a voxel downsampling factor of 1.5 in the implementation available on Open3D ([126]).

IV

Moving Lights and Cameras

4 Multi-view, multi-illumination capture for learning 3D object representations

Synthesizing accurate geometry and photo-realistic appearance of small scenes is an active area of research with compelling use cases in gaming, content creation, virtual reality, and robotics. When applying scene geometry and appearance estimation techniques to robotics, we found that the narrow cone of possible viewpoints due to the limited range of robot motion and scene clutter caused current estimation techniques to produce poor quality estimates or even fail. On the other hand, in robotic applications, dense metric depth can often be measured directly using stereo and illumination can be controlled. Depth can provide a good initial estimate of the object geometry to improve reconstruction. In this work we demonstrate a method to incorporate dense metric depth into the training of neural 3D representations, address an artifact observed while jointly refining geometry and appearance by disambiguating between texture and geometry edges, and propose a new pipeline which accelerates training of neural fields by using dense metric depth. We also discuss a multi-flash stereo camera system developed in-house to capture the necessary data for our pipeline and show results on relighting and view synthesis with a few training views.

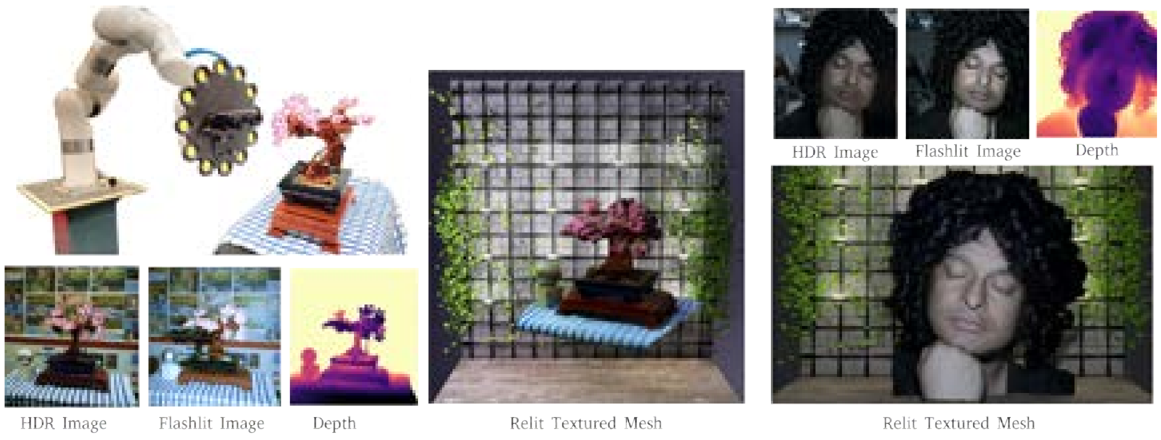


Figure 4.1: We present an approach for the photo-realistic capture of small scenes by incorporating dense metric depth, multi-view, and multi-illumination images into neural 3D scene understanding pipelines. We use a robot mounted multi-flash stereo camera system, developed in-house, to capture the necessary supervision signals needed to optimize our representation with a few input views. The reconstruction of the LEGO plant and the face were generated with 11 and 2 stereo pairs respectively. We relight the textured meshes using [127]. Background design by [128].

4.1 Introduction

Capturing photo realistic appearance and geometry of scenes is a fundamental problem in computer vision and graphics with a set of mature tools and solutions for content creation [129, 130], large scale scene mapping [131], augmented reality and cinematography [132, 133, 134]. Enthusiast level 3D photogrammetry, especially for small or tabletop scenes, has been supercharged by more capable smartphone cameras and new toolboxes like RealityCapture and NeRFStudio. A subset of these solutions are geared towards view synthesis where the focus is on photo-realistic view interpolation rather than recovery of accurate scene geometry. These solutions take the "shape-radiance ambiguity"[135] into stride by decoupling the scene transmissivity (related to geometry) from the scene appearance prediction. But without of diverse training views, several neural scene representations (e.g. [136, 137, 138]) are prone to poor shape reconstructions while estimating accurate appearance.

By only reasoning about appearance as cumulative radiance weighted with the scene's transmissivity, one can achieve convincing view interpolation results, with the quality of estimated scene geometry improving with the diversity and number of training views. However, capturing a diverse set of views, especially for small scenes, often becomes challenging due to the scenes' arrangement.

Without of metric depth measurements, researchers have used sparse depth from structure from motion [139, 140], and dense monocular depth priors [141] to improve reconstruction, with a focus on appearance. Assimilating dense non-metric depth (e.g. [142, 143]) is often challenging due to the presence of an unknown affine degree of freedom which needs to be estimated across many views.

However, without diversity of viewpoints, measuring the geometry directly is often useful. Several hardware solutions for digitizing objects exist, ranging from consumer level 3D scanners (e.g. [144]), and room scale metrology devices ([145, 146, 147]) to high precision hand held 3D scanners (e.g. [148]). Although these systems measure geometry very accurately, they interpret appearance as diffuse reflectance and often fall short in modelling view-dependent appearance.

Despite the known effectiveness of incorporating depth and widespread availability of dense metric depth sensors in smartphone cameras ([149, 150]) and as standalone devices ([151, 152], incorporation of dense metric depth into neural 3D scene understanding is underexplored.

In this work:

1. we present a method to incorporate dense metric depth into the training of neural 3D fields, enabling state-of-the-art methods to use dense metric depth with minor changes.
2. We investigate an artifact (fig. 4.4) commonly observed while jointly refining shape and appearance. We identify its cause as existing methods' inability to differentiate between depth and texture discontinuities. We address it by using depth edges as an additional supervision signal.
3. We present a new approach to accelerate training of neural fields for photo-realistic scene capture and relighting from multi-view and multi-illumination images by incorporating metric depth.

We demonstrate our ideas using a robot mounted multi-flash stereo camera rig developed in-house. This device allows us to capture a diverse range of scenes with varying complexity in both appearance and geometry. Using the captured data, we demonstrate results in reconstruction, view interpolation, geometry capture, and relighting with a few views. We hope that our full-stack solution comprising of the camera system and algorithms will serve as a test bench for automatically capturing small scenes in the future. Additional results may be viewed at https://stereomfc.github.io

4.2 Related Work

View synthesis and reconstruction of shapes from multiple 3D measurements is an important problem in computer vision with highly efficient and general solutions like volumetric fusion ([153]),

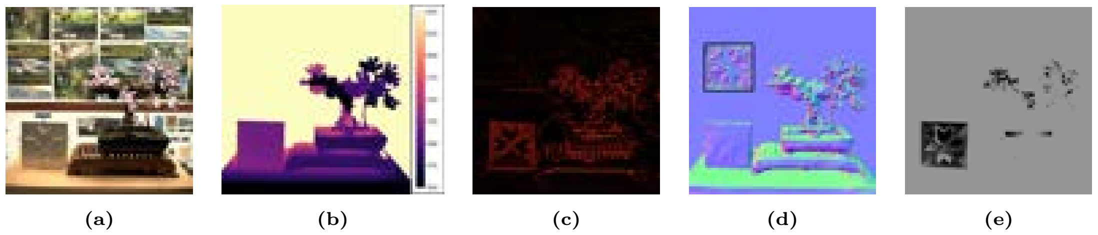


Figure 4.2: A snapshot of the important supervision signals. We capture a high dynamic range image [176] and display it after tonemapping [177] in fig. 4.2a. Figure 4.2b shows the scene depth (in mm) from stereo. Figure 4.2c displays the likelihood of each pixel falling on a depth edge. Figure 4.2d shows the object surface normals. We note that unlike conventional stereo matching ([178]), [171] returns locally smooth surfaces and often ignores local texture variations but is less noisy. The inset shows the surface normals on the textured aluminum plate calculated as gradients of depth from conventional stereo matching. Finally, fig. 4.2e identifies the pixels with the largest appearance variation due to moving lights.

screened Poisson surface reconstruction ([154]), patch based dense stereopsis ([155]) and joint refinement of surface and appearance [156]. While these continue to serve as robust foundations, they fall short in capturing view-dependent appearance. Additionally, even with arbitrary levels of discretization, they often oversmooth texture and surfaces due to data association relying on weighted averages along the object surface.

Recent neural 3D scene understanding approaches (e.g. [157, 158, 159]) have avoided this by adopting a continuous implicit volumetric representation to serve as the geometric and appearance back-end of the view synthesizer. Together with continuous models, reasoning about appearance as radiance, and high frequency preserving embeddings[160], these approaches serve as highly capable view interpolators by reliably preserving view dependent appearance and minute geometric details. More recent work has included additional geometric priors in the form of monocular depth supervision ([141]), sparse depth supervision from structure-from-motion toolboxes ([161]), dense depth maps ([162, 163]), patch based multi-view consistency [164], and multi-view photometric consistency under assumed surface reflectance functions ([165]). Our work builds on the insights from using dense depth supervision to improve scene understanding with only a few training views available.

Novel hardware is often used for collecting supervision signals in addition to color images to aid 3D scene understanding. [166] demonstrate a method to incorporate a time-of-flight sensor. [167] demonstrate a method to extract geometric and radiometric cues from scenes captured with a commercial RGBD sensor and improve view synthesis with a few views. Event based sensors have also been used to understand poorly lit scenes with fast moving cameras ([168, 169]). Researchers have also combined illumination sources with cameras to capture photometric and geometric cues for dense 3D reconstruction of scenes with known reflectances ([6, 170]). Similarly, [83, 84, 85] capture geometry and reflectance of objects by refining multi-view color, depth and multi-illumination images. Given the recent advances in stereo matching ([171]) we use a stereo camera to collect data for view synthesis to disambiguate between shape and appearance at capture.

Pairing illumination sources with imaging can improve reasoning about the appearance in terms of surface reflectance parameters. [84, 172, 173] approaches the problem of material capture using a variety of neural and classical techniques. [83, 139, 174, 175] leverage recent neural scene understanding techniques to jointly learn shape and appearance as reflectance of the scene. Our work also pairs illumination sources with stereo cameras to capture multi-illumination images from the scene and we build on modern neural techniques for view synthesis and relighting the scene.

4.3 Methods

We describe our method of incorporating dense metric depth in section 4.3.2 which enables a variety of neural 3D representations to use it. In section 4.3.3 we jointly optimize shape and appearance of a scene using information about scene depth edges. In section 4.3.4, we propose a new method of neural view synthesis: AdaShell^{++}that accelerates training by incorporating dense depth. In section 4.3.5 we discuss metric depth augmented versions of prior works.

4.3.1 Primer on neural 3D fields

Our scene representation consists of two neural networks – an intrinsic network $\mathcal{N}(\theta)$ and an appearance network $\mathcal{A}(\phi)$ which are jointly optimized to capture the shape and appearance of the object. $\mathcal{N}(\theta)$, an a multi-layer perceptron (MLP) with parameters (θ) with multi-level hash grid encoding ([159]), is trained to approximate the intrinsic properties of the scene – the scene geometry as a neural signed distance field $\mathcal{S}(\theta)$ and an embedding $\mathcal{E}(\theta)$. The appearance network $\mathcal{A}(\phi)$ is another MLP which takes $\mathcal{E}(\theta)$ and a frequency encoded representation of the viewing direction and returns the scene radiance along a ray.

The neural signed distance field $\mathcal{S}(\theta)$ is optimized to return the signed distance of a point from its nearest surface $\mathcal{S}(\theta) : \mathbb{R}^3 \to \mathbb{R}$. The surface of the object can be obtained from the zero-level set of $\mathcal{S}(\theta)$ – i.e. for all surface points $\mathbf{x}_s \in \mathbb{R}^3 \mid \mathcal{S}(\mathbf{x}_s|\theta) = 0$. We train $\mathcal{S}(\theta)$ by minimizing a geometric loss ℓ_D (eq. (4.5)). We follow [157] to transform the distance of a point $\vec{p_i} = \vec{r}|_{t_i}$ in a ray to its closest surface $s_i = \mathcal{S}(\theta, \vec{p_i})$ to the scene density (or transmissivity).

$$\Psi_\beta(s) = \begin{cases} 0.5\exp(\frac{s}{\beta}), & s \leq 0 \\ 1 - 0.5\exp(\frac{-s}{\beta}), & \text{otherwise.} \end{cases} \tag{4.1}$$

To render the color $\mathbf{C}$ of a single pixel of the scene at a target view with a camera centered at $\vec{o}$ and an outgoing ray direction $\vec{d}$, we calculate the ray corresponding to the pixel $\vec{r} = \vec{o} + t\vec{d}$, and sample a set of points t_i along the ray. The networks $\mathcal{N}(\theta)$ and $\mathcal{A}(\phi)$ are then evaluated at all the $\mathbf{x}_i$ corresponding to t_i and the per point color $\mathbf{c}_i$. The transmissivity τ_i is obtained and composited together using the quadrature approximation from [179] as:

$$\mathbf{C} = \sum_i \exp(-\sum_{\mathrm{j<i}} \tau_\mathrm{j}\delta_\mathrm{j})(1 - \exp(-\tau_\mathrm{j}\delta_\mathrm{j}))\mathbf{c}_\mathrm{i}, \quad \delta_\mathrm{i} = \mathrm{t_i} - \mathrm{t_{i-1}} \tag{4.2}$$

The appearance can then be learned using a loss on the estimated and ground truth color $\mathbf{C}_{gt}$

$$\ell_C = \mathbb{E}\left[||\mathbf{C} - \mathbf{C}_{gt}||^2\right] \tag{4.3}$$

The appearance and geometry are jointly estimated by minimizing the losses in eq. (4.4) using stochastic gradient descent [180].

$$\ell = \ell_C + \lambda_g \ell_D + \lambda_c \mathbb{E}(|\nabla^2_{\mathbf{x}} \mathcal{S}(\mathbf{x}_s)|) \tag{4.4}$$

λs are hyperparameters and the third term in eq. (4.4) is the mean surface curvature minimized against the captured surface normals. As the gradients of the loss functions ℓ_C and ℓ_D propagate through $\mathcal{A}$ and $\mathcal{N}$ (and $\mathcal{S}$ as it is part of $\mathcal{N}$) the appearance and geometry are learned together.

4.3.2 Incorporating dense metric depth

Prior work has jointly learned $\mathcal{S}, \mathcal{N}, \mathcal{A}$ with only multi-view images, in contrast, we have access to estimates of true surface depth to any surface point $\mathbf{x}_s$. In this section we describe our method to directly optimize $\mathcal{S}$ with depth.

The depth estimates can be directly used to optimize appearance and render surfaces following [156, 181, 182]. However, to be able to approximate view-dependent appearance, we elect to learn a continuous and locally smooth function that approximates the signed distance function of the surface $\mathbf{x}_s$ which can then be transformed to scene density (eq. (4.1)). To do this, we roughly follow

[183] and consider a loss function of the form

$$\ell_D(\theta) = \ell_{\mathbf{x}_s} + \lambda \mathbb{E}(||\nabla_{\mathbf{x}} \mathcal{S}(\mathbf{x}^{\Delta}, \theta)|| - 1)^2 \tag{4.5}$$
$$\text{where, } \ell_{\mathbf{x}_s} = \frac{1}{N} \Sigma_{\forall \mathbf{x}} \left[\mathcal{S}(\mathbf{x}, \theta) + 1 - \langle \nabla_{\mathbf{x}} \mathcal{S}(\mathbf{x}, \theta), \mathbf{n}_x \rangle \right]$$

Through the two components, the loss encourages the function $\mathcal{S}(\mathbf{x}, \theta)$ to vanish at the observed surface points and the gradients of the surface to align at the measured surface normals. The second component in eq. (4.5) is the Eikonal term ([184]) which encourages the gradients of $\mathcal{S}$ to have a unit L_2 norm everywhere. The individual terms of eq. (4.5) are averaged across all samples in a batch corresponding to N rays projected from a known camera.

The Eikonal constraint applies to the neighborhood points $\mathbf{x}_s^{\Delta}$ of each point in $\mathbf{x}_s$. [183] identifies candidate $\mathbf{x}_s^{\Delta}$ through a nearest neighbor search, where as [157] identifies $\mathbf{x}^{\Delta}$ through random perturbations of the estimated surface point along the projected ray. As we have access to depth maps, we identify the variance of the neighborhood of $\mathbf{x}_s$ through a sliding window maximum filter on the depth images. This lets us avoid expensive nearest neighbor lookups for a batch of $\mathbf{x}_s$ to generate better estimates of $\mathbf{x}_s^{\Delta}$ than [157] at train time. As a result, convergence is accelerated – ($\sim 100\times$ over [183]) with no loss of accuracy. As we used metric depth, noisy depth estimates for parts of the scene are implicitly averaged by $\mathcal{S}$ optimized by minimizing eq. (4.5), making us more robust to errors than [141]. We provide more details in appendix A.

4.3.3 Incorporating depth edges in joint optimization of appearance and geometry

Prior works show the benefits of jointly refining geometry and appearance as it yields lesser independent hyperparameters, some degree of geometric super-resolution, and more stable training. However, some pathological cases may arise when the scene has a large variation in appearance corresponding to a minimal variation in geometry across $\mathbf{x}_s$ and $\mathbf{x}_s^{\Delta}$. We investigate this effect by considering an extreme case – a checkerboard printed on matte paper with an inkjet printer, where there is no geometric variation (planar geometry) or view dependent artifacts (ink on matte paper is close to Lambertian) corresponding to a maximum variation in appearance (white on black). The qualitative results are presented in fig. 4.4.

Consider two rays $\vec{r}_{\mathbf{x}_s}$ and $\vec{r}_{\mathbf{x}_s^{\Delta}}$ connecting the camera center and two neighboring points $\mathbf{x}_s$ and $\mathbf{x}_s^{\Delta}$ on two sides of an checkerboard edge included in the same batch of the gradient descent. The total losses for those rays depend on the sum of the geometry and appearance losses (eq. (4.4)). By default, the current state of the art ([159, 185, 186] etc.) do not have a mechanism to disambiguate between texture and geometric edges (depth discontinuities).

As seen in fig. 4.4, given unsuitable hyperparameters, the approaches will continue to jointly update both geometry and appearance to minimize a combined loss (eq. (4.4)). This can often result in pathological reconstructions (left insets in fig. 4.4) due to ℓ_C gradients dominating over ℓ_D. By gradually increasing the modelling capacity of $\mathcal{N}$ we can somewhat avoid this artifact and force the gradient updates to focus on $\mathcal{A}$ to minimize the cumulative loss. [159] recognize this and provide an excellent set of hyperparameters and training curricula to gradually increase the modelling capacity of $\mathcal{N}(\phi)$. This results in remarkable geometric reconstructions for well known datasets ([187, 188]). Alternatively, if we have per-pixel labels of geometric edges ($\mathbb{E}$), we can preferentially sample image patches with low variation of geometric features when the model capacity is lower ($\mathcal{S}(\theta)$ tends to represent smoother surfaces), and focus on image patches with geometric edges when the model capacity has increased. The modelling capacity of $\mathcal{A}(\phi)$ never changes.

Figure 4.3 describes our sampling procedure while learning a scene with a variety of geometric and texture edges. Equation (4.6) is used to draw pixel samples – the probability of drawing pixel p_i is calculated as a linear blend of the likelihood that it belongs to the set of edge pixels $\mathbb{E}$ and α

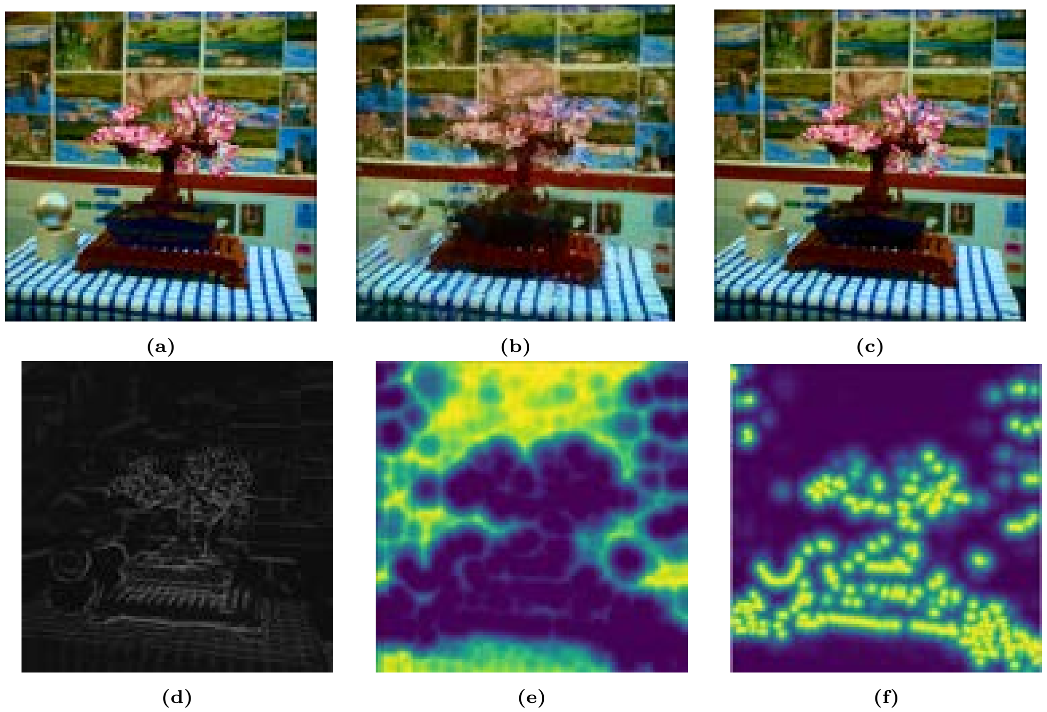


Figure 4.3: Overview of our sampling process during training. Figure 4.3a is the ground truth test image. Figure 4.3b is the reconstruction of the test image after training has progressed 15% (15k gradient steps), fig. 4.3c is the reconstruction of the test image at the end of training (100k gradient steps). Figure 4.3d denotes the per-pixel likelihoods of depth edges in the scene at the same view captured with our device. We note in fig. 4.3b, the parts of scene with complicated geometry (foliage with many depth edges) have lower fidelity of appearance in the reconstruction at an earlier stage of training, which gradually improves in fig. 4.3c. Figure 4.3e indicates the per-pixel sampling likelihood *if the test view were to be used for training* at a training progress of 10%, fig. 4.3f indicates the same at a progress of 90%. Equation (4.6) is used to draw the samples: $\alpha = 0.1$ and 0.9 respectively for figs. 4.3e and 4.3f. Brighter color indicates higher sampling likelihood.

