Light Fields and Synthetic Aperture Photography for Robotic Perception

by
John Gilmore Oberlin
B. S., Florida State University, 2006
M. A., University of California, Berkeley, 2008

A dissertation submitted in partial fulfillment of the
requirements for the Degree of Doctor of Philosophy
in the Department of Computer Science at Brown University

Providence, Rhode Island
May 2019

© Copyright 2019 by John Gilmore Oberlin

This dissertation by John Gilmore Oberlin is accepted in its present form by the Department of Computer Science as satisfying the dissertation requirement for the degree of Doctor of Philosophy.

Date ______________ ______________________________
Stefanie Tellex, Director

Recommended to the Graduate Council

Date ______________ ______________________________
John Hughes, Reader

Date ______________ ______________________________
George Konidaris, Reader

Approved by the Graduate Council

Date ______________ ______________________________
Andrew G. Campbell
Dean of the Graduate School

Curriculum Vitae

John Oberlin holds a B.S. in Mathematics from Florida State University (2006) and an M.A. in Mathematics from U.C. Berkeley (2008). He attended The University of Chicago for the 2010-2011 school year and then attended Brown, defending in May 2018.

Notable papers include *Multiscale Fields of Patterns* (P. Felzenszwalb, J. Oberlin, *Neural Information and Processing Systems (NIPS)*, 2014) and *Time-Lapse Light Field Photography for Perceiving Transparent and Reflective Objects* (John Oberlin and Stefanie Tellex, *Robotics: Science and Systems*, 2017).

He served as a graduate student instructor (GSI) in first semester Calculus as well as Linear Algebra with Differential Equations, each for two semesters at U.C. Berkeley. He then served as a teaching assistant in Algorithms at Brown under multiple professors.

Together with a team of three undergraduates, he developed a quadcopter drone kit as curriculum for a first year course in Robotics covering the construction and programming of the drone. This project is chronicled in *PiDrone: An Autonomous Educational Drone using Raspberry Pi and Python* (Isaiah Brand, Josh Roy, Aaron Ray, John Oberlin, and Stefanie Tellex, *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018).

One of his workshop papers won a travel award: *Acquiring Object Experiences at Scale* (John Oberlin, Maria Meier, Tim Kraska, and Stefanie Tellex, *AAAI-RSS Special Workshop on the 50th Anniversary of Shakey: The Role of AI to Harmonize Robots and Humans*, 2015, Blue Sky Award).

Preface and Acknowledgements

Robotics is a broad, multidisciplinary field with some very diverse and interesting problems. Somebody said that to write a robotics paper you should make a robot do something it couldn't do before and then explain how you did it. For a couple of semesters, our hardware and software systems coalesced and our ability to localize objects exceeded the robot's precision in movement. Suddenly it was possible to implement actions on the table top at an approximately "linear rate": that is, if you could tele-operate the robot to perform an action in an environment in which you were the sole actor, it was also possible to program the robot to perform the action autonomously without too much effort. Clearly that statement was at most *essentially true*, but the feeling that it *was true* led to a different way of thinking about robotic programming, and more experiments than might otherwise have been undertaken. It wasn't too much trouble using the arm to drive nails with a hammer or screw nuts onto bolts, and so those things happened.

I am pleased to acknowledge my family members and friends. A whole mess of extra curricular activities go into my work, and these come from you. I would also like to acknowledge all of my officemates, professors, students, and undergrauate T.A.s. Extra points if you fall into multiple categories.

Contents

List of Tables

List of Figures

Part I

Robotic Light Field Photography

Thesis Statement: Light field photography performed with an eye-in-hand camera can enable a robot to perceive and manipulate most rigid household objects, even those with pathological non-Lambertian surfaces.

