Controlled illumination for perception and manipulation of Lambertian objects

Author Names Omitted for Anonymous Review. Paper-ID 23

***Abstract*—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 https://anonymousprojectsite.github.io/.**

I. Introduction

Imagine the following tasks of 1) finding a tiny screw on the ground especially when the color contrast between the screw and the background is poor or the background is complex, 2) examining the quality of stucco[1] or putty applied to a surface. To complete the first task, one can use a flashlight and illuminate a part of the floor with a low angle of incidence and look for shadow cues. For the second task, low angle of incidence light can highlight high spatial frequency surface discontinuities such as surface roughness and features like creases, while a directional source of light (e.g. sunlight throughout the day) can highlight low spatial frequency information like 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 ([2]), shape from texture([3, 4]), high performance area scanners (e.g. [5, 6]) and tactile sensing (e.g. [7, 8]) 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 adds the additional complexities of discovering correspondences between cameras and tactile sensor data and accounting for tactile sensor degradation.

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 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 – Bajcsy[9, 10] and Aloimonos et al.([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.

We divide the problem of active perception through con-

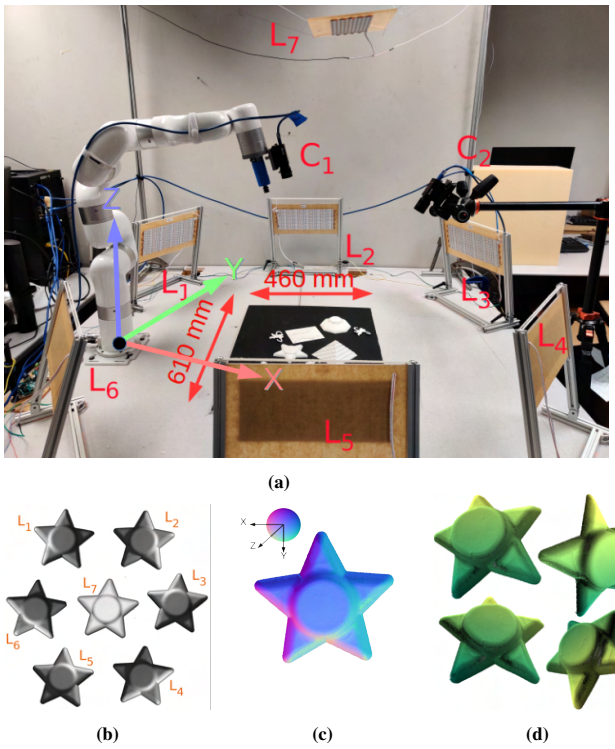


Fig. 1: We demonstrate a robot workspace scaled controlled illumination approach for manipulation. Our setup (fig. 1a) consists of 2 machine vision cameras (C_1 and C_2) overlooking a robot's workspace and the illumination of the workspace is controlled using 7 directional lights – six low angle of incidence light sources $L_{1:6}$ and one overhead light source L_7. We capture seven images of the object (fig. 1b) and use standard photometric stereo to calculate object surface normals (color coded in fig. 1c) and optionally derive a 3D representation of the object's surface (fig. 1d) from calculated normals (fig. 1c). In this initial exploration we painted objects matte white to simplify the analysis (see text).

trolled illumination into two parts – capturing and processing images of objects, and using the processed images to control robot manipulation. 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. Also, all of our objects are single rigid bodies with no articulations and whenever needed, we assume that we have 3D models and segmentation masks of objects 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 here: https://anonymousprojectsite.github.io/.

II. Related Work

Shape from shading, introduced by Horn[12] and Woodham [13], 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 ([14]), proposed accurate methods for recovering object shapes as a collection of algebraic surfaces ([15]), and more recently have demonstrated highly accurate frameworks for recovering object shape, reflectance and geometry by refining multi-view RGBD data captured by commercial sensors ([16]).

A large portion of recent work ([17, 18, 19, 20]) advocates learning implicit representations of objects from multiple views and then using them for robotics tasks ([21, 22]). There is 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 ([23]), learning implicit scene representations from multiple directionally illuminated images ([24]), estimating object geometry from shadows cast by the object under directional illumination ([25]), and reconstructing surface depth and normals from images under directional illumination ([26]). 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. Johnson et al. [27] were the first to demonstrate the use of photometric stereo as a metrically accurate sensor and smaller versions of the setup ([28, 29]) have been shown to be useful for several manipulation tasks. Our work draws inspiration from [27] as we scale up the applicability of object geometry capture to the size of a robot's workspace.

Active sensing for manipulation is also an area of active research. Adhering to the classical definition by Bajcsy, Aloimonos and colleagues ([9, 11, 10]), a significant portion of recent research on perception tasks related to manipulation can be classified as active sensing. Researchers have built active tactile object recognition systems to probe and identify objects([30, 31]), to probe and build 3D representations of objects ([32, 33]), and to localize tactile sensor interaction on object surfaces of known geometry([34]). Sensors designed specifically for active sensing have been demonstrated to actively adapt to the shape of the sensed object for yielding high resolution tactile imprints ([35, 36]) and perceive objects buried into granular media by fluidizing the media through vibration ([37, 38]). Towards fusing multiple sensors into an active sensing framework, Dikhale et al. [39] have demonstrated a combination of external cameras and hand mounted tactile sensors for estimating pose of grasped objects – with the grasped object being re-oriented to improve the pose estimates. Chaudhury et al. [40] have demonstrated fusion of RGB, RGBD and tactile sensors to visually servo a robot manipulator towards an object while maintaining a pose estimate of the object and then refining the pose estimate with the data received at contact. In this work, we draw several inspirations from the limitations of [40, 39] – specifically from the ones due to camera capabilities and sensor geometry hindering discovery of data correspondence between the sensors and leverage the strengths of controlled illumination demonstrated through the performance of the tactile sensor.