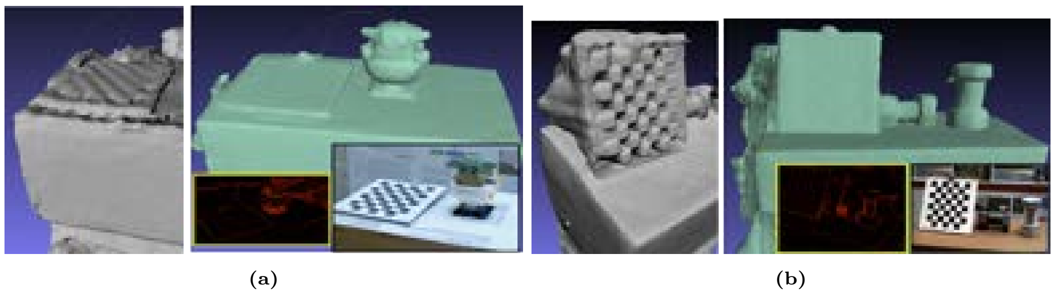


Figure 4.4: We demonstrate a corner case of jointly refining appearance and geometry. The left insets of figs. 4.4a and 4.4b are the scene geometries recovered in the worst cases, the right insets display the better meshes recovered using the method described in section 4.3.3. An image used for training and the edge map used for sampling are in the insets. We recommend zooming into the figure for details. Corresponding quantitative results are in table 4.4.

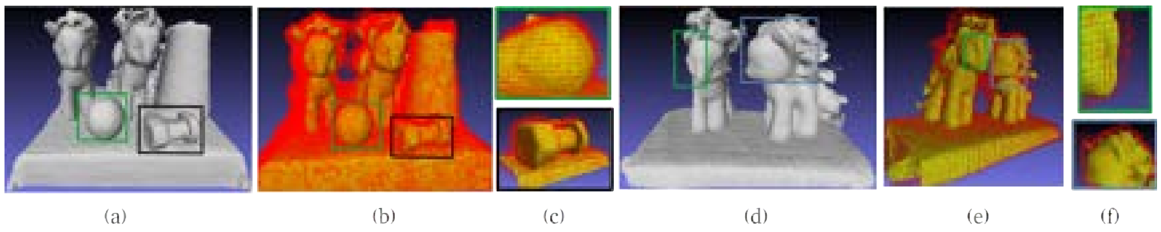


Figure 4.5: AdaShell^{++}recovers sampling volumes similar to [186]. Figures a,d are the geometries recovered after optimization of eq. (4.5) and is the starting point of AdaShell^{++}. Figures b, e display the sampling volumes around the starting geometry after AdaShell^{++}has converged. Details in section 4.3.4.

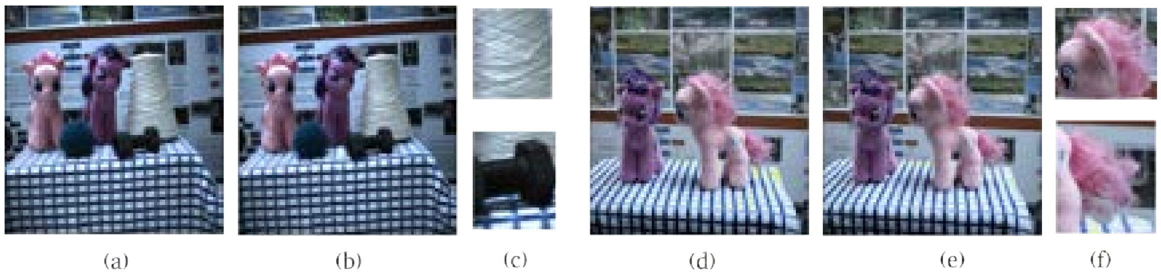


Figure 4.6: Reconstructions with AdaShell^{++}. Figures (a,d) are ground truth test images, (b,e) are reconstructed views, and (c,f) are zoomed in crops. Optimized geometries in fig. 4.5(a,d). Both the scenes were trained on 9 and tested on 1 view for 30K gradient steps. For the diffuse scenes above, AdaShell^{++}recovered thin 'shells' around the estimated geometry, accelerating convergence and rendering.

is a scalar ($\alpha \in [0, 1]$) proportional to the progress of the training.

$$P(p_i|\alpha) = (1-\alpha)P(p_i \in \mathbb{E}) + \alpha P(p_i \notin \mathbb{E}) \tag{4.6}$$

To preserve the geometric nature of the edges while ruling out high frequency pixel labels, we use Euclidean distance transform ([189]) to dilate $\mathbb{E}$ before applying eq. (4.6). We provide implementation details in appendix A for reproducibility.

4.3.4 AdaShell^{++}: Accelerating training with dense depth

The slowest step in training and inference for neural volumetric representations is generally the evaluation of eq. (4.2). In this section we describe our method to accelerate training by incorporating metric depth.

A method to make training more efficient involves drawing the smallest number of the most important samples of t_i for any ray. The sampling of t_i is based on the current estimate of the scene density and although these samples can have a large variance, given a large number of orthogonal view pairs (viewpoint diversity), and the absence of very strong view dependent effects, the training procedure is expected to recover an unbiased estimate of the true scene depth (see e.g. [190]). We can accelerate the convergence by a) providing high quality biased estimate of the scene depth and b) decreasing the number of samples for t_i along the rays.

Given the high quality of modern deep stereo (we use[171]) and a well calibrated camera system, stereo depth can serve as a good initial estimate of the true surface depth. We use stereo depth, aligned across multiple views of the scene to pre-optimize the geometry network $\mathcal{S}(\theta)$. The other channels $\mathcal{E}(\theta)$ of $\mathcal{N}$ remain un-optimized. A pre-optimized $\mathcal{S}$ can then be used for high quality estimates of ray termination depths.

[157, 158, 181] recommend using root finding techniques (e.g. bisection method) on scene transmissivity (eq. (4.1)) to estimate the ray termination depth. The samples for eq. (4.2) are then generated around the estimated surface point. Drawing high variance samples as $\mathcal{N}$ and $\mathcal{A}$ are jointly optimized reduces the effect of low quality local minima, especially in the initial stages of the optimization. As we have a pre-trained scene transmissivity field ($\mathcal{S}$ transformed with eq. (4.1)), we can draw a few high-quality samples to minimize the training effort.

We found uniformly sampling around the estimated ray-termination depth (UniSurf^{++}baseline in section 4.3.5 and fig. 4.4) to be unsuitable. Instead, we pre-calculated a discrete sampling volume by immersing $\mathcal{S}$ in an isotropic voxel grid and culling the voxels which report a lower than threshold scene density. We then used an unbiased sampler from [157] to generate the samples in this volume. This let us greatly reduce the number of root-finding iterations and samples, while limiting the variance by the dimensions of the volume along a ray. As the training progresses, we decrease the culling threshold to converge to a thinner sampling volume around the surface while reducing the number of samples required.

We show the sampling volume (at convergence) and our reconstruction results in figs. 4.5 and 4.6 respectively. We retain the advantages of volumetric scene representation as demonstrated by the reconstruction of the thin structures in the scene, while reducing training effort. We dub our method AdaShell^{++}to acknowledge [186], which demonstrates a related approach to accelerate inference.

4.3.5 Baselines augmented with depth

In addition to AdaShell^{++}we augment three other state-of-the-art methods to incorporate metric depth:

VolSDF^{++}is our augmented version of [157, 185], where we use the metric depths along the rays to optimize the geometry (eq. (4.4)). All the other parts of the original approaches, including the method for generating samples for eq. (4.2), and analogs for eq. (4.1) are left intact.

NeUS^{++}is our augmented version of [159], where we also use metric depths to optimize the geometry. The rest of the algorithm including the background radiance field is left intact.

UniSurf^{++}is our deliberately hamstrung version of [181] where we force the the samples generated for eq. (4.2) to have a very low variance around the current biased estimate of the surface. This makes the algorithm necessarily indifferent to the relative magnitudes of ℓ_D and ℓ_C in eq. (4.4), and helps us exaggerate the pathological effects of not segregating texture and geometric edges. We choose to name the method UniSurf^{++}after we (and [191]) observed that original method was vulnerable to this artifact under certain hyperparameter choices. Across all the methods, we implement and train $\mathcal{N}(\theta)$ following [159], and all of them use the same appearance network $\mathcal{A}(\phi)$.

AdaShell^{++}and UniSurf^{++}require a warm start – $\mathcal{S}$ pre-optimized for 5K gradient steps. All methods except UniSurf^{++}use the sampling strategy from section 4.3.3. More details are in appendix A.

4.4 Setup and Dataset

4.4.1 A multi-flash stereo camera

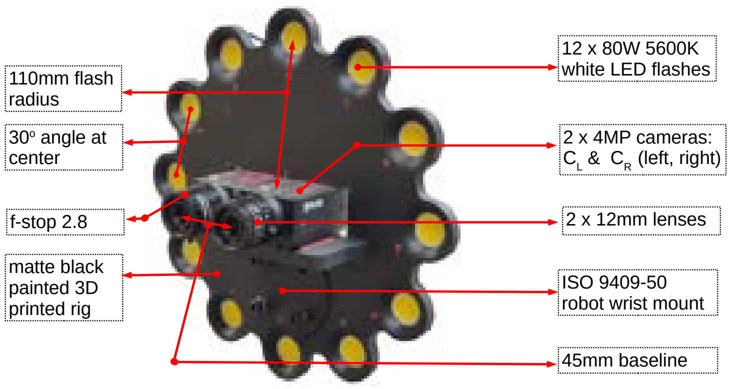

110mm flash radius
30º angle at center
f-stop 2.8
matte black painted 3D printed rig
12 x 80W 5600K white LED flashes
2 x 4MP cameras: C_L & C_R (left, right)
2 x 12mm lenses
ISO 9409-50 robot wrist mount
45mm baseline

Figure 4.7: Our system used to capture the data in fig. 4.2.

In addition to multi-view images, scene depth and depth edges are valuable signals to train neural 3D representations. To capture all the supervision signals, we designed and fabricated a multi-flash stereo camera based on insights from [192, 193]. We capture data by moving our camera rig in front of objects. For each camera pose we capture a stereo pair of high dynamic range (HDR) images, two depth maps from left and right stereo, two corresponding image aligned surface normals (as gradient of depth maps). We augment the HDR images by in-painting them with the depth edges before using [171] to preserve intricate surface details in the depth maps. Additionally we capture 12 pairs of multi-illumination images for 12 flash lights around the cameras, one light at a time. From the multi-flash images we recover a per pixel likelihood of depth edges in the scene and a label of pixels with a large appearance variation under changing illumination – relating to the specularity. We detail the design of our rig and the capture processes in appendix B. Figure 4.2 shows a snapshot of the data captured, fig. 4.7 illustrates our camera rig prototype.

We elected to calculate depth from stereo because it performs better than the following three alternatives we tested.

- Recovering geometry from intrinsic-image-decomposition ([118]) and photometric stereo with a few lights ([6]) did not yield satisfactory results. PIE-Net([118]) requires 256×256 images which were too low-resolution for reconstruction and, our captures were out-of-distribution for the pre-trained model. [6] assumes fixed lights – we'd need new light calibrations per-view.
- Self-calibrating-photometric-stereo ([194, 195]), demonstrated on [196], needs 50-80 light views and accurate masks which we do not capture. Also, our lights are much closer to the camera than [196, 197].
- modern camera-projector systems ([198, 199]) yield better estimates of geometry than stereo, but is not fast enough to capture human subjects (fig. 4.10).

4.4.2 Captured dataset

Although a data set is not the primary contribution of our research, we capture some salient aspects of the scene that are not present in several established datasets. We identify these aspects in table 4.1.

In the rows labeled "specularity" and "depth edges" we note if the dataset has explicit labels for the specular nature of the pixel or a presence of a depth edge at that pixel respectively. Under

Property	BMVS	DTU	ReNe	DGT*	PaNDoRa	OpenIllum.	Ours
depth	✓	✓	×	✓	×	×	✓
OLAT/ Flash	OLAT	×	OLAT	OLAT	×	OLAT	Flash
polarization	×	×	×	×	✓	✓	×
specularity	×	×	×	×	✓	×	✓
depth edges	×	×	×	×	×	×	✓
HDR	✓	×	×	✓	×	✓	✓
illum. model	✓	×	×	✓	✓	✓	×

Table 4.1: We identify some differences between our dataset and a few established datasets: BMVS[200], DTU[187], ReNe[201], DiLiGenT[196], DiLiGenT-MV[196] (both abbreviated as DGT*), PaNDoRa[123] and Open-Illumination[202]. OLAT: one light at a time.

Scene	NRGBD	BF	AdaShell^{++}	NeUS^{++}	VolSDF^{++}
greenroom	**0.013**	0.024	0.015	0.016	0.014
staircase	0.045	0.091	0.024	**0.009**	0.020
kitchen I	0.252	0.234	0.044	**0.036**	0.047
kitchen II	**0.032**	0.089	0.045	**0.032**	0.060

Table 4.2: Accuracy of reconstruction from un-posed RGBD images. For ***un-posed*** RGBD images, we compare the accuracy of scene reconstruction using AdaShell^{++}, NeUS^{++}, and VolSDF^{++}with NeuralRGBD (NRGBD) [162] and BundleFusion (BF) [156]. We report normalized Chamfer distances (lower is better) across four synthetic scenes from [162].

"illum. model" we note if an explicit illumination model is present per scene – we do not capture an environment illumination model, and instead provide light poses. PaNDoRa does not have explicit specularity labels but polarization measurements at pixels may be used to derive high quality specularity labels, which are better than what our system natively captures. We differentiate between "OLAT" (one light at a time) and "flash" by the location of the source of illumination. Similar to WildLight [83], our flashes are parallel to the imaging plane, located close (∼0.1f) to the camera, as opposed to ReNE and OpenIllumination.

4.5 Experiments and results

4.5.1 Accuracy of incorporating metric depth

Scene	`Rm 0`	`Rm 1`	`Rm 2`	`Off 0`	`Off 1`	`Off 2`	`Off 3`	`Off 4`
PointSLAM	0.61	0.41	0.37	0.38	0.48	0.54	0.69	0.72
AdaShell^{++}	**0.11**	**0.10**	**0.09**	**0.12**	**0.06**	**0.08**	**0.14**	**0.11**

Table 4.3: Accuracy of reconstruction from posed RGBD images. For RGBD images with ***ground-truth poses***, we compare the accuracy of reconstructing the scene between AdaShell^{++}and PointSLAM[163]. We report the mean L_1 distances in cm (lower is better) across eight synthetic scenes from the Replica Dataset [203]. For posed images, we report an average improvement of 81% over PointSLAM.

We reconstruct synthetic scenes with ground truth depth from [162, 163] to measure the accuracy of our technique. We use 12-15 RGBD images to reconstruct the scenes and train for an average of 30k gradient steps (∼1500 epochs) in about 75 minutes. In contrast, [162, 163] use 300+ RGBD tuples and 9+ hours of training on comparable hardware. Notably, [162] also optimizes for noise in camera poses and reports metrics with ground truth and optimized poses. We report the best metric among these two. [156] registers the images themselves. We register the RGBD images with

Scene	stereo	VolSDF^{++}	NeUS^{++}	AdaShell^{++}	UniSurf^{++}
horizontal (fig. 4.4a)	6.22	4.74	**2.77**	5.42	13.21
vertical (fig. 4.4b)	6.82	3.87	**3.68**	6.34	16.02

Table 4.4: Depth edges help prioritize learning of texture discontinuities over geometric ones. We report the RMS deviation from a plane (lower is better) for the reconstructed checkerboard surfaces in mm. We note that AdaShell^{++}performs slightly worse than volumetric methods VolSDF^{++}and NeUS^{++}. Except for UniSurf^{++}, all improve the quality of the surface measured with only stereo. Qualitative results are shown in fig. 4.4.

a combination of rigid and photometric registration ([204, 205, 206]). We present the quantitative results in tables 4.2 and 4.3. We replicate or out-perform the baselines by using a fraction of the training data and gradient steps. Among all methods discussed in section 4.3.5, AdaShell^{++}and VolSDF^{++}demonstrate similar performance, NeUS^{++}recovers a smoother surface at the expense of $\sim 1.25\times$ more gradient steps. Our errors on these synthetic datasets closely reflect the performance of [183] on approximating surfaces from low noise point clouds. These datasets do not have large view dependent appearance variations to affect the gradient updates.

4.5.2 The effect of depth edges in training

We tested fused RGBD maps from stereo and four baselines from section 4.3.5 to investigate the effect of depth and texture edges. We use edge guided sampling (section 4.3.3 and eq. (4.6)) for all except stereo and UniSurf^{++}to prioritize learning geometric discontinuities over appearance. We present the results in fig. 4.4 and table 4.4. All of the baselines except UniSurf^{++}improve the reconstruction accuracy due to segregation of texture and depth edges. The smoothness enforced by the curvature loss in eq. (4.4) also improves the surface reconstruction over stereo.

4.5.3 View synthesis with dense depth

Metric	VolSDF^{++}			NeUS^{++}			AdaShell^{++}		
27.5+	21.3	33.1	40.0	70.5	100+	100+	22.6	23.2	94.6
100K	30.28	30.33	30.69	27.82	29.45	25.31	31.56	31.45	28.27

Table 4.5: Training performance for view synthesis. We report two metrics - number of steps required to reach or exceed a PSNR of 27.5 and PSNR at the end of 100K gradient steps. We observe that all the baselines perform competitively and ignoring depth and additional supervision signals (last two columns) leads to failures in the view synthesis tasks.

Incorporating dense metric depth and our sampling strategy from sections 4.3.2 and 4.3.3 enables AdaShell^{++}, VolSDF^{++}, and NeUS^{++}to perform competitively across challenging scenes. Scene A (fig. 4.8(a)) looks at a couple of reflective objects with large variation in view dependent appearance. Additionally, there are large local errors in the captured depth maps due to specularities in the scene. We capture six stereo pairs, train on 11 images and test on one image. Scene B (fig. 4.8(b)) features a rough metallic object of relatively simple geometry captured by a 16mm lens (450 mm focal length, shallow depth of field). We capture four stereo pairs, train on seven images and test on one image. Scene C (fig. 4.8(c)) features a fairly complicated geometry and is captured with 12 stereo pairs. We train on 22 images and test on two. Quantitative results of our experiments are in table 4.5. We observe that AdaShell^{++}, which is roughly 15% faster per gradient step than VolSDF^{++}, generally converges the fastest (wall clock time) to a target PSNR. When the geometry is very complicated (scene C), an equally complicated sampling volume negates the efficiency gains of our sampler. We could not find good parameters for UniSurf^{++}for any of these sequences.

