Supplementary Material: Shape from Shading for Robotic Manipulation

Arkadeep Narayan Chaudhury Leonid Keselman
Christopher G. Atkeson
The Robotics Institute, Carnegie Mellon University
5000 Forbes Avenue, Pittsburgh, Pennsylvania 15213, USA
{arkadeepnc, leonidk, cga}@cmu.edu

The paper website with a narrated summary of the research and interactive demonstrations can be accessed on https://arkadeepnc.github.io/projects/active_workspace/index.html

1. Methods

It is well known from the shape from shading literature (e.g. [5, 15]) 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 Sec. 1.2 and our method of capturing the shape of the object in the form of surface normals in Sec. 1.3. Se start with a detailed description of our hardware setup.

1.1. Details of the hardware platform

Our controlled illumination experimental setup (Fig. 1a)[1] consists of seven 25cm×10cm light panels fabricated from commercially available LED strips [1] 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 Sec. 1.3). 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. 1a) [3]. The camera C_1 is fitted with a 12mm focal length lens and camera C_2 is fitted with a 16mm focal length lens [2]. 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 [4] 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.

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

1.2. 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. [19]). 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 [19], we choose to model the illumination seen by a camera using a linear and a quadratic model. The linear illumination model, following [6], is the Lambertian reflectance equation

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

where I_i^k is the intensity of the k^{th} pixel, given the i^{th} light of $L_{1,\ldots,6}$ 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

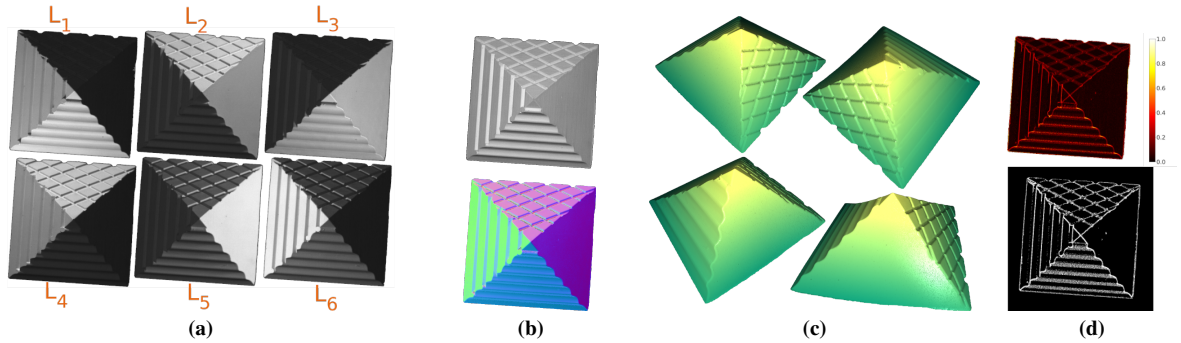

Figure 7. Our pipeline for performing photometric stereo at the scale of a robot's workspace. Figure 7a displays the images captured by the camera C_2 using each illumination source – L_1 through L_6 with the background automatically removed. The top image of Fig. 7b is the image captured using L_7 and 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 7c presents the heightmaps of the object constructed by integrating the normals in Fig. 7b, and, Fig. 7d from top to bottom presents our setup's confidence about a pixel being a depth edge and the recovered edge map.

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 [6], 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 (e.g. light from illumination channels L_1, L_2, L_3 bouncing off the wall (see Fig. 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. [7,23]), for small scale micro-geometry capture systems ([19]) and for recovering shape and illumination from shading of images (e.g. [5]). We follow the formulation in [23] 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.

1.3. 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. (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. [26,29]) 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). 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 direc-

tional sources ($L_{1:6}$) and augment it to get a binary diagonal shadow-illumination matrix $\mathbf{W}_k = \text{diag}(\mathbf{w}_k)$. Using $\mathbf{W}$ and following [6] we can invert a weighted version of Eq. (1) per-pixel to get the initial estimates of *shape* at each pixel as:

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

where $\mathbf{L}_{lin}^k \in \mathbb{R}^{6\times 3}$ is the concatenated illumination matrix at the world location of the k^{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^{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. 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) 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). To do this, we use the previously estimated $\hat{\mathbf{n}}_k$ in Eq. (2) 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. (4)) across all pixels and channels while updating our hypotheses of per-pixel shadow weights using Eq. (5) at every step.

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

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

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 [15, 19] or smoothness priors in [5]) 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 ([30]) – 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 [19], 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 Sec. 6.