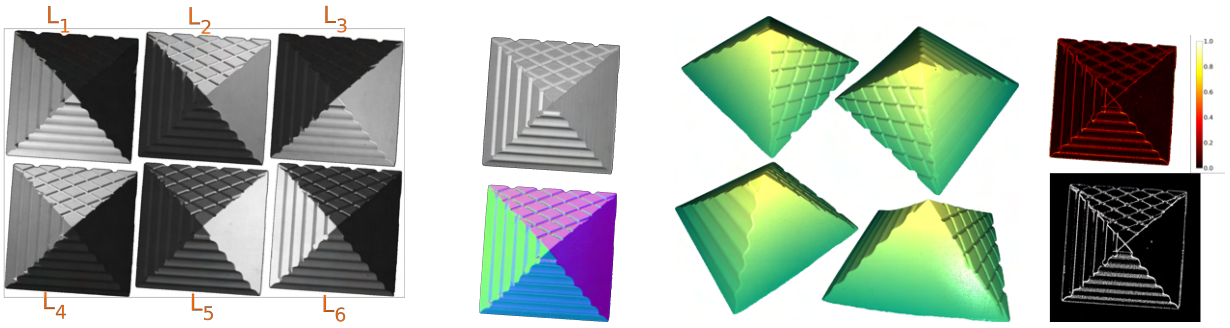
III. Methods

It is well known from the shape from shading literature (e.g. [12, 14]) that the measured intensity through an imaging device is a function of 3 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. Future work will jointly estimate object normals and reflectance functions. In this section, we discuss how we design and measure the illumination of the environment in section III-A and our method of capturing the shape of the object in the form of surface normals in section III-B. A detailed description of our hardware setup in can be found in Appendix A.

A. 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

[1] we use RustOleum 7790830 Flat White spray paint



(a) (b) (c) (d)

Fig. 2: Our pipeline for performing photometric stereo at the scale of a robot's workspace. Figure 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. 2b is the image captured using L_7 and the bottom image represents the normals obtained after solving eq. (4). The normal slopes about world's X,Y and Z axes (see fig. 1a for reference) have been mapped to red, green and blue channels respectively. Figure 2c presents the heightmaps of the object constructed by integrating the normals in fig. 2b, and, fig. 2d from top to bottom presents our setup's confidence about a pixel being a depth edge and the recovered edge map.

is also known as grazing illumination in the literature (see e.g. [27]). 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. 1a) on. The calibration object is a matte white hemishperical target with a 4cm radius.

Following Johnson et al. [27], we choose to model the illumination seen by a camera using a linear and a quadratic model. The linear illumination model, following [41], is the Lambertian reflectance equation

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

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. 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 [41], and use the Moore-Penrose operator to invert eq. (1) 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, 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. [42, 43]), for small scale micro-geometry capture systems ([27]) and for recovering shape and illumination from shading of images (e.g. [14]). We follow the formulation in [42] and can write the quadratic shading model as

$$I_i^k = \tilde{\mathbf{n}}_k^T \mathbf{M}_i \tilde{\mathbf{n}}_k \quad \forall i \in 1..., 6 \tag{2}$$

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. (2) 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.

B. 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. (1) at every pixel to get a initial estimate of the shape and then refining it using eq. (2), 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. [44, 45])

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. 1b or fig. 2a). 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 [41] we can invert a weighted version of eq. (1) per-pixel to get the initial estimates of *shape* at each pixel as:

$$\hat{\mathbf{n}}_k = (\mathbf{W}_k \mathbf{L}^k_{lin})^\dagger \mathbf{W}_k \mathbf{i}_k \tag{3}$$

where $\mathbf{L}^k_{lin} \in \mathbb{R}^{6\times 3}$ 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) is not solvable. 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). To do this, we use the previously estimated $\hat{\mathbf{n}}_k$ in eq. (2) to estimate the intensities $\hat{I}^k_i, i \in (1,...,6)$ 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 $I^k_i,\ i \in 1,...,6, \forall k$ to get a per pixel loss $\ell_k = ||I^k_i - \hat{I}_i||_2$. Finally, we iteratively solve a regularized photometric loss (eq. (4)) across all pixels and channels while updating our hypotheses of per-pixel shadow weights using eq. (5) at every step.

$$\sum_{k,i}(||I^k_i - w^i_k \hat{I}^k_i||_2) + \beta \mathcal{L}_d(\mathbf{n}_k)\ i \in (1,...,6), \forall k \tag{4}$$

$$\mathbf{w}^{t+1}_k = \left\{ \begin{array}{c} \mathbf{w}^t_k, \ \text{if } \ell^{t+1}_\mathrm{k} \leq \ell^\mathrm{t}_\mathrm{k} \\ \epsilon \max\{\epsilon, \ell^{t+1}_k\} \ \text{if } \ell^{t+1}_\mathrm{k} > \ell^\mathrm{t}_\mathrm{k} \end{array} \right\} \tag{5}$$

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 [12, 27] or smoothness priors in [14]) through a Laplacian cost $\beta\mathcal{L}_d(\cdot)$ in eq. (4). The influence of this regularizer can be controlled through the width of the Laplacian filter d and the weight β. We minimize eq. (4) using gradient descent using backtracking line search ([46]) – 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 [27], 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 Appendix F.

C. 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. 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. 2a. This observation was used by [47] to perform non-photorealistic stylized rendering of images and by [48] for detecting edges to localize fabricated parts. In this work, we modified Raskar and colleagues'([47]) 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 2d (top to bottom) shows our confidence (following [47]) about whether a pixel is at a depth edge and the edge map obtained by hysteresis thresholding (see e.g. Canny [49]) the confidence map. We provide more details in Appendix B.

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 I. 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 I summarizes our qualitative findings and for the results in this work, we used the perspective corrected Poisson integration ([50]). Resulting integrated depth maps can be seen in figs. 1d and 2c, obtained by integrating the normals in figs. 1c and 2b respectively.

However, we should note that, unlike other true depth

Method	Speed	Robust	Accurate	Shape
Variational calc. ([51, 52])	✓	×	×	✓
Fourier ([53])	✓	✓	×	×
Least sq. (IRLS) ([27, 54])	×	✓	✓	✓
Poisson Int. ([50])	✓	✓	✓	✓

TABLE I: Qualitative comparison of the different normal integration methods evaluated. 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.

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 II: 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 normals maps corresponding to the next best performing sensor have been provided in fig. 3a along with the data captured by our sensor in those categories.

sensing systems (see table II) which 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. 1d and 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.[40] and Johnson et al.[27]) 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.