View synthesis was unsuccessful without the inclusion of dense depth. We trained VolSDF^{++}with

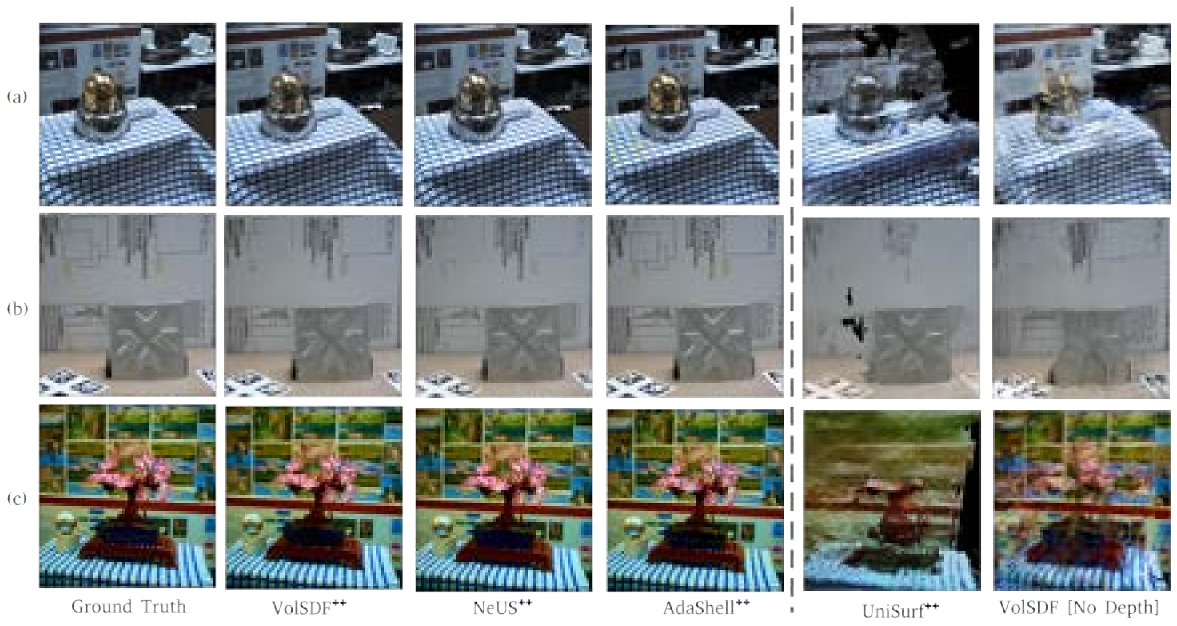


Figure 4.8: Relative performance of all the methods. Details in section 4.5.3, quantitative results in table 4.5. Each method was allocated a budget of 100,000 gradient steps or fewer, and the enhancement in reconstruction exhibited a monotonically increasing trend until (and beyond) the cutoff.

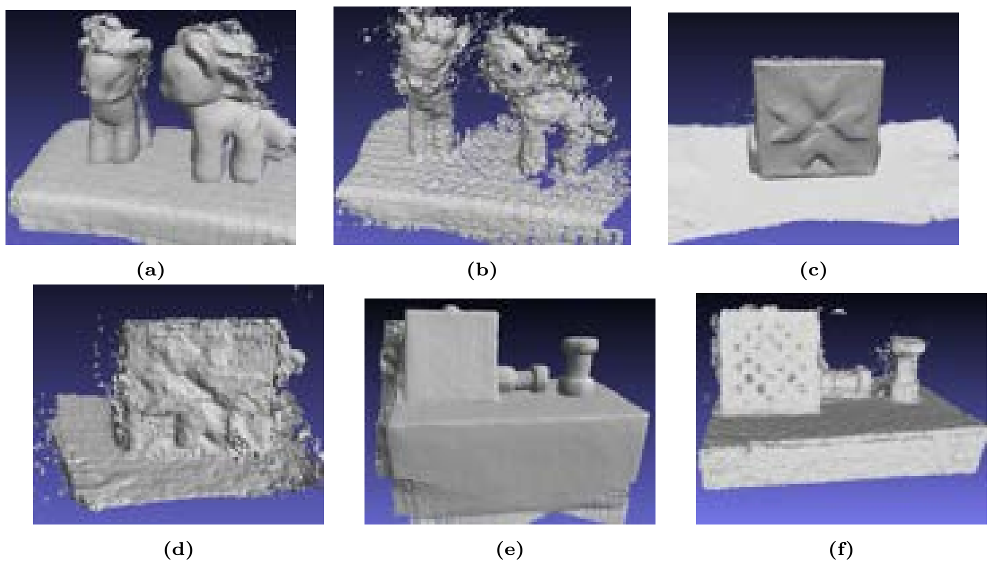


Figure 4.9: AdaShell^{++}can work with noisy depth with little loss in view interpolation performance and training time. However, the surface recovered is also noisy. Figures 4.9a, 4.9c and 4.9e are the geometries recovered with no noise in depth and figs. 4.9b, 4.9d and 4.9f are with noisy depths. Details in section 4.5.4.

no depth supervision (equivalent to [157]) until saturation (less than 0.1 PSNR increase for 1000 consecutive epochs). The reconstructions, none of which had a PSNR of 18 or higher, are shown in the last column of fig. 4.8.

4.5.4 Using noisy depth

To investigate the effects of noise in the depth maps, we obtain the depths of scenes using conventional stereo. We used semi-global matching stereo ([178]) with a dense census cost ([207]) and sub-pixel refinement on tone mapped HDR images to calculate the surface depth. Surface normals were calculated using the spatial gradients of the depth maps. To focus on the performance of our approaches, we did not filter or smooth the depth obtained from conventional stereo. From the top of table 4.6, we observe that NeUS^{++}strictly improves the quality of the surface reconstructed from just noisy stereo (row 1 and 2 versus row 3), especially when edge sampling is enabled. If the end goal is just view synthesis, AdaShell^{++}, which blends the advantages of volumetric and surface based rendering, performs equally well with large noise in depth, whereas NeUS^{++}takes many more

condition	fig. 4.9(a,b)[5]	fig. 4.9(c,d)[7]	fig. 4.9(e,f)[5]
edge sampling	**491**	**403**	**225**
no edge sampling	593	419	251
noisy stereo	600	523	369
27.5+ PSNR w/ noise(AdaShell^{++})	7.93	20.2	5.12
27.5+ PSNR w/o noise (AdaShell^{++})	7.85	23.2	2.71
25.0+ PSNR w/ noise (NeUS^{++})	49.4	36.0	32.9

Table 4.6: Effect of noisy depth and depth edges. Top: The surface reconstruction quality (Hausdorff distance, lower is better) with conventional (noisy) stereo compared with surface recovered by NeUS^{++}on learned stereo. Qualitative comparison in fig. 4.9(b,d,f). Bottom: gradient steps (in 1000s lower is faster) required to surpass a test time accuracy of 27.5dB with AdaShell^{++}. We specify the count of training views in [ ] braces.

Scene	face_1	face_2	face_3	face_4	shiny_1	shiny_2
mask	**28.95**	**31.19**	**29.56**	**27.28**	**25.18**	**24.51**
no mask	27.17	30.05	27.82	25.46	23.65	21.88

Table 4.7: Relighting scenes with a volumetric renderer. We report PSNR (higher is better) under two heads – masked and unmasked relit images, to offset the effects of incorrect shadows cast on the background. The unmasked reconstructions generally have a poorer PSNR because our implicit scene understanding approach does not approximate a ray tracer and cannot cast correct shadows on the background. The lower PSNR for reconstructing the shiny objects is mainly due to the inability of the network to model saturation caused by reflection. Results in fig. 4.10.

iterations to converge. This indicates that photorealistic view synthesis with a volumetric renderer is possible with noisy depth data. However conventional stereo often introduces large local errors (see e.g. surface patterns on fig. 4.9(b,f)) which our approaches were unable to improve significantly.

In the presence of noisy depth, the quality of the reconstructed surface was enhanced through edge-based sampling (section 4.3.3 and eq. (4.6)). Our sampling strategy allocated samples away from depth edges, where the noise was more prevalent, leading to fewer gradient steps spent modelling areas with higher noise. Table 4.6 presents the quantitative details of the experiment.

4.5.5 Relighting

We capture multi-illumination images with known light poses and recover geometry independently of appearance. This allows us to infer the illumination dependent appearance using a combination of physically based appearance parameters – e.g. the Disney Principled BRDF[208]. As a benchmark, we upgraded the closest related work, [83], which uses the full gamut of the Disney BRDF parameters, with NeUS^{++}, to incorporate dense depth. For the data we collected, the optimization process as implemented by [83], was quite brittle and some parameters (e.g. 'clearcoat-gloss') would often take precedence over other appearance parameters (e.g. 'specular-tint') and drive the optimization to a poor local minima. Figure 4.14 demonstrates this problem with textured meshes. We found the optimization of a subset of appearance parameters ('base-color', 'specular-tint', and 'roughness') to be the most stable. [174, 191] conclude the same.

For relighting, we explore two avenues – the inference step of our approach as a volumetric renderer and a mesh created with the appearance parameters as texture. Quantitative and qualitative results of the volumetric renderer are shown in table 4.7 and fig. 4.10. We used [127] to unwrap the geometry and generate texture coordinates whose quality exceeded [209] and our implementation of [210]. None of our approaches worked on the ReNe dataset (table 4.1) due to low view diversity, and the absence of metric depth. We used the labels in fig. 4.2e to allocate more gradient steps in learning the regions with higher appearance variation.

4.6 Limitations

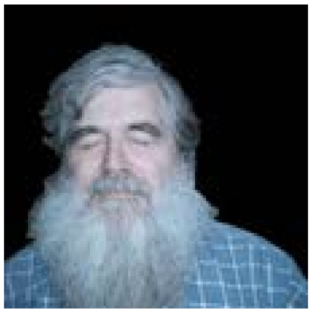
Although we achieve state of the art results in view-synthesis and relighting with a few views, our approach struggles to represent transparent objects and accurately capture the geometry of reflective surfaces. [211] address the problem of reflective objects by modelling background reflections and is based on the architecture proposed by [158]. As NeUS^{++}enables [158] to use possibly noisy metric depth, it can potentially be extended to model reflective objects.

Our approaches require metric depth and depth edges for the best performance. Our approach relies on capture devices with reasonable quality depth measurements. Future work will address incorporation of monocular and sparse depth priors with depth edges.

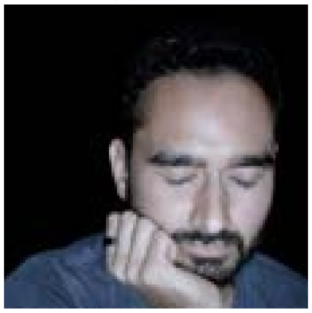
Incorporation of metric depth introduces a strong bias, often limiting super resolution of geometry sometimes achieved in neural 3D scene representation (see e.g. [159]). Decreasing the effect of eq. (4.5) during training may potentially encourage geometric superresolution and is future work.

(a) `face_1`

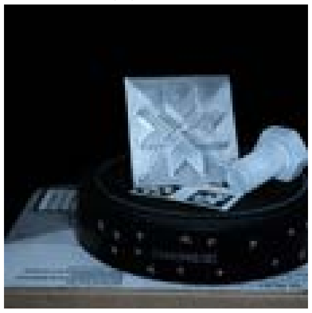
(b) `face_2`



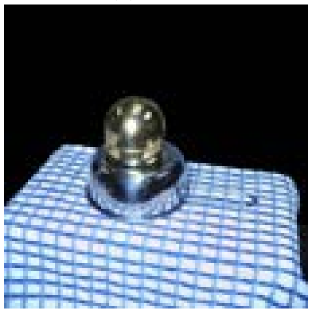
(c) `face_3`



(d) `face_4`



(e) `shiny_1`



(f) `shiny_2`

Figure 4.10: Relighting scenes can be achieved with AdaShell^{++}trained with multi-illumination images. Discussion and results in section 4.5.5 and table 4.7. We capture 12 flash lit images for all the camera views and we use alternate flashes for all the training views (6 per view). Figures above show one (of six) flash configurations for the test view. More results on the project website.

Finally, modern grid based representations (see e.g. [212, 213]) produce very compelling view interpolation results at a fraction of the computational cost of a state of the art volumetric renderer (e.g. [137, 186]). However, they need to be "distilled" from a pre-trained volumetric view interpolator. Future work can investigate the use of depth priors to train a grid based representation directly from color and depth images.

4.7 Conclusions

We present a solution to incorporate dense metric depth into neural 3D reconstruction which enables state of the art geometry reconstruction. With dense metric depth available, we demonstrate an a to accelerate training of neural representations for view synthesis and relighting of small scenes. We examine a corner case of jointly learning appearance and geometry and address it by incorporating additional supervision signals. Additionally, we describe a variant of the multi-flash camera to capture the salient supervision signals needed to improve photorealistic 3D reconstruction.

4.8 Retrospectives

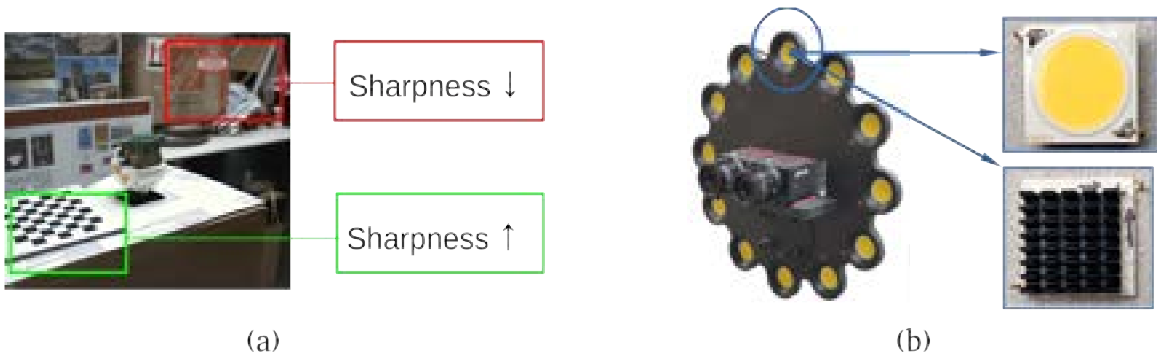


Figure 4.11: Hardware limitations: Inability to model bokeh (a) and open-loop coupling between flashes and cameras (b) are the two main limitations of the hardware we used to capture the data for the work.

The bulk of this research was done during December 2022 through December 2023, with writing and additional results being added through May 2024. This period was also marked by a significant flux in the state of the art in neural rendering and view synthesis. We wanted to highlight the following in that context:

Illumination setup: Aiming at a flexible near-field and environment illumination system, we initially we proposed a system as a combination of the illuminated workspace described in part III with the multi-flash camera described in section 4.4.1. Later in May 2023, RelightMyNerRF ([201]) was published, describing a flexible system consisting of a pair of robot manipulators – one controlling the position of a camera, the other controlling the position of the light. Our experiments with data from [201] and our initial illumination system (consisting of both near-field and environment illumination) indicated that inferring shape of objects from environmental illumination yielded much poorer results than modern stereo ([171]), and neural back-ends struggled at handling high contrast attached shadows.

Gaussian Splatting based methods ([214]) for high resolution view synthesis were introduced in August 2023. This was at a point when we had already made significant progress in view synthesis and relighting using neural back-ends. Given the view interpolation quality of earlier (August 2023 – May 2024) Gaussian Splatting based representations were lower than neural volumetric rendering, we did not switch to a Gaussian based representation. Methods for ray-tracing Gaussians ([215])

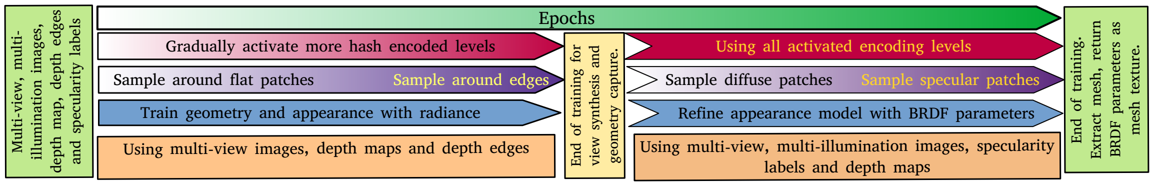

Multi-view, multi-illumination images, depth map, depth edges and specularity labels
Epochs
Gradually activate more hash encoded levels
Sample around flat patches
Sample around edges
Train geometry and appearance with radiance
Using multi-view images, depth maps and depth edges
End of training for view synthesis and geometry capture.
Using all activated encoding levels
Sample diffuse patches
Sample specular patches
Refine appearance model with BRDF parameters
Using multi-view, multi-illumination images, specularity labels and depth maps
End of training. Extract mesh, return BRDF parameters as mesh texture.

Figure 4.12: Our approach to recovering 3D assets from captured data. In the first part, for NeUS^{++}and VolSDF^{++}we jointly optimize geometry and appearance by minimizing eqs. (4.3) and (4.5). For AdaShell^{++}we first optimize eq. (4.5) for a fixed number of gradient steps before the joint optimization. At this stage the geometry is optimized and appearance is recovered as radiance. Following this, we use multi-illumination images with a truncated BRDF parametrization to refine the appearance model, given the geometry, to learn the reflectance parameters.

published recently (October 2024) might be the next stepping stone for physically based relighting of a scene comprised of Gaussians.

Shell based volumetric methods (section 4.3.4) also provide an attractive scene representation alternative for view synthesis. We formulated and implemented the method (section 4.3.4) around October 2023 and noticed significant performance improvements in training and view-interpolation quality. Soon after, a more complete formulation of the idea was published as [186]. In light of this, we renamed our formulation after [186] as AdaShell^{++}. Similar ideas have been implemented with Gaussian splats too (see e.g. [216, 217]).

Hardware: Finally, we used a simplified lens model which was incapable of modelling a complicated bokeh effect which we encountered frequently due to our choice of imaging components (see fig. 4.11). Hua and colleagues ([218]) proposed a solution that may serve as a drop-in replacement in our pipeline to address this drawback. We also did not engineer a system to digitally couple the cameras and flashes, which led to significantly slower capture times and needed manual checks to ensure the quality of synchronization of the lights and the cameras.

4.9 Supplementary Materials

4.9.1 Representations and implementation details

Model details

Our scene representation consists of two networks – an intrinsic network $\mathcal{N}(\theta)$ and an appearance network $\mathcal{A}(\phi)$. We follow [159] to build and train $\mathcal{N}(\theta)$. We use 18 levels of hashgrid encodings [137] to encode the input and a two layer (128 neurons/layer) MLP to generate the intrinsic embedding. The first channel of the embedding, $\mathcal{S}(\theta)$ is trained with eq. (4.5) to recover a signed distance field of the scene as described in section 4.3.2. The rest of the 127 channels of the embedding $\mathcal{E}(\theta)$ are passed on to the appearance network $\mathcal{A}(\phi)$ as an input.

The appearance network takes $\mathcal{E}(\theta)$, the viewing direction (encoded with 6 levels of sinusoidal encodings following [160]), and optionally the illumination direction (if recovering BRDF) to generate colors. The neural network is built with 2 layers of fully connected MLPs (128 neurons/layer) with skip connections.

VolSDF^{++} is our method similar to VolSDF [157] and MonoSDF[141]. We represent the scene with $\mathcal{N}$ and $\mathcal{A}$ and train it with metric depth and color by minimizing eq. (4.4). The samples for eq. (4.3) are drawn using the "error-bounded sampler" introduced by [157].

NeUS^{++} represents a modified version of NeUS [158], where we use the training schedule and structure of $\mathcal{N}$ from [159], the appearance network $\mathcal{A}$ is adopted from NeUS and we optimize eq. (4.5) along with eq. (4.3). In addition to $\mathcal{A}$, NeUS^{++}also has a small 4 layer MLP (32 neurons per layer) to learn the radiance of the background as recommended in the original work by [158].

UniSurf^{++} is our method inspired by UniSurf[181]. We represent the scene's geometry using a pre-optimized implicit network $\mathcal{N}$ as outlined in section 4.3.2. We follow the recommendations of [181] to optimize $\mathcal{A}$. UniSurf exposes a hyperparameter to bias sampling of eq. (4.2) towards the current estimate of the surface. As we pre-optimize the surface, we can find the surface point $\mathbf{x}_s = \mathbf{o} + t_s\mathbf{d}$ through sphere tracing $\mathcal{S}$ along a ray. The intersection point t_s can then be used to generate N samples along the ray to optimize eq. (4.3).

$$t_i = \mathcal{U}\left[t_s + \left(\frac{2i-2}{N} - 1\right)\Delta, t_s + \left(\frac{2i}{N} - 1\right)\Delta\right] \tag{4.7}$$

Equation (4.7) is the distribution used to draw samples and Δ is the hyperparameter that biases the samples to be close to the current surface estimate. We optimize $\mathcal{S}$ independent of eq. (4.2) by just minimizing eq. (4.5) with registered depth maps (see section 4.3.2). We use this method to study the effects of volumetric rendering versus surface rendering. We found this strategy to be very sensitive to the hyperparameter Δ and its decay schedule as the training progressed. While best parameters for some sequences resulted in very quick convergence, they were very hard to come across and generally, poorer choices led to undesirable artifacts (see e.g. fig. 4.4).

Finally, **AdaShell^{++}** is our method inspired by AdaptiveShells[186] and [137, 161]. We warm start with a pre-optimized $\mathcal{S}$ by minimizing eq. (4.5) in section 4.3.2. We then enclose the surface represented by $\mathcal{S}$ in an isotropic voxel grid and progressively cull the voxels at a pre-set distance from the zero-level set of $\mathcal{S}$. This leaves us with a "shell" of voxels around (both inside and outside) the surface (zero-level set of $\mathcal{S}$) of the object which serves a similar purpose to the "shell" around the learned surface in [186]. We roughly follow [161] to generate samples along a segment of the ray guided by the voxels it intersects. The spatial density of samples is inversely proportional to their distance from the estimated surface. We implement this using the tools from NerfStudio[134, 219]. Figure 4.5(a,d). Figure 4.5(b,e) denote the sampling volume as a wire frame around the estimated geometry (fig. 4.5(a,d)) and fig. 4.5(c,f) are zoomed in views of sections of the scene.