1.4. 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. 7a – 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. 7a. This observation was used by [24] to perform non-photorealistic stylized rendering of images and by [20] for detecting edges to localize fabricated parts. In this work, we modified Raskar and colleagues'([24]) 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 7d (top to bottom) shows our confidence (following [24]) about whether a pixel is at a depth edge and the edge map obtained by hysteresis thresholding (see e.g. Canny [9]) the confidence map. We provide more details in Sec. 2.

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 Tab. 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 1 summarizes our qualitative findings and for the results in this work, we used the perspective corrected Poisson integration ([22]). Resulting integrated depth maps can be seen in Figs. 1d and 7c, obtained by integrating the normals in Figs. 1c and 7b respectively.

However, we should note that, unlike other true depth sensing systems (see Tab. 1) which directly measure depth at every pixel of the imaged scene using stereo or time-of-flight sensing, the integrated heightmaps are implicit sur-

Method	Speed	Robust	Accurate	Shape
Variational calc. ([12, 16])	✓	×	×	✓
Fourier ([14])	✓	✓	×	×
Least sq. (IRLS) ([10, 19])	×	✓	✓	✓
Poisson Int. ([22])	✓	✓	✓	✓

Table 1. 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.

faces 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 7c. 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. [11] and Johnson et al. [19]) 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.

2. Details of our steps for calculating occlusion edges

For detecting occlusion edges we adapt the algorithm described in [24] 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 [24] we calculate the ratio images in Eq. (6), making adjustments to avoid numerical errors.

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

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. 7d). Finally following [9], we apply hysteresis thresholding to the confidence map and obtain a binary map of occlusion edges (bottom image of Fig. 7d).

3. 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. 3a and 3b 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 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 [8]. This step identifies a candidate zone for our grasp. The top rows of Figs. 3a and 3b demonstrate the output of this step, projected onto the normals of the scene calculated using the method described in Sec. 1.3.

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 Sec. 1.4 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. 3a and 3b identify the selected grasp in

areas without surface textures projected on the object depth edges calculated by our method in Sec. 1.4.

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. 1a for reference) and score the detected patches. We identify the highest-scoring patches as flat and "pickable". Figures 3a and 3b 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.

4. 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 [25]) 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. 4a) to induce buckling, and with our vision system, we measure the change in curvature on the object and capture its deformed geometry. From Fig. 4a, 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 Sec. 1.3. 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:

$$\ell_{C_1}(\mathcal{M}) = \sum_k \left[1 - \langle \mathbf{N}_S^{C_1}, \mathbf{N}_R^{C_1} \rangle\right] + |\mathbf{M}_S^{C_1} - \mathbf{M}_R^{C_1}|^2 \tag{7}$$

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

$$\ell(\mathcal{M}) = \alpha \ell_{C_1}(\mathcal{M}) + (1-\alpha)\ell_{C_2}(\mathcal{M}) \tag{8}$$

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. (8) 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 [21] 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. (8).

5. Finally, we formulate our objective function as:

$$\min_{v_i} \ell(\mathcal{M}) + L_{Lap.}(\mathcal{M}) + L_{edge}(\mathcal{M})\ \forall v_i \in \mathcal{M} \tag{9}$$

We use gradient descent with backtracking line search to minimize Eq. (9) above.

5. 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 [25] for our work. Using the method described in Secs. 1.3 and 1.4 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. 5a 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

$$\min_{x,y,\theta} \sum_k \left|\mathbf{M}_S^k - \mathbf{M}_R^k|_{(x,y,\theta)}\right|_2^2 \tag{10}$$

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

$$\min_{x,y,\theta} \sum_{k \in {}^i\mathbf{M}_S} \left[1 - \langle {}^i\mathbf{N}_S^k, {}^i\mathbf{N}_R^k|_{(x,y,\theta)} \rangle\right] \forall i \tag{11}$$

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

4. Finally, following Chaudhury et al. [11] 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 [13], we formulate our dense edge alignment cost as

$$\min_{x,y,\theta} \sum_k \left|\mathrm{EDT}(\mathbf{E}_\mathrm{S}) - \mathrm{EDT}(\mathbf{E}_\mathrm{R}|_{(\mathrm{x,y},\theta)})\right|_2^2 \tag{12}$$

   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. 5b 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. 5b. 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) [28]. The fourth inset of Fig. 5b 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. 5b.

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 [32]) to generate initial pose estimates and then use point-to-plane ICP [27] to solve for the change in pose of the object between the two measurements. The last two insets of Fig. 5b 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.