IV. Experiments and Results

In this section we describe our experiments to gauge the capability of our approach as a general purpose sensing system for manipulation. We start with 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.

A. Performance of our sensing approach

In this section we discuss the performance of our approach in perception tasks related to manipulation. We 1) compare the performance of our approach to approaches using several commercial sensors and 2) demonstrate the accuracy of our sensor in measuring the surface normals and depth of known objects.

1) Quality of normals versus true depth sensors: The true 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 II), or through time-of-flight (rows 4, 5 of table II). For the true depth sensors, we calculate the normals as spatial derivatives of the captured depth images for these sensors. 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. As is standard in image comparison literature, we use the earth movers distance ([55, 56]) to compare the normals captured by our approach and the commercial sensors to the ground truth.

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 Keselman et al. [57] by projecting visible or infrared patterns on the objects as applicable. Additionally, to reduce the noise in the measurements, for the stereo sensors (rows 1 through 3 in table II), 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 II) 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 as well. We present our quantitative results in table II and our qualitative results in fig. 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.

2) Accuracy of our approach: To quantify the accuracy of our sensor in measuring object surface normals and object surface depth, we 3D printed a set of objects of footprint less than 12cm^2 using a Prusa MK3S 3D printer (we used Prusa PLA filament and a layer height of 0.15mm) and spray painted the objects matte white. The objects were then imaged with

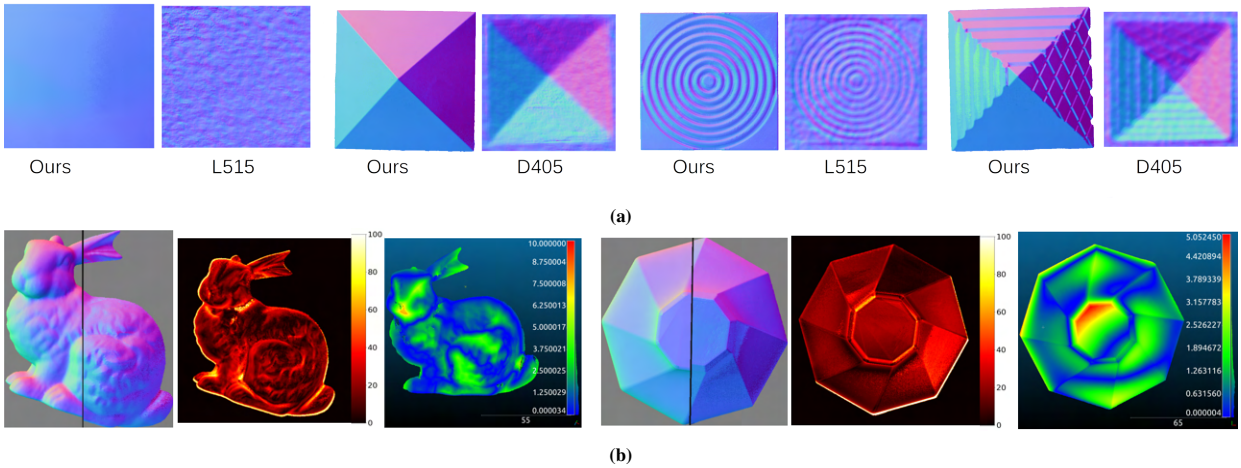


Fig. 3: Performance of our approach. In fig. 3a we compare the quality of normals computed by our sensor with the processed data from the best performing commercial depth sensor (table III). Figure 3b shows the normal and surface quality of the objects bunny and octagon II from table III as sets of 3 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.

	Normal quality ($\triangle^\circ$)		Surface quality ($\triangle$mm)	
Object name	$\mu(\sigma)$	$< 20^\circ$ (%)	$\mu(\sigma)$	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 III: Quantitative measurements of the accuracy of our approach in measuring object surface normals and generating metric surface measurements of the surface. The measurements under normal quality denote the deviations of measured normals from ground truth normals in degrees, The measurements under the surface quality denote the deviation of the reconstructed surface from the ground truth mesh in mm. Please visit the project website for qualitative visualization of the results.

our setup in fig. 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 the procedure described in section III-B to calculate the normals and the method described in section IV-D to estimate the pose of the objects. 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 IV-D1) aligned the simulated and the real data up to a maximum error of 5 pixels, making the per pixel error calculation meaningful. In table III 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 IV-B) 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. 3b), the quality of our normal and depth estimates are similar to classical methods ([14, 15]) and are marginally worse than recent deep learning based methods ([26, 23, 24]).

To measure the surface quality, we generated a surface depth map from the calculated normals using the method described in section III-C. We then registered the integrated depth map to the ground truth mesh using point-to-plane ICP ([58]) and calculated the Hausdorff distance ([59]) 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 III. 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. 6a). We present a qualitative result in fig. 3b. More qualitative results can be viewed on our project website.

B. Controlled illumination for pickup tasks

To perform immobilizing grasps with a single-point vacuum gripper, we need to identify a portion of the object that is large enough for the gripper to fit and flat enough for a vacuum gripper 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 direction of perpendicular to the surface normal to 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. 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.

Picking up 3D Lambertian objects: 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 ([60]). We describe our algorithm for identifying corresponding flat faces on objects in the robot's workspace across the two views C_1 and C_2 (see fig. 1a) in Appendix C.

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 grasp location 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 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.

Picking up thin objects without color contrast: Following Raskar et al. [47], we observe that image brightness variations due to occlusion are more prominent than brightness differences due to color texture. We use this intuition for picking up thin flat objects from the background is difficult. For this experiment we attempted to 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. 4c and 4d, there isn't enough intensity or color texture to segment the object from the background. However, looking along the light direction (see fig. 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. 4c and 4d.

This shadow cue was sufficient for our edge detection procedure (see section III-C) to identify occlusion edge pixels in the scene. We then estimated the center of the disc by fitting a minimum enclosing circle ([61]) to the occlusion edge pixels across the two views C_1 and C_2. We visualize this in fig. 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. 1a). However, in the presence of surface textures, we could not reliably detect normals on the surface of the object, so we assume that the disc lies flat on the workspace. To pick up the disc, we approached the center of the disc along the surface normal o the tabletop by triangulating the object from the two views in fig. 4d. None of the commercial depth sensors we used (see table II) detected the disc reliably.