The "shells" shown in fig. 4.5 and the shells recovered by [186] for small scenes are physically similar quantities. [186] dilate and erode the original level-set of the scene (approximated by $\mathcal{S}$) using a hyperparameter. Our "shells" are jointly estimated with the geometry as the training progresses. [186] estimate the fall-off of the volume density values along a ray to determine the hyperparameters, which in turn determines the thickness of the "shell". They subsequently use uniform sampling (similar to eq. (4.7), where the Δ now denotes the local thickness of the shell) to generate samples for rendering. Our work takes a discrete approach by immersing the zero-level set (in form of pre-optimized $\mathcal{S}$) in a dense isotropic voxel grid and culling the voxels which have a lower volume density, according to a preset hyperparameter that determines the thickness of the shell. Once the shell has been estimated, we use a unbiased density weighted sampler (instead of a uniform sampler) to generate samples along the ray inside the shell. We expect our sampling strategy to be more robust to errors in estimated geometry (as shown in section 4.5.4) than [186]. However, at the time of writing, an implementation of AdaptiveShells was not available to validate this claim.

Training Details

We ran our experiments on a Linux workstation with an Intel Core i9 processor, 64GB RAM, and an Nvidia RTX3090Ti graphics card with 25GB of vRAM. Across all the experiments for learning scene radiance, we implemented a hard cut-off of 100K gradient steps amounting to less than 4.5 hours of training time across all the experiments. The implementations of our baselines, design of the multi-flash camera system, dataset and the hyperparameters will be released in future. Figure 4.5 denotes the training steps graphically.

Difference between our and prior work on neural scene understanding with depth

IGR[183] were among the first to fit a neural surface to point samples of the surface. Our pipeline is largely inspired by that work. However, we have two main differences – we use a smaller network, and periodically activate multi-resolution hash encodings as recommended by [159] instead of using a fully connected set of layers with skip connections. Additionally, as we have access to depth maps, we identify the variance of the neighborhood of a point on the surface through a sliding window filter. We use this local estimate of variance in a normal distribution to draw samples for $\mathbf{x}_s^\Delta$ along each ray. Our strategy assumes that image-space pixel neighbors are also world space neighbors, which is incorrect along the depth edges. However, as the Eikonal equation should be generally valid in $\mathbb{R}^3$ for $\mathcal{S}$, the incorrect samples do not cause substantial errors and only contribute as minor inefficiencies in the pipeline. A more physically based alternative, following [183], would be executing nearest neighbor queries at each surface point along the rays to estimate the variance for sampling. With about 80k rays per batch, $\sim$ 200K points in $(\mathbf{x}_s)$, and about 40k gradient steps executed till convergence, and a smaller network, our approach was more than two orders of magnitude faster than [183], with no measurable decrease in accuracy of approximating the zero-level set of the surface.

NeuralRGBD[162] is the closest prior work based on data needed for the pipeline and its output. The scene is reconstructed using color and aligned dense metric depth maps. The authors aggregate the depth maps as signed distance fields and use the signed distance field to calculate weights for cumulative radiance along samples on a ray (eq. (4.2) in text). The weights are calculated with

$$w_i = \sigma\left(\frac{D_i}{tr}\right) \times \sigma\left(-\frac{D_i}{tr}\right) \tag{4.8}$$

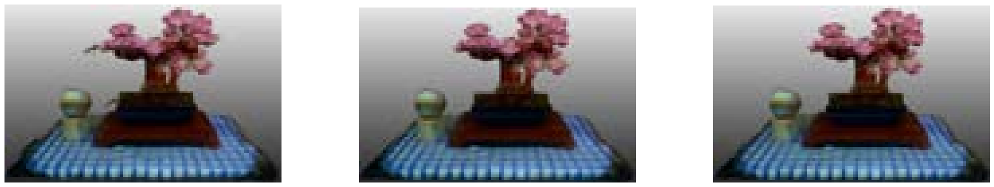
where the D_i is the distance to the surface point along a ray, and the truncation tr denotes how fast the weights fall off away from the surface. Equation (4.8) yields surface biased weights with a variance controlled by the parameter tr. Notably, the depth map aggregation does not yield a learned sign distance field (no Eikonal regularizer in the loss). The authors also include a 'free-space' preserving loss to remove "floaters". As implemented, the pipeline needs the truncation factor to be selected per-scene. As the depth maps are implicitly averaged by a neural network, it is implicitly smoothed and therefore the pipeline is robust to local noise in the depth map.

MonoSDF[141] is mathematically the closest prior method to our work and it uses dense scene depths and normals obtained by a monocular depth and normal prediction network (OmniData[142]). MonoSDF defines the ray length weighted with the scene density as the scene depth $\mathbf{d}_{pred}$ and minimizes

$$\ell_D = \sum_r ||\mathbf{w}\mathbf{d}_{mono} + \mathbf{q} - \mathbf{d}_{pred}||_2^2 \tag{4.9}$$

where $\{\mathbf{w}, \mathbf{q}\}$ are scale and shift parameters. Estimating an affine transformation on the monocular depth $\mathbf{d}_{mono}$ is important because in addition to gauge freedom $(\mathbf{w})$, monocular depths also have an affine degree of freedom $(\mathbf{q})$. The scale and shift can be solved using least squares to align $\mathbf{d}_{mono}$ and $\mathbf{d}_{pred}$. The scene normals are calculated as gradients of $\mathcal{S}$ weighted with scene density along a ray. Through a scale and shift invariant loss, MonoSDF calculates one set of $(\mathbf{w}, \mathbf{q})$ for all the rays in the batch corresponding to a single training RGBD tuple. In the earlier stages of the training, this loss helps the scene geometry converge. The underlying assumption is that there is an unique tuple $\{\mathbf{w}, \mathbf{q}\}$ per training image that aligns $\mathbf{d}_{mono}$ to the actual scene depth captured by the intrinsic network $\mathcal{N}$.

Our experiments with MonoSDF indicate that the network probably memorizes the set of $(\mathbf{w}, \mathbf{q})$ tuples per training image. Explicitly passing an unique scalar tied to the training image (e.g. image index as proposed in [220]) speeds up convergence significantly. Success of MonoSDF in recovering both shape and appearance strongly depends on the quality of the monocular depth and normal predictions. Our experiments on using MonoSDF on the WildLight dataset([83]) or the ReNe dataset ([201]) failed because the pre-trained Omnidata models performed poorly on these datasets. Unfortunately, as implemented, MonoSDF also failed to reconstruct scene geometry when the angles between the training views were small – ReNe dataset views are maximally 45° apart. However,



(a) AdaShell^{++} **(b)** VolSDF^{++} **(c)** NeUS^{++}

Figure 4.13: All the pipelines can be used to extract the "base-color" of the scene. We calculate texture of the meshes from the radiance at convergence (PSNR 27.5+) for one of the scenes in fig. 4.1. The textured meshes are rendered with MeshLab[222].

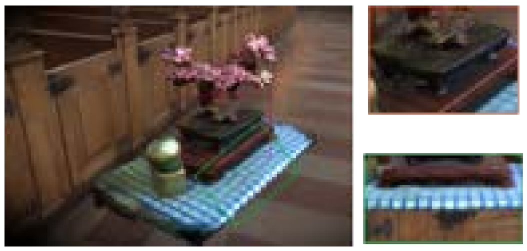
it demonstrates superior performance on the DTU and the BlendedMVS sequences while training with as low as three pre-selected views. Finally, our scenes were captured with a small depth of field and most of the background was out of focus, so the scene background depth was significantly more noisy than the foreground depth. We sidestepped this problem by assigning a fixed 1m depth to all the pixels that were in the background. Although this depth mask simplifies our camera pose estimation problem (by segregating the foreground from the background), it assigns multiple infeasible depths to a single background point. As we aggregate the depth maps into the intrinsic network ($\mathcal{N}$) by minimizing eq. (4.5), the network learns the mean (with some local smoothing) of the multiple depths assigned to the single background point. However, the scale and shift invariant loss is not robust to this and with masked depth maps, we could not reliably optimize MonoSDF on our sequences. We suspect that this is because the scale and shift estimates for each instance of eq. (4.9) on the background points yielded very different results, de-stabilizing the optimization.

[**140**] and [**221**] use sparse scene depth in the form of SfM triangulated points. [140] use learnt spatial propagation [143] to generate dense depth maps from the sparse depth obtained by projecting the world points triangulated by SfM. [221] assign the closest surface depth at a pixel obtained by projecting the triangulated points to the image plane. Neither of these pipelines recover a 3D representation of the scene and focus on view synthesis using few views.

[**163**] introduce a novel 3D representation – "Neural point clouds" which includes geometric and appearance feature descriptors (small MLPs) grounded to a point in 3D. The geometry is recovered as the anchors of the "neural points". The appearance is calculated using a volumetric renderer which composits the outputs of the appearance descriptors of the neural points with the transmissivity of the neural points along the ray. The transmissivity of a neural point is calculated as a function of distances of a pre-set number of neighboring neural points.

Capturing approximate BRDF and generating textured meshes

Multi-illumination images captured by our camera system can be used to estimate surface reflectance properties. We recover a truncated Disney BRDF model([208]. Our model consists of a per pixel specular albedo, a diffuse RGB albedo, and a roughness value to interpret the observed appearance under varying illumination. To estimate the spatially varying reflectance, we first train a model (AdaShell^{++}, VolSDF^{++}or NeUS^{++}) to convergence to learn the appearance as radiance. At convergence, the first channel $\mathcal{S}$ of the intrinsic network $\mathcal{N}$ encodes the geometry and the appearance network $\mathcal{A}$ encodes the radiance. We use two of the embedding channels of $\mathcal{E}$ to predict the roughness and specular albedo at every point on the scene. The diffuse albedo is obtained as the output of the converged appearance network $\mathcal{A}$. To calculate the appearance, we apply the shading model ([208]) to calculate the color at every sample along a ray and volumetrically composite them using eq. (4.2) to infer appearance as reflectance. Figure 4.12 describes our steps graphically. Optimizing for the full set of the Disney BRDF parameters, following [83] did not work with our aproach as the optimization the optimization often got stuck at local minima. Figure 4.14b shows one instance of optimizing the pipeline of [83], where the strengths of the recovered 'clearcoat' and 'clearcoat-gloss' parameters dominated over the optimization of the other parameters, resulting in a


(a) Limited parametrization

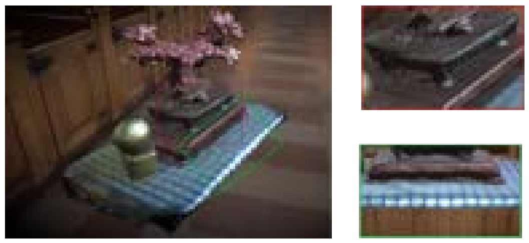

(b) Full Disney BDRF parametrization

Figure 4.14: Optimizing for the full Disney BRDF is difficult. Figure 4.14a shows our results with only specularity, roughness and metallic BRDF parameters. Figure 4.14b depicts identical results utilizing the complete range of Disney BRDF parameters as outlined in [83]. Note the excessive glossy appearance of fig. 4.14b due to the dominance of the clearcoat and clearcoat-gloss parameters. Details in appendix A, meshes rendered with [223].

waxy appearance. Choosing a more conservative set of parameters (only 'base-color', 'specular' and 'roughness') in fig. 4.14a led to a more realistic appearance.

Our process of generating texture and material properties roughly follows the methods described by [83] and [134]. We proceed through the following steps:

1. At convergence (see fig. 4.12), we extracted the scene geometry using the method described in [224].
2. We calculate a depth mask by thresholding the depth images at every training view with an estimate of the scene depth to segregate the foreground from the background.
3. Next, we cull the resulting triangular mesh (step 1) by projecting rays from every unmasked (foreground) pixel corresponding to all the camera views. This lets us extract the main subject of our scene as a mesh. We use Embree[225] to implement this.
4. We generate texture coordinates on the culled mesh using "Smart UV Unwrap" function from [127]. These results were qualitatively better than [209] and our re-implementation of [210]. We then rasterize the culled mesh from step 3 to get points on the surface corresponding to the texture coordinates.
5. We project each of these surface points back on to each of the training views to get the image coordinates. Rays originating from a rasterized surface point and intersecting the surface before reaching the camera are removed to preserve self occlusion.
6. For all the valid projected points, we cast a ray onto the scene and use either AdaShell^{++}, VolSDF^{++}, or NeUS^{++}to generate the color at the pixel along the ray using eq. (4.2). This is repeated for all the training views.
7. At the end of the previous step we have several measurements of colors at every texture coordinate of the scene. We apply a median filter (per color channel) to choose the color – taking averages or maxima of the samples introduces artifacts. If using the radiance as texture is sufficient (often the case for diffuse scenes) this textured mesh can be exported. Figure 4.13 demonstrates using each of AdaShell^{++}, VolSDF^{++}and, NeUS^{++}to calculate the diffuse color of the scene in fig. 4.1.
8. To generate material textures, we follow the same procedures with the corresponding material channels after AdaShell^{++}, VolSDF^{++}or NeUS^{++}has been trained on multi illumination images using the schedule outlined in fig. 4.12.
9. The material properties are also volumetrically composited using eq. (4.2) and median filtered like the base colors. This is different from just querying the value of the network at the estimated surface point in [83].

We use [223], a web browser based tool that supports physically based rendering with the Disney BRDF parameters, and [127] to generate the images in figs. 4.1 and 4.14 respectively.

4.9.2 A multi-flash stereo camera

We capture the scene using a binocular stereo camera pair with a ring of lights that can be flashed at high intensity. For our prototype, we use a pair of machine vision cameras ([95]) with a 1”, 4MP CMOS imaging sensor of resolution of 2048 × 2048 pixels. As we focus mainly on small scenes, we use two sets of lenses that yield a narrow field of view – 12mm and 16mm fixed focal length lens ([96]). We use 80W 5600K white LEDs ([226]) flashed by a high-current DC power supply switched though MOSFETs controlled with an Arduino microcontroller. At each pose of our rig, we captured 12 images with each of the flash lights on (one light at a time) and one HDR image per camera. The cameras are configured to return a 12 bit Bayer image which is then de-Bayered to yield a 16 bit RGB image.

For the HDR images, we performed a sweep of exposures from the sensor’s maximum (22580 microseconds) in 8 stops and used [176] to fuse the exposures captured with ambient illumination (fluorescent light panels in a room). Following the recommendations of [187] we used an f-stop of 2.8 to ensure the whole scene is in the depth of field of the sensors. We found the recommendations from [227] to be incompatible with our pipeline, so we used Reinhard tone-mapping ([177]) to re-interpret the HDR images. Our image localization pipeline, and stereo matching also worked better with tonemapped images.

We set the left and right cameras to be triggered simultaneously by an external synchronization signal. We configured the camera frame acquisition and the illumination control programs to run in the same thread and synchronized the frame acquisition with the flashes through blocking function calls. Figure 4.7 presents a schematic of our prototype device.

Through experiments we observed that the vignetting at the edges of the frames were detrimental to the quality of reconstruction, so we only binned the central 1536 × 1536 pixels. A 16bit 1536×1536 frame saved as a PNG image was often larget than 10MB. To achieve a faster capture and training time without sacrificing the field of view, we down sampled the images to a resolution of 768 × 768 pixels for our experiments. Centered crops of our initial larger frames lead to failures of our pose-estimation pipelines due to the field of view being too narrow(section 4.4.2), so we chose to down sample the images instead. For the images lit by a single LED, we used the camera’s auto exposure function to calculate an admissible exposure for the scene and used 80% of the calculated exposure time for imaging – the built-in auto-exposure algorithm tended to over-expose the images a bit. Estimating the exposure takes about 2 seconds. Once the exposure value is calculated, it is used for all of the 12 flashes for each camera.

Several instances of these RGBD tuples are collected and the colored depth maps are registered in the 3D space in two stages – first coarsely using FGR [205] and then refined by optimizing a pose graph[228]. At the end of this global registration and odometry step, we retain a reprojection error of about 5 - 10 pixels. If the reprojection errors are not addressed, they will cause the final assets to have smudged color textures. To address it, we independently align the color images using image-feature based alignment techniques common in multi-view stereo ([206, 229]), so that a sub 1 pixel mean squared reprojection error is attained. The cameras aligned in the image-space are then robustly transformed to the world space poses using RANSAC[230] with Umeyama-Kabsch’s algorithm[231]. Finally, we mask out the specular parts of the aligned images and use ColorICP [204] to refine the poses. The final refinement step helps remove any small offset in the camera poses introduced by the robust alignment step. A subset of the data collected can be viewed on the project website.

Identifying pixels along depth edges

To identify pixels along depth edges, we follow [192] and derive per-pixel likelihoods of depth edges. Assuming that the flashes are point light sources and the scene is Lambertian, we can model the observed image intensity for the k^{th} light illuminating a point $\mathbf{x}$ with reflectance $\rho(\mathbf{x})$ on the object as

$$\mathbf{I}_k(\mathbf{x}) = \mu_k \rho(\mathbf{x}) \langle \mathbf{l}_k(\mathbf{x}), \mathbf{n}(\mathbf{x}) \rangle \tag{4.10}$$

where μ_k is the intensity of the k^{th} source and $\mathbf{l}_k(\mathbf{x})$ is the normalized light vector at the surface point. $\mathbf{I}_k(\mathbf{x})$ is the image with the ambient component removed. With this, we can calculate a ratio image across all the illumination sources

$$\mathbf{R}(\mathbf{x}) = \frac{\mathbf{I}_k(\mathbf{x})}{\mathbf{I}_{max}(\mathbf{x})} = \frac{\mu_k \langle \mathbf{l}_k(\mathbf{x}), \mathbf{n}(\mathbf{x}) \rangle}{\max_i(\mu_i \langle \mathbf{l}_i(\mathbf{x}), \mathbf{n}(\mathbf{x}) \rangle)} \tag{4.11}$$

It is clear that the ratio image $\mathbf{R}(\mathbf{x})$ of a surface point is exclusively a function of the local geometry. As the light source to camera baselines are much smaller than the camera to scene distance, except for a few detached shadows and inter-reflections, the ratio images (eq. (4.11)) are more sensitive to the variations in geometry than any other parameters. We exploit this effect to look for pixels with largest change in intensity along the direction of the epipolar line between the camera and the light source on the image. This yields a per-light confidence value of whether $\mathbf{x}$ is located on a depth edge or not. Across all 12 illumination sources, we extract the maximum values of the confidences as the depth edge maps. Unlike [192], we use 12 illumination sources 30° apart, and we do not threshold the confidence values to extract a binary edge map. This lets us extract more edges especially for our narrow depth of field imaging system and gets rid of hyper parameters used for thresholding and connecting the edges.

Often parts of our scene violate the assumption of Lambertian reflectances resulting in spurious depth edges. When we use depth edges for sampling, these errors do not affect the accuracy of our pipeline. When using depth edges for enhancing stereo matching (section 4.4.1) we ensure that the stereo pairs do not contain too many of these spurious edge labels to introduce noise in our depth maps.

Identifying patches with non-Lambertian reflectances

We modified the definition of differential images in the context of near-field photometric stereo introduced by [232, 233] to identify non-Lambertian patches. Assuming uniform Lambertian reflectances, eq. (4.10) can be expanded as

$$\mathbf{I}_k(\mathbf{x}) = \mu_k^* \rho(\mathbf{x}) \mathbf{n}(\mathbf{x})^T \frac{\mathbf{s}_k - \mathbf{x}}{|\mathbf{s}_k - \mathbf{x}|^3} \tag{4.12}$$

where $\mathbf{s}_k$ is the location and μ_k^* is the power of the k^{th} light source. We define the differential images as $\mathbf{I}_t = \frac{\partial \mathbf{I}}{\partial \mathbf{s}} \mathbf{s}_t$ where, $\mathbf{s}_t = \frac{\partial \mathbf{s}}{\partial t}$, which when applied to eq. (4.12) can be expanded as

$$\mathbf{I}_t(\mathbf{x}) = \mathbf{I}(\mathbf{x}) \frac{\mathbf{n}^T \mathbf{s}_t}{\mathbf{n}^T(\mathbf{s} - \mathbf{x})} - 3\mathbf{I}(\mathbf{x}) \frac{(\mathbf{s} - \mathbf{x})^T \mathbf{s}_t}{|\mathbf{s} - \mathbf{x}|^2} \tag{4.13}$$

Observing that the light sources move in a circle around the center of projection on the imaging plane, $\mathbf{s}^T \mathbf{s}_t = 0$. Also, the second term of eq. (4.13) is exceedingly small given that the plane spanned by $\mathbf{s}_t$ is parallel to the imaging plane and our choice of lenses limit the field of view of the cameras. The second term is further attenuated by the denominator $|\mathbf{s} - \mathbf{x}|^2$ because the camera-to-light baselines ($\mathbf{s}$) are at least an order of magnitude smaller than the camera to object distance ($\mathbf{x}$). As a result, under isotropic reflectances (Lambertian assumed for this analysis) the differential images $\mathbf{I}_t(\mathbf{x})$ are invariant to circular light motions. Any observed variance therefore can be attributed to the violations of our isotropic BRDF assumptions. We identify specular patches by measuring the variance of this quantity across the 12 instances of the flashlit images.