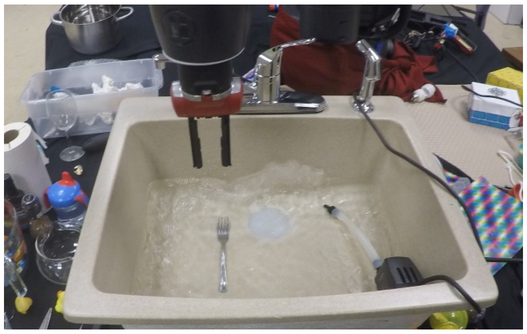
Chapter 1

Introduction

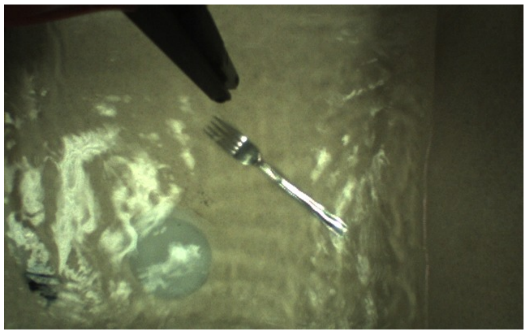
Surfaces like unpolished wood and cloth reflect light diffusely and so look the same when viewed from different perspectives. These are called Lambertian surfaces. Many tasks require a robot to detect and manipulate shiny and transparent objects, such as washing glasses and silverware in a sink full of running water, or assisting a surgeon by picking a metal tool from a metal tray. However existing approaches to object detection struggle with these non-Lambertian objects [15, 45, 12] because the shiny reflections create vivid colors and gradients that change dramatically with camera position, fooling methods that are based on only a single camera image.

Because it can move its camera, a robot can obtain new views of an object, increasing robustness and avoiding difficulties in any single view. To benefit from this technique, the robot must integrate information across multiple observations. One approach is to use feature-based methods on individual images, as in the winning team for the Amazon Picking Challenge [14], but this approach does not incorporate information about the viewing angle and can still struggle with non-Lambertian objects. Other approaches create 3D meshes from multiple views but do not work well on non-Lambertian objects [15, 45].

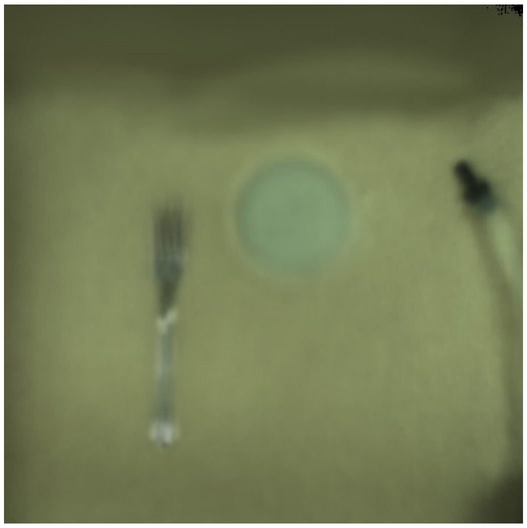
In this work, we demonstrate that light field photography [35], or plenoptic photography, enables the robust solution of many problems in robotic perception because it incorporates information from the intensity as well as the angle of the light rays, information which is readily available from a calibrated camera that can collect multiple views of a scene. Light fields naturally capture phenomena such as parallax, specular reflections, and refraction by scene elements. We present a probabilistic model and associated algorithms for turning a calibrated eye-in-hand camera into a time slice light field camera that can be used for robotic perception. Our approach enables Baxter to pick a metal fork out of a sink filled with running water 50/50 times, using its wrist camera.



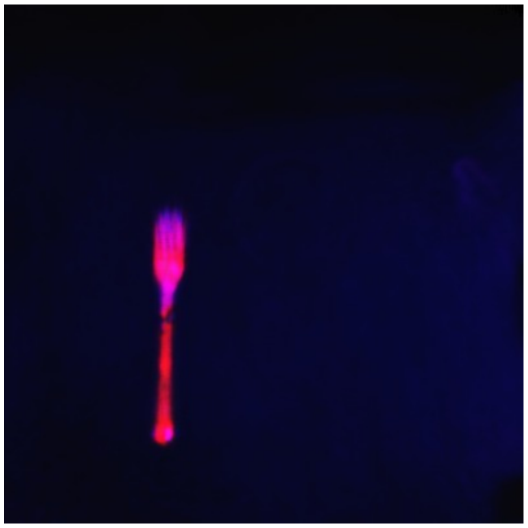
(a) Robot preparing to pick a metal fork in running water.



(b) Wrist camera view of a fork under flowing water.



(c) Rendered orthographic image of the fork; reflections from water and metal are gone.



(d) Discrepancy view, showing the robot can accurately segment the fork from the background.

Figure 1.1: Our approach allows a robot to detect, localize and manipulate under challenging non-Lambertian conditions, such as a fork under flowing water.

Chapter 2

Background and Related Work

Most camera based perception routines either solve problems by analyzing individual camera images or extract information from sequences of images in the form of small patterns, called features, and perform inference based on the relative positions and response strengths of those patterns across frames.