With the procedure described in this section, we performed 35 grasp experiments with 3D objects and ten experiments with flat textured 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 (of 35) while picking up the textured pyramid (fig. 4a) due to the vacuum gripper failing to attach or colliding with the object due to triangulation errors.

C. Controlled illumination for measuring deformation

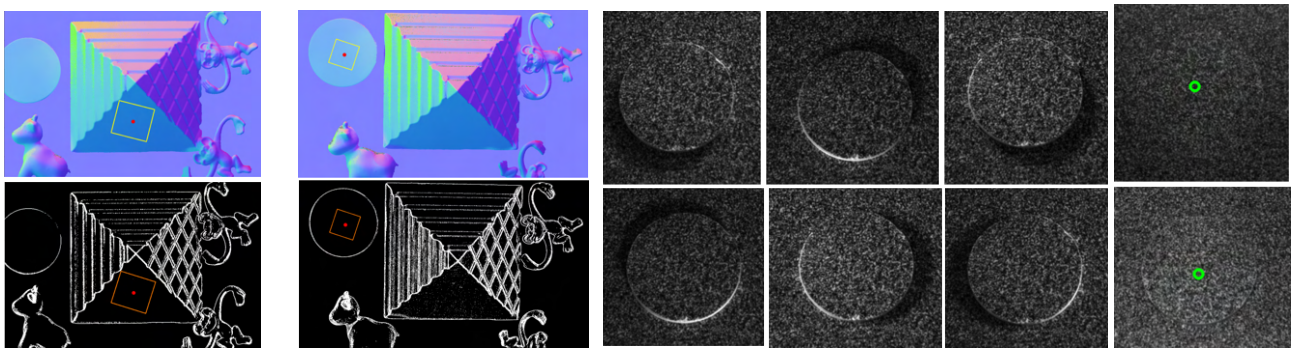
Our approach can be used to perceive the deformation of objects by tracking the change of local normals on the object's surface. In this section, we describe our method for measuring the buckling deformation of a rectangular PVC card of size 86mm×54mm. We measure the deformation in a typical analysis-by-synthesis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

Figure 5b shows a qualitative result of our pipeline – the first image shows the reconstructed 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. 5b, significantly deteriorated 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.

To verify that we indeed captured the physical process behind the card buckling, we perform the following experiment. We note that any vertical section along the longest dimension of the card can be modeled as a fixed-pinned beam with the following equation

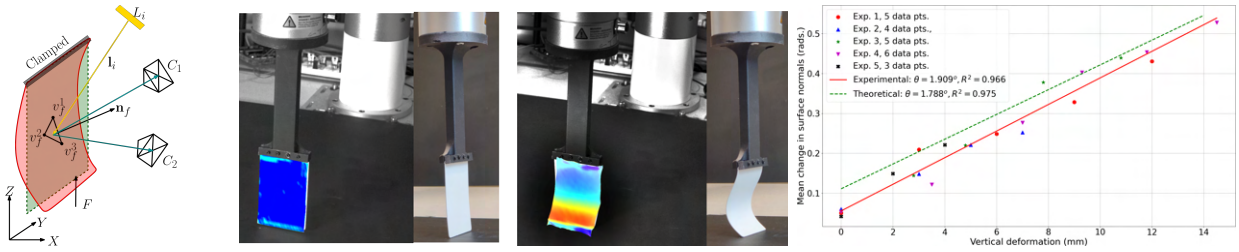
$$y(x) = \delta_{max}\left(1 - \cos\frac{\pi x}{0.7(L_{max} - p)}\right) \tag{6}$$

where δ_{max} denotes the maximum deformation at any portion of the card, L_{max} is the maximum length of the card (86mm in our case) and, p is the deformation of the card along the Z direction in fig. 5a. As the card is pushed down, the card buckles and the value of p increases, but L_{max} remains constant at 86mm. We solve for δ_{max} for 15 values of p ranging from 0mm (no deformation) to 14mm. We uniformly sample each of the 15 curves generated by substituting the values of p and δ_{max} into eq. (6) at 35 points along the theoretical beam's length. We chose the 35 points to align with the 35 vertices along the 86mm edge of $\mathcal{M}$. We then calculated the mean of the change in normals at each sampled point between the undeformed and the deformed curve. Next, we calculated the surface normals of the deformed card using the pipeline discussed above and compared the mean change in surface normal orientation from the undeformed configuration



(a) (b) (c) (d)

Fig. 4: Our pipeline for grasping objects in a robot's workspace. Figures 4a and 4b respectively show the grasps selected by the camera C_1 along X and Z directions respectively. On the top rows of figs. 4a and 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 4c and 4d shows our pipeline for picking up thin objects in absence of obvious depth or texture segmentation cues. Figure 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 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)

Fig. 5: Our results of visually measuring the deformation of known objects.Figure 5a shows a schematic diagram of our procedure, fig. 5b shows the estimated shape of the card overlaid on a camera image. The insets of the images in fig. 5b provide an external view of the state of the cards. Figure 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.

Method	Scale	Error $\mu(\sigma)$	Max	Sample size
Ours	86×54	0.78 (0.52)	4.57	100k
VIRDO ([62])	$89 \times 56^*$	1.148 (–)	–	5.6k
VIRDO++ ([63])	$89 \times 56^*$	1.041 (0.348)	–	5.6k