Although our pipelines for identifying depth edges and patches of varying appearances demonstrate satisfactory qualitative performance, sometimes they yield wrong labels because eqs. (4.11) and (4.13) do not include additional terms for spatially varying BRDFs and interreflections respectively. These errors do not have any significant effect in our reconstruction pipeline as we use this information to generate samples during different phases of training to minimize photometric losses and we do not directly infer shape or reflectances from these steps.

Difference between [192, 234] and our hardware

[192] was the first to propose pairing flashes with cameras and laid the groundwork for identifying depth edges from multi-flash images from a single viewpoint. However, [192] considered a monocular camera and only four flashes along the horizontal and vertical directions of the camera in the demonstrated device. Researchers (see e.g. [6]) have since extended it by placing multiple light sources far apart from a monocular camera and have demonstrated locating depth edges on objects with strictly Lambertian reflectances. In this work, we retain the original light and camera configuration from [192] and increase the number of lights from four to 12.

[234] also investigated a stereo camera in a multi-flash configuration aimed at edge preserving stereo depth maps. They do not extend the application to synthesizing geometry or appearance by capturing and assimilating multiple views of the scene. For obtaining stereo depth maps, we use [171], which performs much better than conventional stereo matching ([178, 207]) largely deployed in off-the shelf systems ([151]).

Both [192, 193] discuss methods to detect specularities (termed “material edges”) through different transforms of the multi-light images. However, we achieve a more continuous circular motion of the lights around the cameras, so we choose to use the photometric invariants described by [232] instead.

V

Conclusion

5 Conclusion

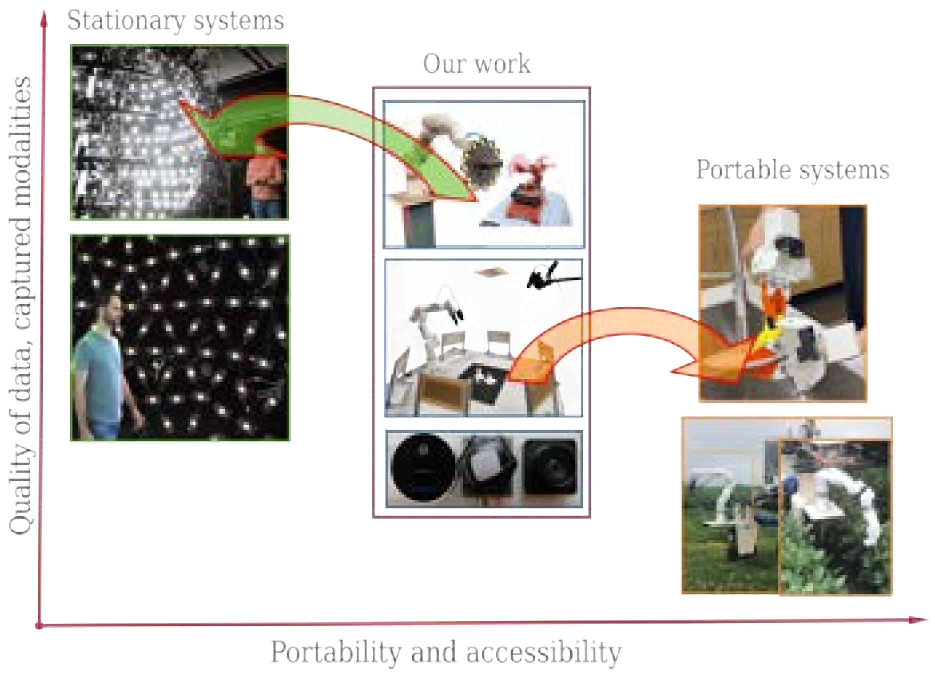

Stationary systems
Our work
Portable systems
Quality of data, captured modalities
Portability and accessibility

Figure 5.1: Looking forward: Improving 3D capture

In this thesis we demonstrated the effectiveness of moving lights and cameras for imaging small scenes. The middle ground we originally proposed was aimed at more accessible systems that capture high quality data from small 3D scenes. To achieve that, we developed three bespoke capturing systems and algorithms. Our systems are more accessible than human capture rigs with hundreds of cameras while capturing high quality and more modes of data than more accessible sensors. However, we still fall short of truly accessible systems e.g. hand-held cameras and modern smartphones. We hope that future work will extend insights from our research to design better hand-held systems for capturing high quality 3D data from small scenes.

As a small step in that direction, in this chapter, we describe two pilot experiments for enhancing small and large capture systems. Particularly, we investigate how we can add back metric depth to portable capture systems and how we may simplify the camera calibration procedures for large multi view stereo capture rigs.

5.1 Improving capture systems

Accessible capture systems ([3, 4, 23]) provide a convenient platform to capture data to understand the 3D world, often sacrificing quality and number of modalities of the data captures. Although the quality of the data captured can be augmented by using more modern cameras with higher quality sensors, capturing additional modalities often involve additional sensors. Incorporation of additional sensors add to the cost of setup, complicate calibration and, eventually lower accessibility or robustness to physical elements (mechanical forces, dust, moisture etc.).

Among the missing modalities, for robotics applications, adding back the capability to measure depth can be potentially most impactful. However, conventional sensors [111, 145, 152]) may not justify their inclusion given the quality of the data captured compared to the complexity added to the system. To mitigate this, researchers have incorporated monocular depth ([235, 236]) as a synthetic modality. Monocular depth is not metric. It has un-resolved scale and shift ambiguities which prevent it from being directly used for robotic manipulation. This is a limitation we would like to address.

For large multi-view imaging rigs ([1, 2]), calibrating cameras is a daunting task. Even for the simplest camera model ($\sim$ 10 parameters) a 100 camera rig can have more than 1000 calibration parameters and simplifying the camera calibration process can be a potentially impactful contribution.

5.1.1 Upgrading monocular depth with metric hints

Modern monocular depth pipelines ([142, 235, 236]) generally yield depth maps that have better quality than depth obtained from current and traditional stereo matching ([171, 178]). We aim to preserve the high quality depth from monocular predictions, while resolving the affine ambiguity associated with them.

In our small scale experiments with monocular depth networks, we observed that the networks have an implicit object-level understanding of all the components of the scene. For a scene with k objects, the network attempts to predict of the order of k monocular depth patches $\mathbf{D}_i^M$, which are then composed together to return the depth maps aligned with the color images. Given state of the art segmentation networks (e.g. [237]), we can extract depths of specific objects in the scene – let all such p, $(p \leq k)$ masks be $\mathbf{M}_i$. We also assume that we have access to a low quality but metric depth map aligned with the color image $\mathbf{D}^T$. The metric depth map may be obtained from stereo ([111, 171]) or time-of-flight sensors ([150, 152]). To upgrade the high-quality but non-metric monocular depth to a high-quality metric depth map, we solve the following set of linear least squared problems for each masked patch $\mathbf{M}_i$:

$$\ell_{D_j} = \sum_{\mathbf{p} \in \mathbf{M}_j} ||\mathbf{w}\mathbf{D}_{\mathbf{p}}^M + \mathbf{q} - \mathbf{D}_{\mathbf{p}}^T||_2^2 \ \forall j \in [1, i] \tag{5.1}$$

where $\mathbf{p}$ denotes a pixel in a mask $\mathbf{M}_j$. The factors $\{\mathbf{w},\ \mathbf{q}\}$ are the scale and shift factors of each patch. [141, 238] demonstrate how eq. (5.1) can be solved in closed form. Traditional image segmentation methods [239, 240] tend to over-segment the images (generally 100+ segments for a 768×768 image), yielding a large number of small masks which generally do not encompass complete objects in the scene. Solving eq. (5.1) with several masks yields several candidate pairs of scales and shifts ($\{\mathbf{w}, \mathbf{q}\}$). A policy for choosing a single scale and shift from the several candidates is not immediately clear, and poorer choices often yield unacceptable results. To address this, we use the state-of-the art segmentation models (e.g. [237]) to obtain masks consistent across whole objects. This simple procedure lets us leverage the merits of monocular depth prediction networks together with implicit understanding of object classes.

5.1.2 Example 1: Super-resolving depth from stereo

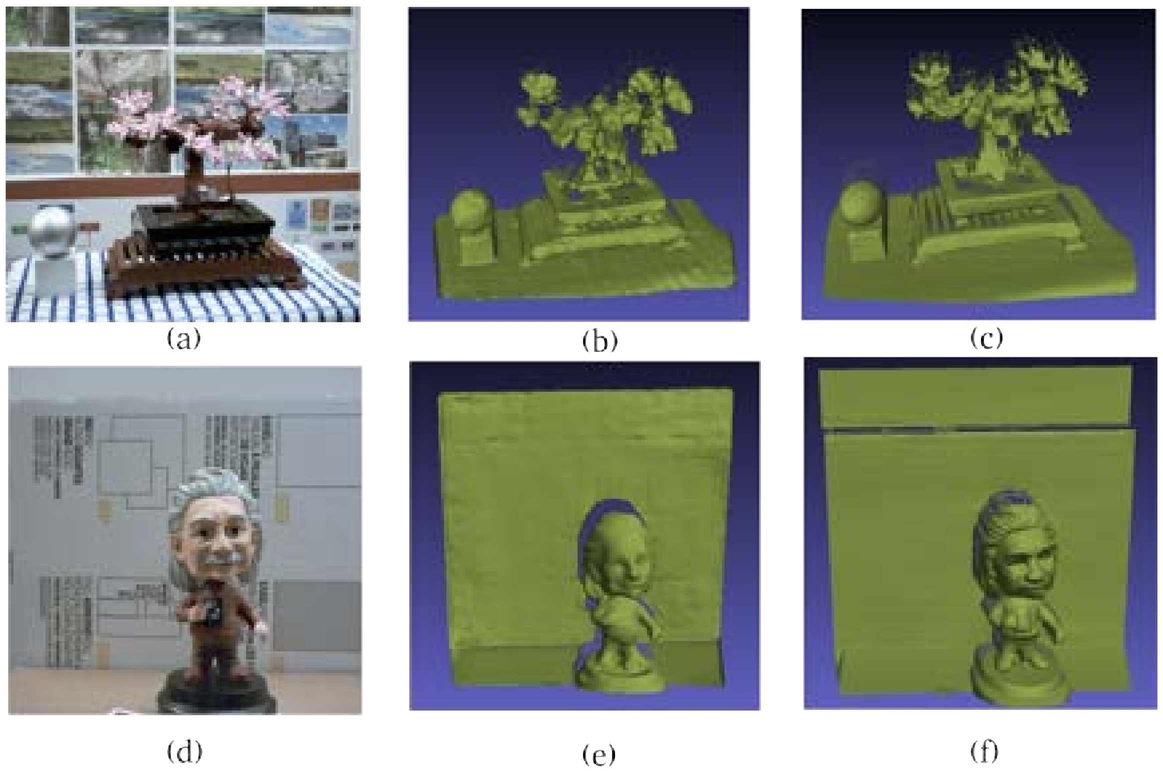


Figure 5.2: Monocular cues can be used to super-resolve depth obtained from stereo. Figure (a,d) are the tonemapped HDR image (left channel) of the sequences. Figures (b,e) are the shaded pointcloud obtained from stereo – we used [171]. We note the absence of sharp details in the depth-map. Figures (c,f) are the upgraded stereo depth map using monocular depth.

Almost all modern monocular depth networks ([142, 235, 236]) preserve exquisite detail along the edges of object surfaces while also preserving local surface smoothness. Although modern stereo matching ([171]) preserves significant amount of detail, qualitatively, it still falls a bit short when compared to monocular depth networks. We show a qualitative example of how the method described in section 5.1.1 can help super-resolve the captured geometry by combining monocular and stereo depths. Figure 5.2 demonstrates this qualitatively.

5.1.3 Example 2: Improving stereo to capture highly non-Lambertian objects

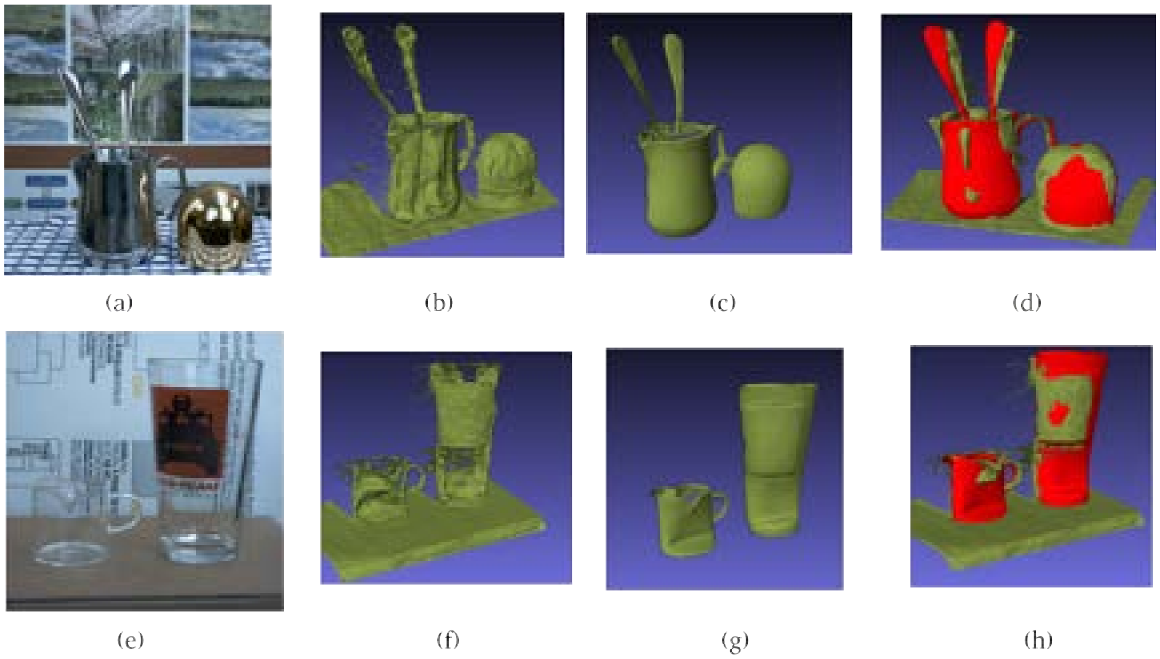


Figure 5.3: Monocular cues can be useful for improving depth perception of transparent and reflective objects. Figures (a,e) are the tonemapped HDR images (left channel) of the sequences, Figures (b,f) is the shaded pointclouds obtained from stereo ([171]). We note the severe artifacts in the geometry reconstructed with stereo. Figures (c,g) are the monocular depths ([236]) obtained from the segmented scene ([237]). Figures (d,h) show the monocular depths (in red) scaled to the metric depth map (in green) using section 5.1.1.

Stereo matching works best when there are minimal view dependent effects between the two viewing directions. Although using HDR images, specularity labels etc. may help (section 4.4.2), stereo matching typically struggles to image reflective, shiny and mostly transparent objects – only capturing accurate depths around parts of the scene with low view-dependent effects. As monocular depth networks have an implicit object-level understanding of the scene, and are not dependent on multi-view consistent appearance, they generally return plausible scene depths irrespective of the objects in the scene. Our method (section 5.1.1) can be used to obtain metrically correct, and more complete depth for scenes with transparent and specular objects by upgrading the incomplete and noisy stereo depths. Figure 5.3 presents a qualitative result.

5.1.4 Example 3: Convenient camera pose estimation for face capture rigs

In this section we take a look at the qualitative performance of two methods for jointly estimating camera poses and scene geometry from images. We base our experiments around the simplest face capture rig – a stereo camera (section 4.4.1) capturing a human face. We look at two methods that recover camera poses and scene geometry from an image pair and compare it against a manually calibrated stereo pair. The results are qualitative and should only be treated as a pilot experiment – our conclusions may not translate to large multi-view rigs.

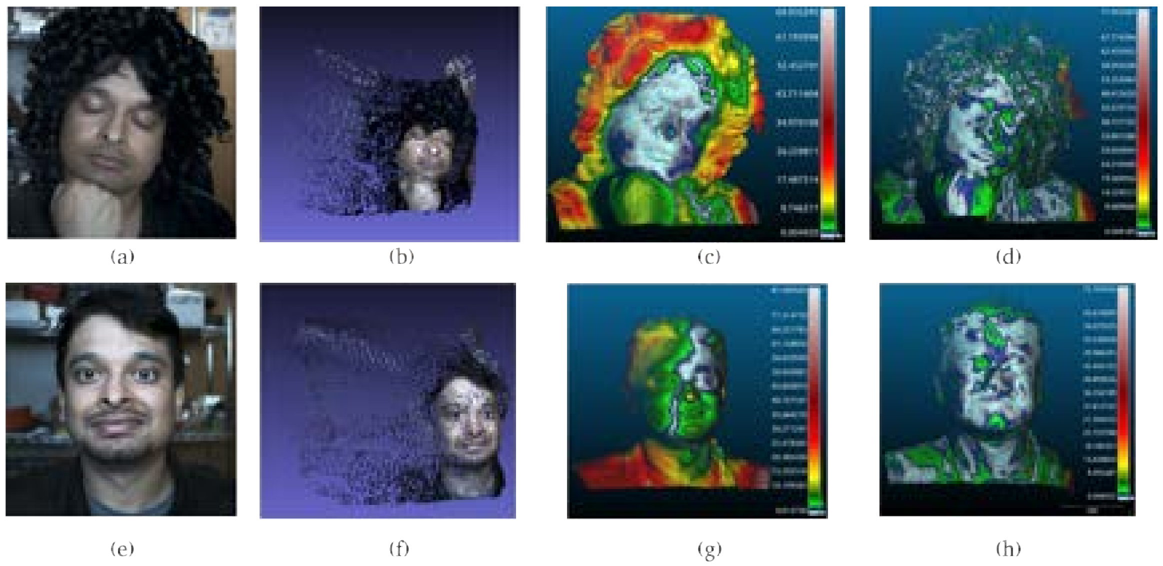


Figure 5.4: Super-resolved, and metrically upgraded depth may help calibrate human capture systems. Figure (a) and (e) are the left channels of the captured frames. Figures (b) and (f) are the geometries obtained by ACE0[241]. The tanh scaled RMSE reprojection errors ([241]) are 0.016 *px* for (b) and 0.017 *px* for (f). Figures (c) and (g) are obtained from our method centered around the Sapiens foundation model([17]). The Hausdorff distance between the two geometries are 9.11 ± 8.48 mm for (c) and 12.49 ± 13.70mm for (g). Finally, figures (d) and (h) are left and right channel depths from calibrated stereo registered using stereo calibration. The Hausdorff distances between the geometries are 2.93 ± 4.69 mm for (d) and 1.81 ± 2.73 mm for (h). We recommend zooming in to the figures for details, the histograms units are equivalent to 0.1mm.

The first method is a state of the art image-to-3D pipeline – ACE0([241]) which solves for 3D camera poses and scene geometry from input images, and pre-calibrated cameras. Qualitatively, we view the reconstructed scene as a point cloud at convergence.

The second method is based around Sapiens ([17]) – a recent foundation model for human capture. We first upgrade the monocular depth of the scene using the technique described in section 5.1.1. We then register the two metric depth maps using facial feature anchor points and robust Procrustes alignment (Umeyama-Kabsch[231] registration and RANSAC[230]). The scene masks, facial feature anchors, and monocular depths are all obtained from Sapiens. We use the pretrained weights with the best validation scores for inference.

Qualitatively, ACE0 performs poorly, even with supplied depths from Sapiens. However, we expect the performance to strongly improve with multiple orthogonal view pairs. As such, 3D from very sparse views (as few as a single stereo pair) is not the ideal use case for ACE0 (or Dust3r([242]) and Mast3r[243]).

Our method based around Sapiens performs much better while still qualitatively falling short of carefully calibrated stereo. From the results of our pilot experiment, we are hopeful that this method, with negligible setup effort, might be able to significantly simplify the multi-camera rig calibration process, while providing results comparable to the current state of the art – carefully calibrated stereo. Figure 5.4 demonstrates the results of our experiment.

Through the pilot experiments described in sections 5.1.2 to 5.1.4, we believe that recent results in 3D computer vision can help simplify and democratize the process of capturing high quality visual data from small scenes.

5.2 Conclusions

In this section we present the conclusions of our research on the systems and science of moving cameras and illumination sources for the better perception of table-top scale scenes.

5.2.1 Relevance of our work in robotics research

The camera and tactile sensor ensemble, described in part II, closely pairs imaging sensors with a tactile sensor. Although the tactile sensor is not *truly collocated* with the cameras as in [244, 245], our sensors capture the objects at a much higher resolution than the sensors described by Yin and colleagues ([245]) and Hogan and colleagues ([244]). This enables geometric correlation between the measurements of the tactile and image sensors, not demonstrated before in the literature. Our insights may be useful for design of visuo-tactile contact policies for robots exploring a workspace, e.g. a humanoid robot interacting with household objects with a palm mounted camera and finger mounted tactile sensors.