Our approach aggregates camera data from many pose annotated images taken close together in time. These images are collected on a robot called Baxter, manufactured by Rethink. Baxter is a stationary robot with two 7 DoF arms. On the end effector of each arm there is a parallel electric gripper, a fixed-focus color camera, and a Sharp IR triangulation range finder. The encoders in the joints give position readings for the end effector which are accurate to within $2mm$, and the joint controllers can produce motion that is repeatable within $2cm$. These bounds come from the robot specification provided by the company and are conservatively large. In practice, the end effector position seems to be accurate within $1mm$ over relative dispacements of tens of centimeters, which is on the order of the movements we use when collecting data. If the robot moves slowly, translations can be performed with sub-centimeter repeatibility, and deviations become larger with fast movement and rotations of the end effector. In order to analyze this data, we conceptualize the series of images as a different representation called the *light field*, a standard tool in graphics. We can then use the light field to synthesize 2D photographs from an orthographic projection over a very large field of view, or we can other representations of the light field to extract data about 3D structure and lighting.

2.1 Background

A typical photograph $P(x, y)$ measures the intensity of the light emanating from points in a designated focal plane, but forgets the angle at which this light originated from those points. A light field photograph F records the intensity of light not only as a function of space, but also as a function of the two angular dimensions s (roll) and t (pitch). This yields a four dimensional function of intensity $F(x, y, s, t)$, as illustrated in Figure 2.1. Unlike a single photograph, light field data can represent the fact that a metal surface shines brightly when viewed from one angle but is dull gray from another. There are other ways to parameterize the light field, and some are not restricted to a plane, but we stick to the plane because it simplifies representation and computation.

It is not currently possible to record an arbitrary lightfield at perfect fidelity, just as it is not possible to record a photograph at perfect fidelity: we must approximate each continuous function by taking samples. Whereas a photograph is a function on two dimensions, the light field is a function on four dimensions, and thus takes many more samples to populate than a photograph. It is possible to record certain light fields at a very high fidelity. A *hologram* is an interference pattern created by a the two parts of a split laser beam [34] as they both hit a common target photographic emulsion plate. One part of the beam illuminates the scene before hitting the plate and another part, the reference beam, travels straight to the plate. The interference of the two beams creates a pattern which, when illuminated by laser light of the original wavelength in the pattern of the reference beam, recreates the light field of the illuminating beam at the time of recording. This technique is clearly limited in scale and is not capable of recording light fields of scenes under natural lighting.

It is possible to use more traditional photography to capture light fields. It is typical in such practice to express the light field as a collection of sub-photographs where each sub-photograph is a function of the two variables s and t. Each sub-photograph records the light field $F(x, y, s, t)$ for a specific (x, y) pair. Such a collection can be taken over time by a moving camera (in which case it is a time-slice approximation), taken simultaneously using a lens array, or taken almost simultaneously using a camera array.

The first photographic representation of a dense light field occurred in 1908 and is due to G. Lippmann [7]. His technique, the first example of *integral imaging*, used a lens array to photographically capture an array of sub images which encoded light field data, although neither the function nor the term existed at the time and no reliable tools existed for viewing or manipulating the data. A notable analog method for viewing such data was introduced in 1967 by R.V. Pole, wherein a photograph from a lens array is used to create a (relatively sparse) hologram [7]. The hologram combines the information from the sub-photographs into a single representation which can be viewed in the manner of a traditional hologram.

Digital representation and manipulation of the light field started gaining traction in the 1980's. In 1987, Bolles showed how to construct an epipolar plane image (EPI) 2.3 from images taken sequentially in space [9], and how to use line detection in the EPI to determine the distances of points in the scene from the camera. Although the term *light field* was coined in 1936 by Andrey Gershun [18], his notion was incomplete and referred to a three dimensional vector field. In 1991, Adelson and Bergen introduced the 5 dimensional *plenoptic function* $P(x, y, z, s, t)$ [1], which effectively describes the four dimensional light field $F(x, y, s, t)$ at all focal planes z. They also outline the roles of low order derivative filters across different dimensions of the plenoptic function in capturing salient, low dimensional information about the environment, and propose ways these mechanisms might be employed in the primate brain. If there are no occluding objects in a region of space, the light field within planes of that space can be calculated in closed form from one another. The five dimensional function is in our cases, therefor, usually very redundant. It is for this reason that we use the four dimensional light field rather than the five dimensional plenoptic function.