6. 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. (9)) and all the alignment costs (Eqs. (10) to (12)) 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 Tab. 2.

We also discuss our major design decisions and the observed effects of important hyperparameters of the system we discovered during implementing the system below.

Step	Processing + [overhead]
Capture (Fig. 1b)	0.20 (per cam.) + [5]
Normal (eq. 1,2)	3.5 (1000^2 px)
Depth (Sec. 1.4)	0.15 (1000^2 px)
Edges (Sec. 1.4)	0.05 (1000^2 px)
Pickup (Fig. 3a)	0.5 (1000^2 px)
Pickup (robot)	20 (pick and drop)
Bending (Eq. (9))	15 (1500 vert, 300 iter)) + [5]
pose est. (eq. 10, 11, 12)	12.5 - 15.5 (150 iter/eq.) + [5]

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

6.1. Practical considerations in design

We implemented the system demonstrated in the work on a robot table of $1.5\text{m} \times 1.5\text{m}$, The black mat of size ($460\text{mm} \times 610\text{mm}$) in Fig. 1a denotes the dexterous workspace of the robot. These dimensions governed the placement of the light sources, their power, and some of the camera hyper-parameters (exposure, f-stops) for our experiments. To achieve good illumination of the workspace (see Sec. 6.2, 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 Sec. 6.2, we could obtain a reasonable performance with our system.
As the lights L_1 through L_6 (Figs. 1a and 7) 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 Sec. 1.2. 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. (1) and (2)) 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) and a resulting poorer performance of Eqs. (4) and (5). Our two step illumination model requires at least three light sources to illuminate every portion of the scene for Eq. (3) 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 ([11, 17, 18, 31]) we decided to have six illumination sources.

6.2. 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. (2) and (9) to (12)) 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. 4a) 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. 4c, we made sure that the value of α in Eq. (8) was in between 0.45 to 0.55.

- Finally, for the global registration step in Fig. 5b, we used a FPFH size of 35, and a voxel downsampling factor of 1.5 in the implementation available on Open3D ([33]).

References

[1] Centric Daylight led Strip Lights for commercial and retail. 1

[2] C series fixed focal length lenses: Edmund Optics. 1

[3] Grasshopper3 USB3 Models: GS3-U3-41C6M-C and GS3-U3-89S6M-C. 1

[4] UFACTORY xArm 7. 1

[5] Jonathan T Barron and Jitendra Malik. Shape, illumination, and reflectance from shading. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(8):1670–1687, 2014. 1, 2, 3

[6] 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. 1, 2, 3

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

[8] Gary R Bradski. Computer vision face tracking for use in a perceptual user interface. 1998. 4

[9] John Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (6):679–698, 1986. 3, 4

[10] 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. 4

[11] 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. 4, 6, 7

[12] David Eberly. Reconstructing a Height Field from a Normal Map. *Geometric Tools, Redmond WA 98052. USA*. 4

[13] Pedro F Felzenszwalb and Daniel P Huttenlocher. Distance transforms of sampled functions. *Theory of Computing*, pages 415–428, 2012. 6

[14] 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. 4

[15] Berthold KP Horn. Shape from shading: A method for obtaining the shape of a smooth opaque object from one view. 1970. 1, 3

[16] 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. 4

[17] GelSight Inc. GelSight Mini. [Online; accessed 09-Dec-2022]. 7

[18] GelSight Inc. GelSight Mobile. [Online; accessed 09-Dec-2022]. 7

[19] 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. 1, 2, 3, 4

[20] 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. 3

[21] 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. 5

[22] Y. Quéau, J.-D. Durou, and J.-F. Aujol. Normal Integration: A Survey. *Journal of Mathematical Imaging and Vision*, 60(4):576–593, 2018. 3, 4

[23] 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. 2

[24] 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. 3, 4

[25] 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. 5

[26] 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. 2

[27] 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. 6

[28] 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. 6

[29] Marshall Tappen, William Freeman, and Edward Adelson. Recovering intrinsic images from a single image. *Advances in Neural Information Processing Systems*, 15, 2002. 2

[30] Stephen Wright, Jorge Nocedal, et al. Numerical optimization. *Springer Science*, 35(67-68):7, 1999. 3

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

[32] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Fast global registration. In *European Conference on Computer Vision*, pages 766–782. Springer, 2016. 6

[33] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3D: A modern library for 3D data processing. *arXiv preprint arXiv:1801.09847*, 2018. 7