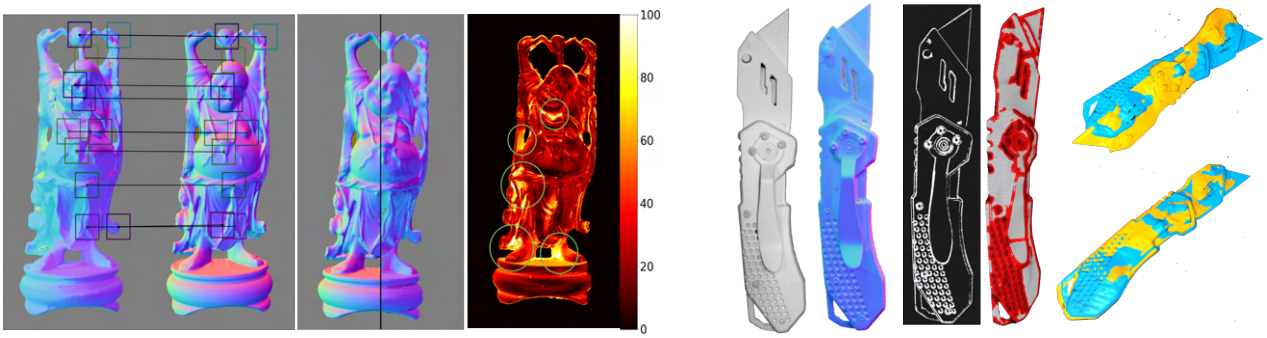
TABLE IV: Comparison of our method of measuring deformation (Chamfer Distance) with Wi et al.([62, 63]). Our experiment was carried out in simulation because we do not have access to a sensor accurate enough to measure ground truth deformations, and we are comparing the performance of Wi and colleagues' results during inference. 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. (*)We assume Wi et al. use typical spatulas in their work. All length measurements in mm.

of the card in fig. 5c.

The theoretical data showed a linear trend between the mean change in normals and the vertical deformation of the card. A line with a slope of 1.78°, 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 and obtained 23 data points as shown in fig. 5c with five different markers corresponding to the experiments. The experimental data too 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 violation of the boundary condition due to slipping of the card at the base, from fig. 5c we can conclude that our pipeline is repeatable and can approximately 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. ([62, 63]) 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 IV. 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 [63].

We also note that our "analysis-by-synthesis" procedure works better than integrating the captured normal map (section III-C) 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 camera 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.



(a) (b)

Fig. 6: Our pipelines for estimating object pose in the robot workspace. Figure 6a shows our steps in estimating object poses when a 3D model of the object is available apriori. From left to right in fig. 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 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 point-features overlaid on the points in red and, initial and final stages of our estimation of the object's pose.

D. Controlled illumination for localization

In this section, we describe our results for estimating object poses from data captured by our system.

1) 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 commonly employed pipelines in the pose estimation literature (see e.g. [40, 48, 64]) by generating good initial pose estimates through discovery of object patch correspondences between the observed and simulated surface normals (inset 1 of fig. 6a). We describe our method in detail in Appendix E.

We present a qualitative result in fig. 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. 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. 6a has $\sim$50% of pixels with erroneous normals ($> 20^\circ$ deviation from ground truth).

We present quantitative measurements of our pose estimation approach for all the objects in table III in table V. Our experiments were done with C_1 imaging a 227mm $\times$ 129mm area at a resolution of 7 pixels/mm. We placed each of the objects inside the imaging area 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 V.

Object	Bounding Box in mm	Errors $\mu(\sigma)$ in px.
pyramid	$100 \times 100 \times 55$	2.50(1.70)
star	$110 \times 110 \times 15$	3.62(1.03)
bent cylinder	$115.5 \times 48 \times 12.5$	5.50(2.12)
octagon I	$100 \times 100 \times 28.5$	3.35(0.88)
octagon II	$100 \times 100 \times 28.5$	2.85(1.83)
spot	$52 \times 14.5 \times 52.5$	1.5 (0.23)
bas-relief	$83 \times 110.0 \times 6.52$	0.8 (0.2)
bunny	$93 \times 93.5 \times 42.5$	4.06(1.19)
text pyramid	$100 \times 100 \times 55$	4.02(0.96)
happy Buddha	$115 \times 45 \times 36.5$	3.1(0.69)

TABLE V: Performance of our pose estimation pipeline for known objects. All the experiments were done with a resolution of $\sim$7 pixels/mm. Qualitative results for the objects happy Buddha, bunny and octagon II can be found in fig. 6a and the first and fourth insets of fig. 3b respectively. More qualitative results can be found on the project website.

2) 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 generated by the process described in section III-C. 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. 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. 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 Appendix E.

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

Object	Bounding Box (mm)	$\triangle X$(mm)	$\triangle Y$(mm)	$\triangle\theta(^\circ)$
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	$(12.5 \times 20.5 \times 33)$	3.28	2.40	1.28
IO shield	$(120 \times 39 \times 21)$	1.48	1.86	0.25

TABLE VI: 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. 6b, to view results of the other objects, please visit the project website.

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 VI and report our pipeline's uncertainty in recovering the pose perturbations. We present our quantitative results in table VI – 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 IV-D1 actually led to poorer pose estimation because the successive processing steps (normal calculation, integration, and meshing) deteriorated the quality of the mesh input. Also, the fact that the 3D representation generated is strongly dependent on the viewpoint of the camera during the experiment meant that the 3D model was metrically incorrect along views other than the one used to generate it, thus making the resulting mesh unfit for the previous pipeline.

V. 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. We envision combining a neural representation for reflectances with our physically based method for capturing shape to extend our method to general objects. 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. 3b, and third inset of fig. 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. The coverage of a pixel is dependent on the resolution of the camera and the relative pose of the camera and the object of interest, and we observed that the accuracy of our methods fall sharply when the image resolution was decreased. Current literature indicates that a triangle-based representation ([16]) or locally smooth patch-based representations ([15]) 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 ([17, 20]) 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. Finally, the question of how much fidelity of measurement is needed for manipulation remains unanswered across the present literature. In this work, we achieve a higher fidelity of measurement than some commercial depth sensors albeit by imposing restrictive priors on object reflectances. A natural extension of the current work is to see if we can augment the performance of a commercial depth sensor by using it in conjunction with our controlled illumination approach.

VI. Conclusions

In this work, we presented a robot workspace scaled controlled illumination system for the manipulation of objects with Lambertian reflectances. We showed that, by enforcing a reflectance prior on the objects, reasoning about observed object intensities conditioned on the direction of illumination can yield reasonably 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 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.

References

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

[2] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise View Selection for Unstructured Multi-View Stereo. In *European Conference on Computer Vision (ECCV)*, 2016.

[3] John Aloimonos. Shape from texture. *Biological Cybernetics*, 58(5): 345–360, 1988.