The workspace scaled photometric stereo system, described in part III, captures object surface information at a much higher resolution than commercial sensors (see section 3.4.1). The system, primarily designed to image monochromatic Lambertian objects, can be extended to measure objects in a warehouse environment where there is not much variation in the appearance of the objects – e.g. brown-paper boxes, white plastic bags under warehouse illumination. However, the precision of the system comes at a significant trade-off in portability.

The robot mounted multi-flash stereo camera (discussed in part IV) was aimed at separately capturing the geometry and appearance of small scenes, under changing illumination and view-points. The relatively smaller robot-manipulator mounted platform lets us position the sensor somewhat arbitrarily. This can potentially help a user capture scenes challenging for larger capture rigs – e.g. inside or under a kitchen sink.

5.2.2 Frequently asked questions

In this section we address some questions asked during the presentation of the work described in part IV at various venues.

Why did you use stereo instead of other modalities for capturing depth?

Answer: We chose stereo because it generated the highest quality depth maps most conveniently. We address this question in detail in section 4.4.1.

With a robot mounted camera, one should be able to calculate a policy to move the robot to a few locations to capture the most information rich views.

Answer: We looked at it from two angles following Sellán and Jacobson ([246, 247] and Kopanas and Drettakis ([248]).

Sellán and Jacobson show that access to a noise model for the depth sensor can help select the next-best camera pose by optimizing an information gain objective function. We have two kinds of depth estimates at our disposal – disparities calculated by a deep network ([171]) and through conventional block matching ([178]). The noise model for block matching is well understood. Generally, noise for block matching stereo can be described by local Gaussian distributions along surfaces. Block matched disparities also have missing data along geometric edges ruled out by left-right consistency checks. However, the noise models for deep stereo is not well studied – we did not find any literature on this at the time of writing this dissertation. Incorporating depth from SGBM([178]) into our pipeline would have made an automated capture policy possible, however, the depth from SGBM was much lower in quality than deep stereo and would have affected all of the downstream tasks.

Kopanas and Drettakis ([248]) recommend angular uniformity in capturing the images, which we already implement, and is more of a heuristic than an algorithm.

Does your work simulate shadows in a data-driven way?

Answer: Regrettably, we do not handle shadows. Any shadow seen in our demonstrations is purely an artifact of interpolation. One of the reasons for choosing near-field illumination was to avoid large attached shadows cast by objects. Additionally, we captured depth from stereo, which allowed us to very effectively reject the background with the shadows. Although we could recreate the results from Toschi and colleagues ([201]), reasoning about shadows volumetrically, in a purely data driven way, seemed to perform much poorly than ray-tracing meshes. We discuss some more details in section 4.5.5.

For modelling photorealistic appearance, did you explore pre-computed radiance transfer?

Answer: No, we did not. While pre-computed radiance transfer methods ([249, 250]) show promising results, they require significantly more data (and possibly object masks) to work reliably. We show results with a fraction of required data and no additional supervision signals.

What are some immediate improvements one can make to the multi-flash camera system?

Answer: Addition of polarized flashes and a polarizing camera can be the simplest modification that will expand the device's ability to capture a completely new modality of information. We initially looked at this possibility, but until very recently, high resolution polarized cameras were out-of-stock with almost all of the major scientific imaging component vendors.

Given the availability of multiple models to process captured data, how should one select candidates for view synthesis tasks?

Answer: We observed that the inclusion of metric depth into neural view-synthesis pipelines was significantly more effective than other supervision signals. In absence of metric depth, high-quality monocular depth models, free from significant warping, was also effective, and incorporating low-quality monocular depth cues was counter productiv – e.g. incorporating monocular depth cues from OmniData[142] for the sequences captured by Toschi et al.[201] significantly deteriorated the reconstruction quality. Therefore, we recommend incorporating additional models only if they add high quality geometric information to the synthesis task.

What are the main limitations of the multi-flash camera device?

Answer: The multi-flash camera device from part IV has the following limitations:

Resolution: We capture the images at 1536 × 1536px, which is on the lower end of resolution when compared to several established datasets. As we use deep stereo to calculate scene depth, our resolution is limited by the largest effective image the networks can efficiently process (800×800px), and the largest disparity ranges it can reliably match ($\sim$ 350px). Increasing the capture and processing resolution will be definitely helpful.

Portability: We traded off too much portability in favor of captured modalities. With the availability of more capable foundation models for processing visual data, a smartphone camera ensemble can be possibly augmented to closely approximate the capabilities of our device. Discussions in section 5.1 may be a good starting point for this effort.

Camera-flash coupling: Our flashes and cameras are not tightly coupled. Adding hardware synchronization systems between the stereo pairs and the flashes and the cameras can increase the capture speed, and enable the capture of flash-lit images with high dynamic range.

Near-field illumination: To achieve a compact sensor footprint, we elected to have near-field illumination and imaged the scenes with comparatively narrow field-of-view lenses. The near-field illumination complicated the capture process by frequently over-exposing parts of the scene, whereas the narrower field of view images, acquired by a shallower depth-of-field lens introduced a significant amount of "bokeh" (foreground appearing sharper than background) which is not modelled using pin-hole cameras.

I sincerely thank you for your thoughtful readership. Throughout my research and the writing of this dissertation, I've been fortunate to receive support from many individuals. The accomplishments belong to all of us, but any mistakes are entirely my own.

Thank you!

Bibliography

[1] K. Guo, P. Lincoln, P. Davidson, J. Busch, X. Yu, M. Whalen, G. Harvey, S. Orts-Escolano, R. Pandey, J. Dourgarian, *et al.*, "The relightables: Volumetric performance capture of humans with realistic relighting," *ACM Transactions on Graphics (ToG)*, vol. 38, no. 6, pp. 1–19, 2019.

[2] Y. Sheikh and collaborators, "The Mugsy capture system," *Meta Reality Labs Research : Codec Avatars*, 2019.

[3] AAIRA Consortium, "AI Institute for Resilient Agriculture," 2024.

[4] C. Chi, Z. Xu, C. Pan, E. Cousineau, B. Burchfiel, S. Feng, R. Tedrake, and S. Song, "Universal manipulation interface: In-the-wild robot teaching without in-the-wild robots," *arXiv preprint arXiv:2402.10329*, 2024.

[5] A. N. Chaudhury, T. Man, W. Yuan, and C. G. Atkeson, "Using Collocated Vision and Tactile Sensors for Visual Servoing and Localization," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 3427–3434, 2022.

[6] A. N. Chaudhury, L. Keselman, and C. G. Atkeson, "Shape from shading for robotic manipulation," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2024, pp. 8389–8398. [Online]. Available: https://openaccess.thecvf.com/content/WACV2024/html/Chaudhury_Shape_From_Shading_for_Robotic_Manipulation_WACV_2024_paper.html

[7] A. Narayan Chaudhury, I. Vasiljevic, S. Zakharov, V. Guizilini, R. Ambrus, S. Narasimhan, and C. G. Atkeson, "Incorporating dense metric depth into neural 3d representations for view synthesis and relighting," *arXiv e-prints*, pp. arXiv–2409, 2024. [Online]. Available: https://arxiv.org/abs/2409.03061

[8] R. Bajcsy, "Active perception," *Proceedings of the IEEE*, vol. 76, no. 8, pp. 966–1005, 1988.

[9] R. Bajcsy, Y. Aloimonos, and J. K. Tsotsos, "Revisiting active perception," *Autonomous Robots*, vol. 42, no. 2, pp. 177–196, 2018.

[10] Y. Aloimonos, *Active perception.* Psychology Press, 2013.

[11] J. Aloimonos, I. Weiss, and A. Bandyopadhyay, "Active vision," *International Journal of Computer Vision*, vol. 1, no. 4, pp. 333–356, 1988.

[12] P. E. Debevec *et al.*, "The Presedential Portrait," *USC Institute for Creative Technologies*, 2012.

[13] H. Joo, H. Liu, L. Tan, L. Gui, B. Nabbe, I. Matthews, T. Kanade, S. Nobuhara, and Y. Sheikh, "Panoptic Studio: A Massively Multiview System for Social Motion Capture," in *The IEEE International Conference on Computer Vision (ICCV)*, 2015.

[14] H. Joo, T. Simon, X. Li, H. Liu, L. Tan, L. Gui, S. Banerjee, T. S. Godisart, B. Nabbe, I. Matthews, T. Kanade, S. Nobuhara, and Y. Sheikh, "Panoptic Studio: A Massively Multiview System for Social Interaction Capture," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[15] S. Lombardi, T. Simon, G. Schwartz, M. Zollhoefer, Y. Sheikh, and J. Saragih, "Mixture of volumetric primitives for efficient neural rendering," *ACM Transactions on Graphics (ToG)*, vol. 40, no. 4, pp. 1–13, 2021.

[16] S. Saito, G. Schwartz, T. Simon, J. Li, and G. Nam, "Relightable Gaussian Codec Avatars," in *CVPR*, 2024.

[17] R. Khirodkar, T. Bagautdinov, J. Martinez, S. Zhaoen, A. James, P. Selednik, S. Anderson, and S. Saito, "Sapiens: Foundation for human vision models," 2024. [Online]. Available: https://arxiv.org/abs/2408.12569

[18] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song, "Diffusion policy: Visuomotor policy learning via action diffusion," *The International Journal of Robotics Research*, 2024. [Online]. Available: https://diffusion-policy.cs.columbia.edu/

[19] A. Yamaguchi and C. G. Atkeson, "Combining finger vision and optical tactile sensing: Reducing and handling errors while cutting vegetables," in *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*. IEEE, 2016, pp. 1045–1051.

[20] W. Yuan, S. Dong, and E. H. Adelson, "Gelsight: High-resolution robot tactile sensors for estimating geometry and force," *Sensors*, vol. 17, no. 12, p. 2762, 2017.

[21] W. Yuan, M. A. Srinivasan, and E. H. Adelson, "Estimating object hardness with a gelsight touch sensor," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 208–15.

[22] "Detectron: FAIR's platform for Object Detection Research."

[23] S. Song, A. Zeng, J. Lee, and T. Funkhouser, "Grasping in the wild: Learning 6dof closed-loop grasping from low-cost demonstrations," *IEEE Robotics and Automation Letters 2020*, vol. 5, no. 3.

[24] A. Yamaguchi and C. G. Atkeson, "Implementing tactile behaviors using fingervision," in *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*. IEEE, 2017, pp. 241–248.

[25] S. Wang, J. Wu, X. Sun, W. Yuan, W. T. Freeman, J. B. Tenenbaum, and E. H. Adelson, "3D shape perception from monocular vision, touch, and shape priors," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 1606–13.

[26] E. J. Smith, R. Calandra, A. Romero, G. Gkioxari, D. Meger, J. Malik, and M. Drozdzal, "3D shape reconstruction from vision and touch," *arXiv preprint arXiv:2007.03778*, 2020.

[27] G. Izatt, G. Mirano, E. Adelson, and R. Tedrake, "Tracking objects with point clouds from vision and touch," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017.

[28] Y. Li, J.-Y. Zhu, R. Tedrake, and A. Torralba, "Connecting touch and vision via cross-modal prediction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition 2019*.

[29] S. Luo, W. Mou, K. Althoefer, and H. Liu, "Localizing the object contact through matching tactile features with visual map," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*.

[30] R. Kelly, R. Carelli, O. Nasisi, B. Kuchen, and F. Reyes, "Stable visual servoing of camera-in-hand robotic systems," *IEEE/ASME Transactions on Mechatronics*, vol. 5, no. 1, pp. 39–48, 2000.

[31] T. Mori and S. Scherer, "First results in detecting and avoiding frontal obstacles from a monocular camera for micro unmanned aerial vehicles," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 1750–1757.

[32] J. R. Serres and F. Ruffier, "Optic flow-based collision-free strategies: From insects to robots," *Arthropod structure & development*, vol. 46, no. 5, pp. 703–717, 2017.

[33] D. Raviv, *A quantitative approach to looming*. US Department of Commerce, National Institute of Standards and Technology, 1992.

[34] D. Raviv and K. Joarder, "The visual looming navigation cue: a unified approach," *Computer Vision and Image Understanding*, 2000.

[35] G. Yang and D. Ramanan, "Upgrading optical flow to 3d scene flow through optical expansion," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

[36] S. Luo, W. Mou, K. Althoefer, and H. Liu, "Novel tactile-SIFT descriptor for object shape recognition," *IEEE Sensors Journal*, vol. 15, no. 9, pp. 5001–5009, 2015.

[37] ——, "Iterative closest labeled point for tactile object shape recognition," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 3137–3142.

[38] E. Donlon, S. Dong, M. Liu, J. Li, E. Adelson, and A. Rodriguez, "Gelslim: A high-resolution, compact, robust, and calibrated tactile-sensing finger," pp. 1927–1934, 2018. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8593661

[39] R. Li, R. Platt, W. Yuan, A. ten Pas, N. Roscup, M. A. Srinivasan, and E. Adelson, "Localization and manipulation of small parts using gelsight tactile sensing," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 3988–3993.

[40] Y. She, S. Wang, S. Dong, N. Sunil, A. Rodriguez, and E. Adelson, "Cable manipulation with a tactile-reactive gripper," 2019.

[41] M. Bauza, O. Canal, and A. Rodriguez, "Tactile mapping and localization from high-resolution tactile imprints," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019.

[42] M. Bauza, E. Valls, B. Lim, T. Sechopoulos, and A. Rodriguez, "Tactile object pose estimation from the first touch with geometric contact rendering," *arXiv preprint arXiv:2012.05205*, 2020.

[43] P. Sodhi, M. Kaess, M. Mukadam, and S. Anderson, "Learning Tactile Models for Factor Graph-based State Estimation," *arXiv preprint arXiv:2012.03768*, 2020.

[44] A. Alspach, K. Hashimoto, N. Kuppuswamy, and R. Tedrake, "Soft-bubble: A highly compliant dense geometry tactile sensor for robot manipulation," in *2019 2nd IEEE International Conference on Soft Robotics (RoboSoft)*. IEEE, 2019, pp. 597–604.

[45] S. Suresh, Z. Si, J. G. Mangelson, W. Yuan, and M. Kaess, "Efficient shape mapping through dense touch and vision," *arXiv preprint arXiv:2109.09884*, 2021.

[46] S. Dikhale, K. Patel, D. Dhingra, I. Naramura, A. Hayashi, S. Iba, and N. Jamali, "VisuoTactile 6D Pose Estimation of an In-Hand Object using Vision and Tactile Sensor Data," *IEEE Robotics and Automation Letters*, 2022.

[47] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "PoseCNN: A convolutional neural network for 6d object pose estimation in cluttered scenes," *arXiv preprint arXiv:1711.00199*, 2017.

[48] M. K. Johnson, F. Cole, A. Raj, and E. H. Adelson, "Microgeometry capture using an elastomeric sensor," *ACM Transactions on Graphics (TOG)*, vol. 30, no. 4, pp. 1–8, 2011.

[49] OpenCV, "Open Source Computer Vision Library," 2021.

[50] G. Farnebäck, "Two-frame motion estimation based on polynomial expansion," in *Scand. conf. on Image anal.* Springer, 2003.

[51] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert, "High accuracy optical flow estimation based on a theory for warping," in *European conference on computer vision.* Springer, 2004, pp. 25–36.

[52] M.-Y. Liu, O. Tuzel, A. Veeraraghavan, Y. Taguchi, T. K. Marks, and R. Chellappa, "Fast object localization and pose estimation in heavy clutter for robotic bin picking," *The International Journal of Robotics Research*, vol. 31, no. 8, pp. 951–973, 2012.

[53] M. Imperoli and A. Pretto, "D^2CO: Fast and Robust Registration of 3D Textureless Objects Using the Directional Chamfer Distance," in *International Conference on Computer Vision Systems.* Springer, 2015.

[54] C. Choi and H. I. Christensen, "3d textureless object detection and tracking: An edge-based approach," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems.* IEEE, 2012.

[55] P. Henderson and V. Ferrari, "Learning single-image 3d reconstruction by generative modelling of shape, pose and shading," *International Journal of Computer Vision*, pp. 1–20, 2019.

[56] P. F. Felzenszwalb and D. P. Huttenlocher, "Distance transforms of sampled functions," *Theory of Computing*, pp. 415–428, 2012.

[57] S. M. Prakhya, B. Liu, W. Lin, V. Jakhetiya, and S. C. Guntuku, "B-SHOT: a binary 3D feature descriptor for fast Keypoint matching on 3D point clouds," *Autonomous Robots*, vol. 41, no. 7, 2017.

[58] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (FPFH) for 3D registration," in *2009 IEEE International Conference on Robotics and Automation.* IEEE, 2009, pp. 3212–3217.

[59] B. Calli, A. Singh, J. Bruce, A. Walsman, K. Konolige, S. Srinivasa, P. Abbeel, and A. M. Dollar, "Yale-CMU-Berkeley dataset for robotic manipulation research," *The International Journal of Robotics Research*, vol. 36, no. 3, pp. 261–268, 2017.

[60] E. Corona, K. Kundu, and S. Fidler, "Pose estimation for objects with rotational symmetry," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* IEEE, 2018, pp. 7215–7222.

[61] "Einscan-SE." [Online]. Available: https://www.einscan.com/desktop-3d-scanners/einscan-se/

[62] Y. Kuniyoshi, N. Kita, T. Suehiro, and S. Rougeaux, "Active stereo vision system with foveated wide angle lenses," in *Recent Developments in Computer Vision: Second Asian Conference on Computer Vision, ACCV'95 Singapore, December 5–8, 1995 Invited Session Papers 2.* Springer, 1996, pp. 191–200.

[63] Wikipedia, "Stucco," 2022, [Online; accessed 09-July-2022].

[64] J. L. Schönberger, E. Zheng, M. Pollefeys, and J.-M. Frahm, "Pixelwise view selection for unstructured multi-view stereo," in *European Conference on Computer Vision (ECCV)*, 2016. [Online]. Available: https://colmap.github.io/

[65] J. Aloimonos, "Shape from texture," *Biological Cybernetics*, vol. 58, no. 5, pp. 345–360, 1988.

[66] D. Verbin and T. Zickler, "Toward a Universal Model for Shape From Texture," in *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[67] "Photoneo PhoXi3D scanners." [Online]. Available: https://www.photoneo.com/phoxi-3d-scanner/

[68] "Ensenso XR series scanners." [Online]. Available: https://www.ids-imaging.us/ensenso-3d-camera-xr-series.html

[69] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, *et al.*, "Scalable deep reinforcement learning for vision-based robotic manipulation," in *Conference on Robot Learning*. PMLR, 2018, pp. 651–673.

[70] O. Kroemer, S. Niekum, and G. Konidaris, "A review of robot learning for manipulation: Challenges, representations, and algorithms," *The Journal of Machine Learning Research*, vol. 22, no. 1, pp. 1395–1476, 2021.

[71] B. K. Horn, "Shape from shading: A method for obtaining the shape of a smooth opaque object from one view," 1970.

[72] R. J. Woodham, "Photometric method for determining surface orientation from multiple images," *Optical Engineering*, vol. 19, no. 1, pp. 139–144, 1980.

[73] J. T. Barron and J. Malik, "Shape, illumination, and reflectance from shading," *IEEE Transactions on Pattern Analysis and Machine Isntelligence*, vol. 37, no. 8, pp. 1670–1687, 2014.

[74] Y. Xiong, A. Chakrabarti, R. Basri, S. J. Gortler, D. W. Jacobs, and T. Zickler, "From shading to local shape," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 1, pp. 67–79, 2014.

[75] F. Luan, S. Zhao, K. Bala, and Z. Dong, "Unified shape and svbrdf recovery using differentiable Monte Carlo rendering," in *Computer Graphics Forum*, vol. 40, no. 4. Wiley Online Library, 2021, pp. 101–113.

[76] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "NeRF: Representing scenes as neural radiance fields for view synthesis," *Communications of the ACM*, vol. 65, no. 1, pp. 99–106, 2021.

[77] M. Boss, R. Braun, V. Jampani, J. T. Barron, C. Liu, and H. Lensch, "NeRD: Neural reflectance decomposition from image collections," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 12 684–12 694.

[78] P. P. Srinivasan, B. Deng, X. Zhang, M. Tancik, B. Mildenhall, and J. T. Barron, "NeRV: Neural reflectance and visibility fields for relighting and view synthesis," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 7495–7504.

[79] S. Fridovich-Keil, A. Yu, M. Tancik, Q. Chen, B. Recht, and A. Kanazawa, "Plenoxels: Radiance Fields Without Neural Networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 5501–5510.

[80] L. Yen-Chen, P. Florence, J. T. Barron, T.-Y. Lin, A. Rodriguez, and P. Isola, "NeRF-Supervision: Learning Dense Object Descriptors from Neural Radiance Fields," *arXiv preprint arXiv:2203.01913*, 2022.