[4] Dor Verbin and Todd Zickler. Toward a Universal Model for Shape From Texture. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[5] Photoneo PhoXi3D scanners. URL https://www.photoneo.com/phoxi-3d-scanner/.

[6] Ensenso XR series scanners. URL https://www.ids-imaging.us/ensenso-3d-camera-xr-series.html.

[7] Wenzhen Yuan, Siyuan Dong, and Edward H Adelson. Gelsight: High-resolution robot tactile sensors for estimating geometry and force. *Sensors*, 17(12):2762, 2017.

[8] Elliott Donlon, Siyuan Dong, Melody Liu, Jianhua Li, Edward Adelson, and Alberto Rodriguez. Gelslim: A high-resolution, compact, robust, and calibrated tactile-sensing finger. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1927–1934. IEEE, 2018.

[9] Ruzena Bajcsy. Active perception. *Proceedings of the IEEE*, 76(8): 966–1005, 1988.

[10] Ruzena Bajcsy, Yiannis Aloimonos, and John K Tsotsos. Revisiting active perception. *Autonomous Robots*, 42(2):177–196, 2018.

[11] John Aloimonos, Isaac Weiss, and Amit Bandyopadhyay. Active vision. *International Journal of Computer Vision*, 1(4):333–356, 1988.

[12] Berthold KP Horn. Shape from shading: A method for obtaining the shape of a smooth opaque object from one view. 1970.

[13] Robert J Woodham. Photometric method for determining surface orientation from multiple images. *Optical Engineering*, 19(1):139–144, 1980.

[14] Jonathan T Barron and Jitendra Malik. Shape, illumination, and reflectance from shading. *IEEE Transactions on Pattern Analysis and Machine Isntelligence*, 37(8):1670–1687, 2014.

[15] Ying Xiong, Ayan Chakrabarti, Ronen Basri, Steven J Gortler, David W Jacobs, and Todd Zickler. From shading to local shape. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(1):67–79, 2014.

[16] Fujun Luan, Shuang Zhao, Kavita Bala, and Zhao Dong. Unified shape and svbrdf recovery using differentiable Monte Carlo rendering. In *Computer Graphics Forum*, volume 40, pages 101–113. Wiley Online Library, 2021.

[17] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

[18] Mark Boss, Raphael Braun, Varun Jampani, Jonathan T Barron, Ce Liu, and Hendrik Lensch. NeRD: Neural reflectance decomposition from image collections. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12684–12694, 2021.

[19] Pratul P Srinivasan, Boyang Deng, Xiuming Zhang, Matthew Tancik, Ben Mildenhall, and Jonathan 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*, pages 7495–7504, 2021.

[20] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance Fields Without Neural Networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5501–5510, 2022.

[21] Lin Yen-Chen, Pete Florence, Jonathan T Barron, Tsung-Yi Lin, Alberto Rodriguez, and Phillip Isola. NeRF-Supervision: Learning Dense Object Descriptors from Neural Radiance Fields. *arXiv preprint arXiv:2203.01913*, 2022.

[22] Lin Yen-Chen, Pete Florence, Jonathan T Barron, Alberto Rodriguez, Phillip Isola, and Tsung-Yi Lin. iNeRF: Inverting neural radiance fields for pose estimation. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1323–1330. IEEE, 2021.

[23] Wenqi Yang, Guanying Chen, Chaofeng Chen, Zhenfang Chen, and Kwan-Yee K Wong. PS-NeRF: Neural inverse rendering for multi-view photometric stereo. In *European Conference on Computer Vision*, pages 266–284. Springer, 2022.

[24] Wenqi Yang, Guanying Chen, Chaofeng Chen, Zhenfang Chen, and Kwan-Yee K Wong. S3-NeRF: Neural Reflectance Field from Shading and Shadow under a Single Viewpoint. *arXiv preprint arXiv:2210.08936*, 2022.

[25] Kushagra Tiwary, Tzofi Klinghoffer, and Ramesh Raskar. Towards learning neural representations from shadows. *arXiv preprint arXiv:2203.15946*, 2022.

[26] Yacov Hel-Or Asaf Karnieli, Ohad Fried. DeepShadow: Neural shape from shadows. In *ECCV*, 2022.

[27] Micah K Johnson, Forrester Cole, Alvin Raj, and Edward H Adelson. Microgeometry capture using an elastomeric sensor. *ACM Transactions on Graphics (TOG)*, 30(4):1–8, 2011.

[28] GelSight Inc. GelSight Mobile, . [Online; accessed 09-Dec-2022].

[29] GelSight Inc. GelSight Mini, . [Online; accessed 09-Dec-2022].

[30] Alexander Schneider, Jürgen Sturm, Cyrill Stachniss, Marco Reisert, Hans Burkhardt, and Wolfram Burgard. Object identification with tactile sensors using bag-of-features. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 243–248. IEEE, 2009.

[31] Shan Luo, Wenxuan Mou, Kaspar Althoefer, and Hongbin Liu. Novel tactile-SIFT descriptor for object shape recognition. *IEEE Sensors Journal*, 15(9):5001–5009, 2015.

[32] Sudharshan Suresh, Zilin Si, Joshua G Mangelson, Wenzhen Yuan, and Michael Kaess. ShapeMap 3-D: Efficient shape mapping through dense touch and vision. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 7073–7080. IEEE, 2022.

[33] Maria Bauza, Oleguer Canal, and Alberto Rodriguez. Tactile mapping and localization from high-resolution tactile imprints. In *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019.

[34] Sudharshan Suresh, Zilin Si, Stuart Anderson, Michael Kaess, and Mustafa Mukadam. MidasTouch: Monte-Carlo inference over distributions across sliding touch. *arXiv preprint arXiv:2210.14210*, 2022.

[35] Sandra Q Liu and Edward H Adelson. GelSight Fin Ray: Incorporating Tactile Sensing into a Soft Compliant Robotic Gripper. In *2022 IEEE 5th International Conference on Soft Robotics (RoboSoft)*, pages 925–931. IEEE, 2022.

[36] Yu She, Sandra Q Liu, Peiyu Yu, and Edward Adelson. Exoskeleton-covered soft finger with vision-based proprioception and tactile sensing. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10075–10081. IEEE, 2020.