[81] L. Yen-Chen, P. Florence, J. T. Barron, A. Rodriguez, P. Isola, and T.-Y. Lin, "iNeRF: Inverting neural radiance fields for pose estimation," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 1323–1330.

[82] W. Yang, G. Chen, C. Chen, Z. Chen, and K.-Y. K. Wong, "PS-NeRF: Neural inverse rendering for multi-view photometric stereo," in *European Conference on Computer Vision*. Springer, 2022, pp. 266–284.

[83] Z. Cheng, J. Li, and H. Li, "Wildlight: In-the-wild inverse rendering with a flashlight," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 4305–4314. [Online]. Available: https://junxuan-li.github.io/wildlight-website/

[84] C. Schmitt, B. Antić, A. Neculai, J. H. Lee, and A. Geiger, "Towards scalable multi-view reconstruction of geometry and materials," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=10251698

[85] C. Schmitt, S. Donne, G. Riegler, V. Koltun, and A. Geiger, "On joint estimation of pose, geometry and svbrdf from a handheld scanner," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. [Online]. Available: https://openaccess.thecvf.com/content_CVPR_2020/html/Schmitt_On_Joint_Estimation_of_Pose_Geometry_and_svBRDF_From_a_CVPR_2020_paper.html

[86] T. Higo, Y. Matsushita, N. Joshi, and K. Ikeuchi, "A hand-held photometric stereo camera for 3-d modeling," in *2009 IEEE 12th International Conference on Computer Vision*. IEEE, 2009, pp. 1234–1241. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/5459331

[87] W. Yang, G. Chen, C. Chen, Z. Chen, and K.-Y. K. Wong, "S3-NeRF: Neural Reflectance Field from Shading and Shadow under a Single Viewpoint," *arXiv preprint arXiv:2210.08936*, 2022.

[88] K. Tiwary, T. Klinghoffer, and R. Raskar, "Towards learning neural representations from shadows," *arXiv preprint arXiv:2203.15946*, 2022.

[89] Y. H.-O. Asaf Karnieli, Ohad Fried, "DeepShadow: Neural shape from shadows," in *ECCV*, 2022.

[90] G. Inc., "GelSight Mobile," [Online; accessed 09-Dec-2022].

[91] ——, "GelSight Mini," [Online; accessed 09-Dec-2022].

[92] S. Barsky and M. Petrou, "The 4-source photometric stereo technique for three-dimensional surfaces in the presence of highlights and shadows," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 10, pp. 1239–1252, 2003.

[93] R. Ramamoorthi and P. Hanrahan, "An efficient representation for irradiance environment maps," in *Proceedings of the 28th annual conference on Computer graphics and Interactive Techniques*, 2001, pp. 497–500.

[94] "Centric daylight led strip lights for commercial & retail." [Online]. Available: https://store.waveformlighting.com/collections/led-strips/products/ultra-high-95-cri-led-strip-lights-for-commercial?variant=12104387919974

[95] FLIR, "Grasshopper3 USB3 model: GS3-U3-41C6C," 2024. [Online]. Available: https://www.flir.com/products/grasshopper3-usb3/?model=GS3-U3-41C6C-C

[96] EdmundOptics, "C series fixed focal length lenses: Edmund Optics," 2024. [Online]. Available: https://www.edmundoptics.com/f/c-series-fixed-focal-length-lenses/13679/

[97] "Ufactory xArm 7." [Online]. Available: https://www.ufactory.cc/product-page/ufactory-xarm-7

[98] R. Basri and D. W. Jacobs, "Lambertian reflectance and linear subspaces," *IEEE transactions on pattern analysis and machine intelligence*, vol. 25, no. 2, pp. 218–233, 2003.

[99] M. Tappen, W. Freeman, and E. Adelson, "Recovering intrinsic images from a single image," *Advances in Neural Information Processing Systems*, vol. 15, 2002.

[100] C. Rother, M. Kiefel, L. Zhang, B. Schölkopf, and P. Gehler, "Recovering intrinsic images with a global sparsity prior on reflectance," *Advances in Neural Information Processing Systems*, vol. 24, 2011.

[101] S. Wright, J. Nocedal, *et al.*, "Numerical optimization," *Springer Science*, vol. 35, no. 67-68, p. 7, 1999.

[102] R. Raskar, K.-H. Tan, R. Feris, J. Yu, and M. Turk, "Non-photorealistic camera: depth edge detection and stylized rendering using multi-flash imaging," *ACM Transactions on Graphics (TOG)*, vol. 23, no. 3, pp. 679–688, 2004.

[103] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, no. 6, pp. 679–698, 1986.

[104] Y. Quéau, J.-D. Durou, and J.-F. Aujol, "Normal Integration: A Survey," *Journal of Mathematical Imaging and Vision*, vol. 60, no. 4, pp. 576–593, 2018.

[105] B. K. Horn and M. J. Brooks, "The variational approach to shape from shading," *Computer Vision, Graphics, and Image Processing*, vol. 33, no. 2, pp. 174–208, 1986.

[106] D. Eberly, "Reconstructing a Height Field from a Normal Map," *Geometric Tools, Redmond WA 98052. USA*. [Online]. Available: https://www.geometrictools.com/

[107] R. T. Frankot and R. Chellappa, "A method for enforcing integrability in shape from shading algorithms," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 10, no. 4, pp. 439–451, 1988.

[108] X. Cao, H. Santo, B. Shi, F. Okura, and Y. Matsushita, "Bilateral normal integration," in *European Conference on Computer Vision*. Springer, 2022, pp. 552–567.

[109] Y. Rubner, C. Tomasi, and L. J. Guibas, "The earth mover's distance as a metric for image retrieval," *International Journal of Computer Vision*, vol. 40, no. 2, pp. 99–121, 2000.

[110] H. Ling and K. Okada, "An efficient earth mover's distance algorithm for robust histogram comparison," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 5, pp. 840–853, 2007.

[111] L. Keselman, J. Iselin Woodfill, A. Grunnet-Jepsen, and A. Bhowmik, "Intel realsense stereoscopic depth cameras," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2017, pp. 1–10.

[112] S. Rusinkiewicz and M. Levoy, "Efficient variants of the ICP algorithm," in *Proceedings of the Third International Conference on 3-D Digital Imaging and Modeling*. IEEE, 2001, pp. 145–152.

[113] P. Cignoni, C. Rocchini, and R. Scopigno, "Metro: measuring error on simplified surfaces," in *Computer Graphics Forum*, vol. 17, no. 2. Wiley Online Library, 1998, pp. 167–174.

[114] G. R. Bradski, "Computer vision face tracking for use in a perceptual user interface," 1998.

[115] T. Flash and N. Hogan, "The coordination of arm movements: an experimentally confirmed mathematical model," *Journal of Neuroscience*, vol. 5, no. 7, pp. 1688–1703, 1985.

[116] Y. Wi, P. Florence, A. Zeng, and N. Fazeli, "VIRDO: Visio-tactile implicit representations of deformable objects," *arXiv preprint arXiv:2202.00868*, 2022.

[117] Y. Wi, A. Zeng, P. Florence, and N. Fazeli, "VIRDO++: Real-World, Visuo-tactile Dynamics and Perception of Deformable Objects," *arXiv preprint arXiv:2210.03701*, 2022.

[118] P. Das, S. Karaoglu, and T. Gevers, "Pie-net: Photometric invariant edge guided network for intrinsic image decomposition," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 19 790–19 799. [Online]. Available: https://openaccess.thecvf.com/content/CVPR2022/html/Das_PIE-Net_Photometric_Invariant_Edge_Guided_Network_for_Intrinsic_Image_Decomposition_CVPR_2022_paper.html

[119] B. Shi, Z. Wu, Z. Mo, D. Duan, S.-K. Yeung, and P. Tan, "A benchmark dataset and evaluation for non-lambertian and uncalibrated photometric stereo," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 3707–3716.

[120] J. T. Barron and J. Malik, "Intrinsic scene properties from a single rgb-d image," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2013.

[121] A. S. Baslamisli, H.-A. Le, and T. Gevers, "CNN Based Learning Using Reflection and Retinex Models for Intrinsic Image Decomposition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[122] D. Forsyth and J. J. Rock, "Intrinsic image decomposition using paradigms," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 11, pp. 7624–7637, 2021.

[123] A. Dave, Y. Zhao, and A. Veeraraghavan, "Pandora: Polarization-aided neural decomposition of radiance," in *European Conference on Computer Vision*. Springer, 2022, pp. 538–556. [Online]. Available: https://akshatdave.github.io/pandora/index.html

[124] N. Ravi, J. Reizenstein, D. Novotny, T. Gordon, W.-Y. Lo, J. Johnson, and G. Gkioxari, "Accelerating 3d deep learning with PyTorch3D," *arXiv preprint arXiv:2007.08501*, 2020.

[125] Q.-Y. Zhou, J. Park, and V. Koltun, "Fast global registration," in *European Conference on Computer Vision*. Springer, 2016, pp. 766–782.

[126] ——, "Open3D: A modern library for 3D data processing," *arXiv preprint arXiv:1801.09847*, 2018.

[127] Blender, *Blender - a 3D modelling and rendering package*, Blender Foundation, Blender Institute, Amsterdam, 2024. [Online]. Available: http://www.blender.org

[128] Ginibird, "Garden tea lights [license: Cc-0]," Mar 2021. [Online]. Available: https://www.blendswap.com/blend/27683

[129] P. Boudoin, "Hyper capture: 3d object scan," Dec 2023. [Online]. Available: https://apps.apple.com/us/app/hyper-capture-3d-object-scan/

[130] LumaLabs, "Luma ai: Ai for gorgeous 3d capture," Dec 2023. [Online]. Available: https://lumalabs.ai/

[131] 3DZephyr, "3df zephyr - photogrammetry software - 3d models from photos," Mar 2022. [Online]. Available: https://www.3dflow.net/3df-zephyr-photogrammetry-software/

[132] RealityCapture, "Realitycapture: 3d models from photos and/or laser scans," Mar 2022. [Online]. Available: https://www.capturingreality.com/

[133] AliceVision, "Alicevision meshroom," Mar 2022. [Online]. Available: https://alicevision.org/

[134] M. Tancik, E. Weber, E. Ng, R. Li, B. Yi, J. Kerr, T. Wang, A. Kristoffersen, J. Austin, K. Salahi, A. Ahuja, D. McAllister, and A. Kanazawa, "Nerfstudio: A modular framework for neural radiance field development," in *ACM SIGGRAPH 2023 Conference Proceedings*, ser. SIGGRAPH '23, 2023. [Online]. Available: https://docs.nerf.studio/

[135] K. N. Kutulakos and S. M. Seitz, "A theory of shape by space carving," in *Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 1. IEEE, 1999, pp. 307–314. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=791235

[136] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," *Communications of the ACM*, vol. 65, no. 1, pp. 99–106, 2021. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/3503250

[137] T. Müller, A. Evans, C. Schied, and A. Keller, "Instant neural graphics primitives with a multiresolution hash encoding," *ACM Transactions on Graphics (ToG)*, vol. 41, no. 4, pp. 1–15, 2022. [Online]. Available: https://dl.acm.org/doi/pdf/10.1145/3528223.3530127

[138] S. Fridovich-Keil, A. Yu, M. Tancik, Q. Chen, B. Recht, and A. Kanazawa, "Plenoxels: Radiance fields without neural networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 5501–5510. [Online]. Available: https://openaccess.thecvf.com/content/CVPR2022/html/Fridovich-Keil_Plenoxels_Radiance_Fields_Without_Neural_Networks_CVPR_2022_paper.html

[139] K. Kang, M. Gu, C. Xie, X. Yang, H. Wu, and K. Zhou, "Neural reflectance capture in the view-illumination domain," *IEEE Transactions on Visualization and Computer Graphics*, vol. 29, no. 2, pp. 1450–1462, 2021. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9557800

[140] B. Roessle, J. T. Barron, B. Mildenhall, P. P. Srinivasan, and M. Nießner, "Dense depth priors for neural radiance fields from sparse input views," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022. [Online]. Available: https://barbararoessle.github.io/dense_depth_priors_nerf/

[141] Z. Yu, S. Peng, M. Niemeyer, T. Sattler, and A. Geiger, "Monosdf: Exploring monocular geometric cues for neural implicit surface reconstruction," *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. [Online]. Available: https://niujinshuchong.github.io/monosdf/

[142] A. Eftekhar, A. Sax, J. Malik, and A. Zamir, "Omnidata: A scalable pipeline for making multi-task mid-level vision datasets from 3d scans," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 10 786–10 796. [Online]. Available: https://openaccess.thecvf.com/content/ICCV2021/html/Eftekhar_Omnidata_A_Scalable_Pipeline_for_Making_Multi-Task_Mid-Level_Vision_Datasets_ICCV_2021_paper.html

[143] X. Cheng, P. Wang, and R. Yang, "Learning depth with convolutional spatial propagation network," *IEEE transactions on pattern analysis and machine intelligence*, vol. 42, no. 10, pp. 2361–2379, 2019. [Online]. Available: https://arxiv.org/pdf/1810.02695.pdf

[144] Shining3D, "Einscansp," Jul 2023. [Online]. Available: https://www.einscan.com/desktop-3d-scanners/

[145] "Ensenso XR series scanners," Dec 2023. [Online]. Available: https://www.ids-imaging.us/ensenso-3d-camera-xr-series.html

[146] "Photoneo PhoXi3D scanners," Dec 2023. [Online]. Available: https://www.photoneo.com/phoxi-3d-scanner/

[147] Matterport, "Matterport: Drive results with digital twins." Dec 2023. [Online]. Available: https://matterport.com/

[148] "Artecleo," Dec 2023. [Online]. Available: https://www.artec3d.com/portable-3d-scanners/artec-leo

[149] Y. Zhang, N. Wadhwa, S. Orts-Escolano, C. Häne, S. Fanello, and R. Garg, "Du 2 net: Learning depth estimation from dual-cameras and dual-pixels," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16*. Springer, 2020, pp. 582–598. [Online]. Available: https://augmentedperception.github.io/du2net/

[150] ZDNet, Mar 2023. [Online]. Available: https://www.zdnet.com/article/how-to-use-lidar-on-the-iphone-and-ipad/

[151] L. Keselman, J. Iselin Woodfill, A. Grunnet-Jepsen, and A. Bhowmik, "Intel RealSense stereoscopic depth cameras," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2017, pp. 1–10. [Online]. Available: https://openaccess.thecvf.com/content_cvpr_2017_workshops/w15/papers/Keselman_Intel_RealSense_Stereoscopic_CVPR_2017_paper.pdf

[152] "Azure Kinect 3D sensors," Dec 2023. [Online]. Available: https://www.microsoft.com/en-us/d/azure-kinect-dk

[153] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, 1996, pp. 303–312. [Online]. Available: https://dl.acm.org/doi/pdf/10.1145/237170.237269

[154] M. Kazhdan and H. Hoppe, "Screened poisson surface reconstruction," *ACM Transactions on Graphics (ToG)*, vol. 32, no. 3, pp. 1–13, 2013. [Online]. Available: https://www.cs.jhu.edu/~misha/MyPapers/ToG13.pdf

[155] S. Galliani, K. Lasinger, and K. Schindler, "Massively parallel multiview stereopsis by surface normal diffusion," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015. [Online]. Available: https://openaccess.thecvf.com/content_iccv_2015/html/Galliani_Massively_Parallel_Multiview_ICCV_2015_paper.html

[156] A. Dai, M. Nießner, M. Zollhöfer, S. Izadi, and C. Theobalt, "Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration," *ACM Transactions on Graphics (ToG)*, vol. 36, no. 4, p. 1, 2017. [Online]. Available: https://dl.acm.org/doi/pdf/10.1145/3072959.3054739

[157] L. Yariv, J. Gu, Y. Kasten, and Y. Lipman, "Volume rendering of neural implicit surfaces," *Advances in Neural Information Processing Systems*, vol. 34, pp. 4805–4815, 2021. [Online]. Available: https://lioryariv.github.io/volsdf/

[158] P. Wang, L. Liu, Y. Liu, C. Theobalt, T. Komura, and W. Wang, "Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction," *arXiv preprint arXiv:2106.10689*, 2021. [Online]. Available: https://proceedings.neurips.cc/paper/2021/file/e41e164f7485ec4a28741a2d0ea41c74-Paper.pdf

[159] Z. Li, T. Müller, A. Evans, R. H. Taylor, M. Unberath, M.-Y. Liu, and C.-H. Lin, "Neuralangelo: High-fidelity neural surface reconstruction," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. [Online]. Available: https://research.nvidia.com/labs/dir/neuralangelo/

[160] M. Tancik, P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. Barron, and R. Ng, "Fourier features let networks learn high frequency functions in low dimensional domains," *Advances in Neural Information Processing Systems*, vol. 33, pp. 7537–7547, 2020. [Online]. Available: https://proceedings.neurips.cc/paper/2020/file/55053683268957697aa39fba6f231c68-Paper.pdf

[161] J. Sun, X. Chen, Q. Wang, Z. Li, H. Averbuch-Elor, X. Zhou, and N. Snavely, "Neural 3d reconstruction in the wild," in *ACM SIGGRAPH 2022 Conference Proceedings*, 2022, pp. 1–9. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/3528233.3530718

[162] D. Azinović, R. Martin-Brualla, D. B. Goldman, M. Nießner, and J. Thies, "Neural rgb-d surface reconstruction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022, pp. 6290–6301. [Online]. Available: https://dazinovic.github.io/neural-rgbd-surface-reconstruction/

[163] E. Sandström, Y. Li, L. Van Gool, and M. R. Oswald, "Point-slam: Dense neural point cloud-based slam," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023. [Online]. Available: https://github.com/eriksandstroem/Point-SLAM

[164] Q. Fu, Q. Xu, Y. S. Ong, and W. Tao, "Geo-NeUS: Geometry-consistent neural implicit surfaces learning for multi-view reconstruction," *Advances in Neural Information Processing Systems*, vol. 35, pp. 3403–3416, 2022. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2022/file/16415eed5a0a121bfce79924db05d3fe-Paper-Conference.pdf

[165] V. Guizilini, I. Vasiljevic, J. Fang, R. Ambrus, S. Zakharov, V. Sitzmann, and A. Gaidon, "Delira: Self-supervised depth, light, and radiance fields," *arXiv preprint arXiv:2304.02797*, 2023. [Online]. Available: https://sites.google.com/view/tri-delira

[166] B. Attal, E. Laidlaw, A. Gokaslan, C. Kim, C. Richardt, J. Tompkin, and M. O'Toole, "Törf: Time-of-flight radiance fields for dynamic scene view synthesis," *Advances in neural information processing systems*, vol. 34, pp. 26 289–26 301, 2021. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2021/file/dd03de08bfdff4d8ab01117276564cc7-Paper.pdf

[167] A. Shandilya, B. Attal, C. Richardt, J. Tompkin, and M. O'toole, "Neural fields for structured lighting," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2023, pp. 3512–3522. [Online]. Available: https://openaccess.thecvf.com/content/ICCV2023/html/Shandilya_Neural_Fields_for_Structured_Lighting_ICCV_2023_paper.html

[168] S. Klenk, L. Koestler, D. Scaramuzza, and D. Cremers, "E-nerf: Neural radiance fields from a moving event camera," *IEEE Robotics and Automation Letters*, vol. 8, no. 3, pp. 1587–1594, 2023. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10028738

[169] W. F. Low and G. H. Lee, "Robust e-nerf: Nerf from sparse & noisy events under non-uniform motion," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2023, pp. 18 335–18 346. [Online]. Available: https://openaccess.thecvf.com/content/ICCV2023/html/Low_Robust_e-NeRF_NeRF_from_Sparse__Noisy_Events_under_Non-Uniform_ICCV_2023_paper.html

[170] P. F. U. Gotardo, T. Simon, Y. Sheikh, and I. Matthews, "Photogeometric scene flow for high-detail dynamic 3d reconstruction," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015. [Online]. Available: https://openaccess.thecvf.com/content_iccv_2015/html/Gotardo_Photogeometric_Scene_Flow_ICCV_2015_paper.html

[171] H. Xu, J. Zhang, J. Cai, H. Rezatofighi, and D. Tao, "Gmflow: Learning optical flow via global matching," in *Proceedings of the IEEE/CVF Conference on*

Computer Vision and Pattern Recognition, 2022, pp. 8121–8130. [Online]. Available: https://github.com/autonomousvision/unimatch

[172] K. Kang, C. Xie, C. He, M. Yi, M. Gu, Z. Chen, K. Zhou, and H. Wu, "Learning efficient illumination multiplexing for joint capture of reflectance and shape." *ACM Trans. Graph.*, vol. 38, no. 6, pp. 165–1, 2019. [Online]. Available: https://svbrdf.github.io/publications/jointcap/jointcap.pdf

[173] Z. Zhou, Z. Wu, and P. Tan, "Multi-view photometric stereo with spatially varying isotropic materials," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 1482–1489. [Online]. Available: https://openaccess.thecvf.com/content_cvpr_2013/papers/Zhou_Multi-view_Photometric_Stereo_2013_CVPR_paper.pdf

[174] K. Zhang, F. Luan, Z. Li, and N. Snavely, "Iron: Inverse rendering by optimizing neural sdfs and materials from photometric images," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 5565–5574. [Online]. Available: https://openaccess.thecvf.com/content/CVPR2022/html/Zhang_IRON_Inverse_Rendering_by_Optimizing_Neural_SDFs_and_Materials_From_CVPR_2022_paper.html

[175] S. Bi, Z. Xu, P. Srinivasan, B. Mildenhall, K. Sunkavalli, M. Hašan, Y. Hold-Geoffroy, D. Kriegman, and R. Ramamoorthi, "Neural reflectance fields for appearance acquisition," *arXiv preprint arXiv:2008.03824*, 2020. [Online]. Available: https://arxiv.org/pdf/2008.03824.pdf

[176] T. Mertens, J. Kautz, and F. Van Reeth, "Exposure fusion," in *15th Pacific Conference on Computer Graphics and Applications (PG'07)*. IEEE, 2007, pp. 382–390. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4392748

[177] E. Reinhard, M. Stark, P. Shirley, and J. Ferwerda, "Photographic tone reproduction for digital images," in *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*, 2023, pp. 661–670. [Online]. Available: https://dl.acm.org/doi/10.1145/566654.566575

[178] H. Hirschmuller, "Accurate and efficient stereo processing by semi-global matching and mutual information," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 2. IEEE, 2005, pp. 807–814. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1467526

[179] N. Max, "Optical models for direct volume rendering," *IEEE Transactions on Visualization and Computer Graphics*, vol. 1, no. 2, pp. 99–108, 1995. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=468400

[180] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014. [Online]. Available: https://arxiv.org/pdf/1412.6980.pdf

[181] M. Oechsle, S. Peng, and A. Geiger, "Unisurf: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 5589–5599. [Online]. Available: https://moechsle.github.io/unisurf/

[182] M. Zollhöfer, A. Dai, M. Innmann, C. Wu, M. Stamminger, C. Theobalt, and M. Nießner, "Shading-based refinement on volumetric signed distance functions," *ACM Transactions on Graphics (ToG)*, vol. 34, no. 4, pp. 1–14, 2015. [Online]. Available: https://dl.acm.org/doi/pdf/10.1145/2766887

[183] A. Gropp, L. Yariv, N. Haim, M. Atzmon, and Y. Lipman, "Implicit geometric regularization for learning shapes," *arXiv preprint arXiv:2002.10099*, 2020. [Online]. Available: https://arxiv.org/abs/2002.10099

[184] M. G. Crandall and P.-L. Lions, "Viscosity solutions of hamilton-jacobi equations," *Transactions of the American mathematical society*, vol. 277, no. 1, pp. 1–42, 1983. [Online]. Available: https://www.ams.org/journals/tran/1983-277-01/S0002-9947-1983-0690039-8/S0002-9947-1983-0690039-8.pdf

[185] L. Yariv, P. Hedman, C. Reiser, D. Verbin, P. P. Srinivasan, R. Szeliski, J. T. Barron, and B. Mildenhall, "Bakedsdf: Meshing neural sdfs for real-time view synthesis," *arXiv preprint arXiv:2302.14859*, 2023. [Online]. Available: https://bakedsdf.github.io/

[186] Z. Wang, T. Shen, M. Nimier-David, N. Sharp, J. Gao, A. Keller, S. Fidler, T. Müller, and Z. Gojcic, "Adaptive shells for efficient neural radiance field rendering," *ACM Trans. Graph.*, vol. 42, no. 6, 2023. [Online]. Available: https://doi.org/10.1145/3618390

[187] R. Jensen, A. Dahl, G. Vogiatzis, E. Tola, and H. Aanæs, "Large scale multi-view stereopsis evaluation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 406–413. [Online]. Available: https://www.cv-foundation.org/openaccess/content_cvpr_2014/papers/Jensen_Large_Scale_Multi-view_2014_CVPR_paper.pdf

[188] A. Knapitsch, J. Park, Q.-Y. Zhou, and V. Koltun, "Tanks and temples: Benchmarking large-scale scene reconstruction," *ACM Transactions on Graphics (ToG)*, vol. 36, no. 4, pp. 1–13, 2017. [Online]. Available: https://dl.acm.org/doi/pdf/10.1145/3072959.3073599

[189] P. F. Felzenszwalb and D. P. Huttenlocher, "Distance transforms of sampled functions," *Theory of computing*, vol. 8, no. 1, pp. 415–428, 2012. [Online]. Available: https://theoryofcomputing.org/articles/v008a019/v008a019.pdf

[190] L. Goli, C. Reading, S. Sellán, A. Jacobson, and A. Tagliasacchi, "Bayes' Rays: Uncertainty quantification in neural radiance fields," *CVPR*, 2024. [Online]. Available: https://bayesrays.github.io/

[191] M. Brahimi, B. Haefner, T. Yenamandra, B. Goldluecke, and D. Cremers, "Supervol: Super-resolution shape and reflectance estimation in inverse volume rendering," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2024, pp. 3139–3149. [Online]. Available: https://openaccess.thecvf.com/content/WACV2024/html/Brahimi_SupeRVol_Super-Resolution_Shape_and_Reflectance_Estimation_in_Inverse_Volume_Rendering_WACV_2024_paper.html

[192] R. Raskar, K.-H. Tan, R. Feris, J. Yu, and M. Turk, "Non-photorealistic camera: depth edge detection and stylized rendering using multi-flash imaging," *ACM transactions on graphics (TOG)*, vol. 23, no. 3, pp. 679–688, 2004. [Online]. Available: https://dl.acm.org/doi/pdf/10.1145/1015706.1015779

[193] R. Feris, R. Raskar, K.-H. Tan, and M. Turk, "Specular reflection reduction with multi-flash imaging," in *Proceedings. 17th Brazilian symposium on computer graphics and image processing*. IEEE, 2004, pp. 316–321. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1352976

[194] J. Li and H. Li, "Self-calibrating photometric stereo by neural inverse rendering," in *European Conference on Computer Vision*. Springer, 2022, pp. 166–183. [Online]. Available: https://arxiv.org/abs/2207.07815

[195] G. Chen, K. Han, B. Shi, Y. Matsushita, and K.-Y. K. Wong, "Self-calibrating deep photometric stereo networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8739–8747. [Online]. Available: https://arxiv.org/abs/1903.07366

[196] B. Shi, Z. Wu, Z. Mo, D. Duan, S.-K. Yeung, and P. Tan, "A benchmark dataset and evaluation for non-lambertian and uncalibrated photometric stereo," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 3707–3716. [Online]. Available: https://openaccess.thecvf.com/content_cvpr_2016/papers/Shi_A_Benchmark_Dataset_CVPR_2016_paper.pdf

[197] M. Li, Z. Zhou, Z. Wu, B. Shi, C. Diao, and P. Tan, "Multi-view photometric stereo: A robust solution and benchmark dataset for spatially varying isotropic materials," *IEEE Transactions on Image Processing*, vol. 29, pp. 4159–4173, 2020. [Online]. Available: https://arxiv.org/abs/2001.06659

[198] P. Mirdehghan, M. Wu, W. Chen, D. B. Lindell, and K. N. Kutulakos, "Turbosl: Dense accurate and fast 3d by neural inverse structured light," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2024, pp. 25 067–25 076. [Online]. Available: https://openaccess.thecvf.com/content/CVPR2024/html/Mirdehghan_TurboSL_Dense_Accurate_and_Fast_3D_by_Neural_Inverse_Structured_CVPR_2024_paper.html

[199] W. Chen, P. Mirdehghan, S. Fidler, and K. N. Kutulakos, "Auto-tuning structured light by optical stochastic gradient descent," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. [Online]. Available: https://www.dgp.toronto.edu/autotuningsl/

[200] Y. Yao, Z. Luo, S. Li, J. Zhang, Y. Ren, L. Zhou, T. Fang, and L. Quan, "Blendedmvs: A large-scale dataset for generalized multi-view stereo networks," *Computer Vision and Pattern Recognition (CVPR)*, 2020. [Online]. Available: https://github.com/YoYo000/BlendedMVS

[201] M. Toschi, R. De Matteo, R. Spezialetti, D. De Gregorio, L. Di Stefano, and S. Salti, "Relight my nerf: A dataset for novel view synthesis and relighting of real world objects," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2023, pp. 20 762–20 772. [Online]. Available: https://eyecan-ai.github.io/rene/

[202] I. Liu, L. Chen, Z. Fu, L. Wu, H. Jin, Z. Li, C. M. R. Wong, Y. Xu, R. Ramamoorthi, Z. Xu, *et al.*, "Openillumination: A multi-illumination dataset for inverse rendering evaluation on real objects," *arXiv preprint arXiv:2309.07921*, 2023. [Online]. Available: https://oppo-us-research.github.io/OpenIllumination/

[203] J. Straub, T. Whelan, L. Ma, Y. Chen, E. Wijmans, S. Green, J. J. Engel, R. Mur-Artal, C. Ren, S. Verma, A. Clarkson, M. Yan, B. Budge, Y. Yan, X. Pan, J. Yon, Y. Zou, K. Leon, N. Carter, J. Briales, T. Gillingham, E. Mueggler, L. Pesqueira, M. Savva, D. Batra, H. M. Strasdat, R. D. Nardi, M. Goesele, S. Lovegrove, and R. Newcombe, "The Replica dataset: A digital replica of indoor spaces," *arXiv preprint arXiv:1906.05797*, 2019. [Online]. Available: https://github.com/facebookresearch/Replica-Dataset

[204] J. Park, Q.-Y. Zhou, and V. Koltun, "Colored point cloud registration revisited," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 143–152. [Online]. Available: http://www.open3d.org/docs/release/tutorial/pipelines/colored_pointcloud_registration.html

[205] Q.-Y. Zhou, J. Park, and V. Koltun, "Fast global registration," in *European Conference on Computer Vision*. Springer, 2016, pp. 766–782. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-319-46475-6_47

[206] P.-E. Sarlin, C. Cadena, R. Siegwart, and M. Dymczyk, "From coarse to fine: Robust hierarchical localization at large scale," in *CVPR*, 2019. [Online]. Available: https://github.com/cvg/Hierarchical-Localization

[207] R. Zabih and J. Woodfill, "Non-parametric local transforms for computing visual correspondence," in *Computer Vision—ECCV'94: Third European Conference on Computer Vision Stockholm, Sweden, May 2–6 1994 Proceedings, Volume II 3*. Springer, 1994, pp. 151–158. [Online]. Available: https://woodfill.com/Papers/Census94.pdf

[208] B. Burley, "Physically-based shading at disney," in *Acm Siggraph*, vol. 2012. vol. 2012, 2012, pp. 1–7. [Online]. Available: https://media.disneyanimation.com/uploads/production/publication_asset/48/asset/s2012_pbs_disney_brdf_notes_v3.pdf

[209] J. Young, "xatlas: A small c++11 library with no external dependencies that generates unique texture coordinates suitable for baking lightmaps or texture painting." 2024. [Online]. Available: https://github.com/jpcy/xatlas

[210] P. P. Srinivasan, S. J. Garbin, D. Verbin, J. T. Barron, and B. Mildenhall, "Nuvo: Neural uv mapping for unruly 3d representations," *arXiv*, 2023. [Online]. Available: https://pratulsrinivasan.github.io/nuvo/

[211] Y. Liu, P. Wang, C. Lin, X. Long, J. Wang, L. Liu, T. Komura, and W. Wang, "Nero: Neural geometry and brdf reconstruction of reflective objects from multiview images," *arXiv preprint arXiv:2305.17398*, 2023. [Online]. Available: https://arxiv.org/abs/2305.17398

[212] C. Reiser, R. Szeliski, D. Verbin, P. Srinivasan, B. Mildenhall, A. Geiger, J. Barron, and P. Hedman, "Merf: Memory-efficient radiance fields for real-time view synthesis in unbounded scenes," *ACM Transactions on Graphics (TOG)*, vol. 42, no. 4, pp. 1–12, 2023. [Online]. Available: https://creiser.github.io/merf/

[213] D. Duckworth, P. Hedman, C. Reiser, P. Zhizhin, J.-F. Thibert, M. Lučić, R. Szeliski, and J. T. Barron, "Smerf: Streamable memory efficient radiance fields for real-time large-scene exploration," 2023. [Online]. Available: https://smerf-3d.github.io/

[214] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, "3d gaussian splatting for real-time radiance field rendering," *ACM Transactions on Graphics*, vol. 42, no. 4, 2023. [Online]. Available: https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/

[215] N. Moenne-Loccoz, A. Mirzaei, R. d. L. Or Perel, J. M. Esturo, G. State, S. Fidler, N. Sharp, and Z. Gojcic, "3D Gaussian Ray Tracing: Fast tracing of particle scenes," *ACM Transactions on Graphics and SIGGRAPH Asia*, 2024. [Online]. Available: https://gaussiantracer.github.io/

[216] A. Guédon and V. Lepetit, "Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering," *CVPR*, 2024. [Online]. Available: https://anttwo.github.io/sugar/

[217] ——, "Gaussian frosting: Editable complex radiance fields with real-time rendering," *ECCV*, 2024. [Online]. Available: https://anttwo.github.io/frosting/

[218] Y. Hua, C. Lassner, C. Stoll, and I. Matthews, "fnerf: High quality radiance fields from practical cameras," *arXiv preprint arXiv:2406.10633*, 2024. [Online]. Available: https://arxiv.org/abs/2406.10633

[219] R. Li, M. Tancik, and A. Kanazawa, "Nerfacc: A general nerf acceleration toolbox," *arXiv preprint arXiv:2210.04847*, 2022. [Online]. Available: https://www.nerfacc.com/

[220] R. Martin-Brualla, N. Radwan, M. S. M. Sajjadi, J. T. Barron, A. Dosovitskiy, and D. Duckworth, "NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections," in *CVPR*, 2021. [Online]. Available: https://nerf-w.github.io/

[221] K. Deng, A. Liu, J.-Y. Zhu, and D. Ramanan, “Depth-supervised NeRF: Fewer views and faster training for free,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022. [Online]. Available: https://www.cs.cmu.edu/~dsnerf/

[222] P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, and G. Ranzuglia, “MeshLab: an Open-Source Mesh Processing Tool,” in *Eurographics Italian Chapter Conference*, V. Scarano, R. D. Chiara, and U. Erra, Eds. The Eurographics Association, 2008. [Online]. Available: https://www.meshlab.net/

[223] Sketchfab, “Sketchfab: Online 3d geometry viewer,” 2024. [Online]. Available: https://sketchfab.com/

[224] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger, “Occupancy networks: Learning 3d reconstruction in function space,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. [Online]. Available: https://openaccess.thecvf.com/content_CVPR_2019/html/Mescheder_Occupancy_Networks_Learning_3D_Reconstruction_in_Function_Space_CVPR_2019_paper.html

[225] S. Woop, L. Feng, I. Wald, and C. Benthin, “Embree ray tracing kernels for cpus and the xeon phi architecture,” in *ACM SIGGRAPH 2013 Talks*, 2013, pp. 1–1. [Online]. Available: https://dl.acm.org/doi/pdf/10.1145/2504459.2504515

[226] CREE, “Cree xlamp cxa2540,” 2024. [Online]. Available: https://www.digikey.com/en/products/detail/creeled-inc/CXA2540-0000-000N00W257F/4437055

[227] B. Mildenhall, P. Hedman, R. Martin-Brualla, P. P. Srinivasan, and J. T. Barron, “NeRF in the dark: High dynamic range view synthesis from noisy raw images,” *CVPR*, 2022. [Online]. Available: https://bmild.github.io/rawnerf/

[228] S. Choi, Q.-Y. Zhou, and V. Koltun, “Robust reconstruction of indoor scenes,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 5556–5565. [Online]. Available: http://www.open3d.org/docs/release/tutorial/pipelines/multiway_registration.html

[229] J. L. Schönberger and J.-M. Frahm, “Structure-from-motion revisited,” in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[230] M. A. Fischler and R. C. Bolles, “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography,” *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[231] S. Umeyama, “Least-squares estimation of transformation parameters between two point patterns,” *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 13, no. 04, pp. 376–380, 1991. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=88573

[232] M. Chandraker, J. Bai, and R. Ramamoorthi, “On differential photometric reconstruction for unknown, isotropic brdfs,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 12, pp. 2941–2955, 2012. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6327191

[233] C. Liu, S. G. Narasimhan, and A. W. Dubrawski, “Near-light photometric stereo using circularly placed point light sources,” in *2018 IEEE International Conference on Computational Photography (ICCP)*. IEEE, 2018, pp. 1–10. [Online]. Available: https://www.cs.cmu.edu/~ILIM/projects/IM/nearPS/

[234] R. Feris, R. Raskar, L. Chen, K.-H. Tan, and M. Turk, "Discontinuity preserving stereo with small baseline multi-flash illumination," in *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, vol. 1. IEEE, 2005, pp. 412–419. [Online]. Available: https://rogerioferis.com/multi-flash-stereo/

[235] B. Ke, A. Obukhov, S. Huang, N. Metzger, R. C. Daudt, and K. Schindler, "Repurposing diffusion-based image generators for monocular depth estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 9492–9502.

[236] L. Yang, B. Kang, Z. Huang, X. Xu, J. Feng, and H. Zhao, "Depth anything: Unleashing the power of large-scale unlabeled data," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 10 371–10 381.

[237] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, *et al.*, "Segment anything," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 4015–4026.

[238] R. Ranftl, K. Lasinger, D. Hafner, K. Schindler, and V. Koltun, "Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer," *IEEE transactions on pattern analysis and machine intelligence*, vol. 44, no. 3, pp. 1623–1637, 2020.

[239] A. Vedaldi and S. Soatto, "Quick shift and kernel methods for mode seeking," in *Computer Vision–ECCV 2008: 10th European Conference on Computer Vision, Marseille, France, October 12-18, 2008, Proceedings, Part IV 10*. Springer, 2008, pp. 705–718.

[240] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "SLIC superpixels compared to state-of-the-art superpixel methods," *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 11, pp. 2274–2282, 2012.

[241] E. Brachmann, J. Wynn, S. Chen, T. Cavallari, Á. Monszpart, D. Turmukhambetov, and V. A. Prisacariu, "Scene coordinate reconstruction: Posing of image collections via incremental learning of a relocalizer," in *ECCV*, 2024. [Online]. Available: https://nianticlabs.github.io/acezero/

[242] S. Wang, V. Leroy, Y. Cabon, B. Chidlovskii, and J. Revaud, "Dust3r: Geometric 3d vision made easy," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 20 697–20 709. [Online]. Available: https://openaccess.thecvf.com/content/CVPR2024/html/Wang_DUSt3R_Geometric_3D_Vision_Made_Easy_CVPR_2024_paper.html

[243] V. Leroy, Y. Cabon, and J. Revaud, "Grounding image matching in 3d with mast3r," *arXiv preprint arXiv:2406.09756*, 2024. [Online]. Available: https://github.com/naver/mast3r

[244] F. R. Hogan, M. Jenkin, S. Rezaei-Shoshtari, Y. Girdhar, D. Meger, and G. Dudek, "Seeing through your skin: Recognizing objects with a novel visuotactile sensor," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2021, pp. 1218–1227. [Online]. Available: https://openaccess.thecvf.com/content/WACV2021/papers/Hogan_Seeing_Through_Your_Skin_Recognizing_Objects_With_a_Novel_Visuotactile_WACV_2021_paper.pdf

[245] J. Yin, G. M. Campbell, J. Pikul, and M. Yim, "Multimodal proximity and visuotactile sensing with a selectively transmissive soft membrane," in *2022 IEEE 5th International Conference on Soft Robotics (RoboSoft)*. IEEE, 2022, pp. 802–808. [Online]. Available: https://arxiv.org/pdf/2204.08586

[246] S. Sellán and A. Jacobson, "Neural stochastic screened poisson reconstruction," *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)*, 2023. [Online]. Available: https://www.silviasellan.com/projects/neural-stochastic-screened-psr/

[247] ——, "Stochastic poisson surface reconstruction," *ACM Transactions on Graphics*, 2022. [Online]. Available: https://www.dgp.toronto.edu/projects/stochastic-psr/

[248] G. Kopanas and G. Drettakis, "Improving nerf quality by progressive camera placement for free-viewpoint navigation," *The Eurographics Association*, 2023. [Online]. Available: https://arxiv.org/pdf/2309.00014

[249] M. Boss, R. Braun, V. Jampani, J. T. Barron, C. Liu, and H. Lensch, "Nerd: Neural reflectance decomposition from image collections," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 12 684–12 694. [Online]. Available: https://markboss.me/publication/2021-nerd/

[250] S. Zhu, S. Saito, A. Bozic, C. Aliaga, T. Darrell, and C. Lassner, "Neural relighting with subsurface scattering by learning the radiance transfer gradient," *arXiv preprint arXiv:2306.09322*, 2023. [Online]. Available: https://arxiv.org/pdf/2306.09322