[37] Radhen Patel, Rui Ouyang, Branden Romero, and Edward Adelson. Digger finger: Gelsight tactile sensor for object identification inside granular media. In *International Symposium on Experimental Robotics*, pages 105–115. Springer, 2020.

[38] Togzhan Syrymova, Yerkebulan Massalim, Yerbolat Khassanov, and Zhanat Kappassov. Vibro-Tactile Foreign Body Detection in Granular Objects based on Squeeze-Induced Mechanical Vibrations. In *2020 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 175–180. IEEE, 2020.

[39] Snehal Dikhale, Karankumar Patel, Daksh Dhingra, Itoshi Naramura, Akinobu Hayashi, Soshi Iba, and Nawid Jamali. VisuoTactile 6D Pose Estimation of an In-Hand Object using Vision and Tactile Sensor Data. *IEEE Robotics and Automation Letters*, 2022.

[40] Arkadeep Narayan Chaudhury, Timothy Man, Wenzhen Yuan, and Christopher G Atkeson. Using Collocated Vision and Tactile Sensors for Visual Servoing and Localization. *IEEE Robotics and Automation Letters*, 7(2):3427–3434, 2022.

[41] Svetlana Barsky and Maria 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*, 25(10):1239–1252, 2003.

[42] Ravi Ramamoorthi and Pat Hanrahan. An efficient representation for irradiance environment maps. In *Proceedings of the 28th annual conference on Computer graphics and Interactive Techniques*, pages 497–500, 2001.

[43] Ronen Basri and David W Jacobs. Lambertian reflectance and linear subspaces. *IEEE transactions on pattern analysis and machine intelligence*, 25(2):218–233, 2003.

[44] Marshall Tappen, William Freeman, and Edward Adelson. Recovering intrinsic images from a single image. *Advances in Neural Information Processing Systems*, 15, 2002.

[45] Carsten Rother, Martin Kiefel, Lumin Zhang, Bernhard Schölkopf, and Peter Gehler. Recovering intrinsic images with a global sparsity prior on reflectance. *Advances in Neural Information Processing Systems*, 24, 2011.

[46] Stephen Wright, Jorge Nocedal, et al. Numerical optimization. *Springer Science*, 35(67-68):7, 1999.

[47] Ramesh Raskar, Kar-Han Tan, Rogerio Feris, Jingyi Yu, and Matthew Turk. Non-photorealistic camera: depth edge detection and stylized rendering using multi-flash imaging. *ACM Transactions on Graphics (TOG)*, 23(3):679–688, 2004.

[48] Ming-Yu Liu, Oncel Tuzel, Ashok Veeraraghavan, Yuichi Taguchi, Tim K Marks, and Rama Chellappa. Fast object localization and pose estimation in heavy clutter for robotic bin picking. *The International Journal of Robotics Research*, 31(8):951–973, 2012.

[49] John Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (6):679–698, 1986.

[50] Y. Quéau, J.-D. Durou, and J.-F. Aujol. Normal Integration: A Survey. *Journal of Mathematical Imaging and Vision*, 60(4):576–593, 2018. doi: 10.1007/s10851-017-0773-x.

[51] Berthold KP Horn and Michael J Brooks. The variational approach to shape from shading. *Computer Vision, Graphics, and Image Processing*, 33(2):174–208, 1986.

[52] David Eberly. Reconstructing a Height Field from a Normal Map. *Geometric Tools, Redmond WA 98052. USA*. URL https://www.geometrictools.com/.

[53] Robert T. Frankot and Rama Chellappa. A method for enforcing integrability in shape from shading algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(4):439–451, 1988.

[54] Xu Cao, Hiroaki Santo, Boxin Shi, Fumio Okura, and Yasuyuki Matsushita. Bilateral normal integration. In *European Conference on Computer Vision*, pages 552–567. Springer, 2022.

[55] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. The earth mover's distance as a metric for image retrieval. *International Journal of*

Computer Vision, 40(2):99–121, 2000.
[56] Haibin Ling and Kazunori Okada. An efficient earth mover's distance algorithm for robust histogram comparison. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(5):840–853, 2007.
[57] Leonid Keselman, John Iselin Woodfill, Anders Grunnet-Jepsen, and Achintya Bhowmik. Intel realsense stereoscopic depth cameras. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–10, 2017.
[58] Szymon Rusinkiewicz and Marc Levoy. Efficient variants of the ICP algorithm. In *Proceedings of the Third International Conference on 3-D Digital Imaging and Modeling*, pages 145–152. IEEE, 2001.
[59] Paolo Cignoni, Claudio Rocchini, and Roberto Scopigno. Metro: measuring error on simplified surfaces. In *Computer Graphics Forum*, volume 17, pages 167–174. Wiley Online Library, 1998.
[60] Gary R Bradski. Computer vision face tracking for use in a perceptual user interface. 1998.
[61] OpenCV. minEnclosingCircle(): Open Source Computer Vision Library, 2022.
[62] Youngsun Wi, Pete Florence, Andy Zeng, and Nima Fazeli. VIRDO: Visio-tactile implicit representations of deformable objects. *arXiv preprint arXiv:2202.00868*, 2022.
[63] Youngsun Wi, Andy Zeng, Pete Florence, and Nima Fazeli. VIRDO++: Real-World, Visuo-tactile Dynamics and Perception of Deformable Objects. *arXiv preprint arXiv:2210.03701*, 2022.
[64] Marco Imperoli and Alberto 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.
[65] Centric daylight led strip lights for commercial & retail. URL https://store.waveformlighting.com/collections/led-strips/products/ultra-high-95-cri-led-strip-lights-for-commercial?variant=12104387919974.
[66] Grasshopper3 USB3 models: GS3-U3-41C6M-C and GS3-U3-89S6M-C. URL https://www.flir.com/products/grasshopper3-usb3.
[67] C series fixed focal length lenses: Edmund Optics. URL https://www.edmundoptics.com/f/c-series-fixed-focal-length-lenses/13679/.
[68] Ufactory xArm 7. URL https://www.ufactory.cc/product-page/ufactory-xarm-7.
[69] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3d deep learning with PyTorch3D. *arXiv preprint arXiv:2007.08501*, 2020.
[70] Pedro F Felzenszwalb and Daniel P Huttenlocher. Distance transforms of sampled functions. *Theory of Computing*, pages 415–428, 2012.
[71] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (FPFH) for 3D registration. In *2009 IEEE International Conference on Robotics and Automation*, pages 3212–3217. IEEE, 2009.
[72] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Fast global registration. In *European Conference on Computer Vision*, pages 766–782. Springer, 2016.
[73] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3D: A modern library for 3D data processing. *arXiv preprint arXiv:1801.09847*, 2018.

APPENDIX A
DETAILS OF THE HARDWARE PLATFORM

Our controlled illumination experimental setup (fig. 1a) consists of seven 25cm×10cm light panels fabricated from commercially available LED strips[65] fixed in the robot workspace as shown in fig. 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 III-B). 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™. 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. 1a) [66]. The camera C_1 is fitted with a 12mm focal length lens and camera C_2 is fitted with a 16mm focal length lens[67]. 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[68] 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.

APPENDIX B
DETAILS OF OUR STEPS FOR CALCULATING OCCLUSION EDGES

For detecting occlusion edges we adapt the algorithm described in Raskar et al. [47] 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. 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 [47] we calculate the ratio images in eq. (7), making adjustments to avoid numerical errors.

$$\mathbf{R}_{1:6} = \frac{\mathbf{I}_{1:6}}{\mathbf{I}_7} \tag{7}$$

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. 2d). Finally following [49], we apply hysteresis thresholding to the confidence map and obtain a binary map of occlusion edges (bottom image of fig. 2d).

APPENDIX C
DESCRIPTION OF OUR PICKUP ALGORITHM

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. 4a and 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. 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 of 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 [60]. This step identifies a candidate zone for our grasp. The top rows of figs. 4a and 4b demonstrate the output of this step, projected onto the normals of the scene calculated using the method described in section III-B.
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 III-C 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. 4a and 4b identify the selected grasp in areas without surface textures projected on the object depth edges calculated by our method in section III-C.

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. 1a for reference) and score the detected patches. We identify the highest-scoring patches as flat and "pickable". Figures 4a and 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.

APPENDIX D
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[69]) 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. 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. 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 III-B. 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:
$$\ell_{C_1}(\mathcal{M}) = \sum_k \left[1 - \langle \mathbf{N}_S^{C_1}, \mathbf{N}_S^{C_1} \rangle\right] + |\mathbf{M}_S^{C_1} - \mathbf{M}_R^{C_1}|^2 \tag{8}$$
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
$$\ell(\mathcal{M}) = \alpha \ell_{C_1}(\mathcal{M}) + (1-\alpha)\ell_{C_2}(\mathcal{M}) \tag{9}$$
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. (9) 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 Luan et al. [16] 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. (9).

5) Finally, we formulate our objective function as:

$$\min_{v_i} \ell(\mathcal{M}) + L_{Lap.}(\mathcal{M}) + L_{edge}(\mathcal{M}) \ \forall v_i \in \mathcal{M} \quad (10)$$

We use gradient descent with backtracking line search to minimize eq. (10) above.

APPENDIX E
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[69] for our work. Using the method described in sections III-B and III-C 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. 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

$$\min_{x,y,\theta} \sum_k \left| \mathbf{M}_S^k - \mathbf{M}_R^k|_{(x,y,\theta)} \right|_2^2 \quad (11)$$

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 ${}^i\mathbf{M}_S$:

$$\min_{x,y,\theta} \sum_{k \in {}^i\mathbf{M}_S} \left[1 - \langle {}^i\mathbf{N}_S^k, {}^i\mathbf{N}_R^k|_{(x,y,\theta)} \rangle \right] \forall i \quad (12)$$

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

4) Finally, following Chaudhury et al.[40] we align the rendered and measured depth edge images $\mathbf{E}_S$ and $\mathbf{E}_R$ by minimizing a pixel-wise sum of absolute differences between the Euclidean distance transforms (EDT) of images $\mathbf{E}_S$ and $\mathbf{E}_R$. Using the definition of Euclidean distance transform from [70] in we formulate our dense edge alignment cost as

$$\min_{x,y,\theta} \sum_k \left| \mathrm{EDT}(\mathbf{E}_\mathrm{S}) - \mathrm{EDT}(\mathbf{E}_\mathrm{R}|_{(\mathrm{x,y},\theta)}) \right|_2^2 \quad (13)$$

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. 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. 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) [71]. The fourth inset of fig. 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. 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 [72]) to generate initial pose estimates and then use point-to-plane ICP [58] to solve for the change in pose of the object between the two measurements. The last two insets of fig. 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.

Step	Pre-processing	Processing
Capture (fig. 1b)	5.0 (serial init.)	0.20 (per cam.)
Normal (eq. 1,2)	0.8 (GPU malloc)	3.5 (1000×1000px)
Depth (section III-C)	0.05 (GPU → host)	0.15 (1000×1000px)
Edges (section III-C)	–	0.05 (1000×1000px)
Pickup (fig. 4a)	–	0.5 (1000×1000px)
Pickup (robot)	–	20 (pick and drop)
bending (eq. (10))	5 (capture+ GPU init)	15 (1500 vert, 300 iter))
pose est. (eq. 11, 12, 13)	5 (1.5M vert.)	12.5 - 15.5 (150 iter/eq.)

TABLE VII: Approximate time breakdown of various steps in seconds

APPENDIX F
IMPLEMENTATION DETAILS AND HYPERPARAMETERS

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. (4)), bending estimation (eq. (10)) and all the alignment costs (eqs. (11) to (13)) have been implemented with custom GPU backends, 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 VII.

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 consistent images. 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, reflectivity of the ceiling and walls, and the color of the wall.
- We used backtracking line search for all the gradient descent steps (eqs. (2) and (10) to (13)) 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.
- For estimating bending deformation, we found that locating the cameras (C_1 and C_2 in fig. 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. 5c, we made sure that the value of α in eq. (9) was in between 0.45 to 0.55.
- Finally, for the global registration step in fig. 6b, we used a FPFH size of 35, and a voxel downsample factor of 1.5 in the implementation available on Open3D ([73]).