At SIGGRAPH 1996, there were two influential works that set the stage for much of the future work involving light fields. In *The Lumigraph*, Cohen et al. describe a system for collecting photographs of an object on a special capture stage [21]. This capture stage provides pose information for the camera, which can then be used to construct the 4D light field over a 6 plane surface enclosing the object. This information can then be used to perform 3D reconstruction of the object and to render novel views. In *Light Field Rendering*, Levoy and Hanrahan describe a method for collecting pose annoted images of a scene with a camera gantry, as well as methods for rendering new views of the scene using only the collected photographic data [35]. These two works have strongly influenced many modern techniques of digital light field capture and manipulation.

We store light fields as pose annotated images so that no information is lost during binning: we remember the 6 DoF pose of the camera when it records each image, and we can use this to further approximate the light field in whatever form is useful for a particular task.

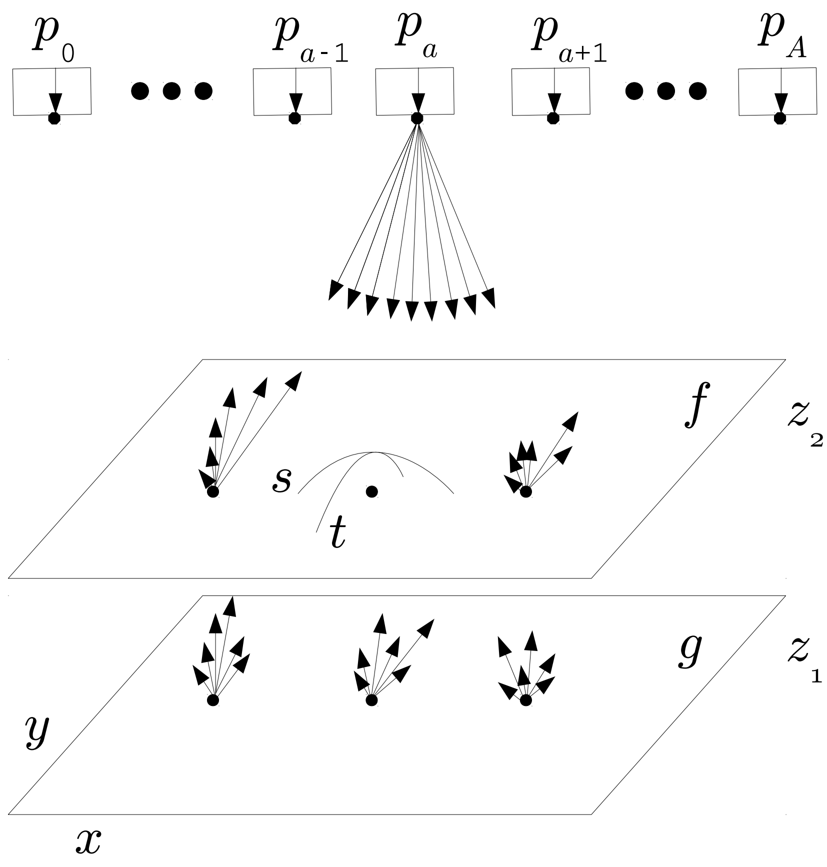
We can capture the light field F in one plane and use it to determine the light field G in another (not necessarily parallel) plane, as in Figure 2.1, but the parallel case is made efficient by the convolution theorem [42]. Following the precedent of Lumigraph and Light Field Rendering, given a light field, we can synthesize 2D *synthetic photographs* in a target plane by casting rays from the light field into the target plane and summing their intensities at each pixel in the target plane. Equivalently, we can form a photograph $P(x, y)$ in a target plane by first calculating the light field $F(x, y, s, t)$ in the target plane and setting

$$P(x, y) = \sum_{s,t} F(x, y, s, t).$$

By thinking of pixel measurements in terms of the light field, we can generate 2D synthetic photographs from arbitrary perspectives, as long as we have collected the necessary rays. One such class of photographs is *orthographic projections*, where each point in the image appears as if it is viewed from directly above with respect to the image plane. Orthographic images are free from perspective distortion, which makes them easier to analyze. But beyond 2D synthetic photographs, there are two ways we will represent light field data as an image, which will improve intuition and make computing certain quantities easier.

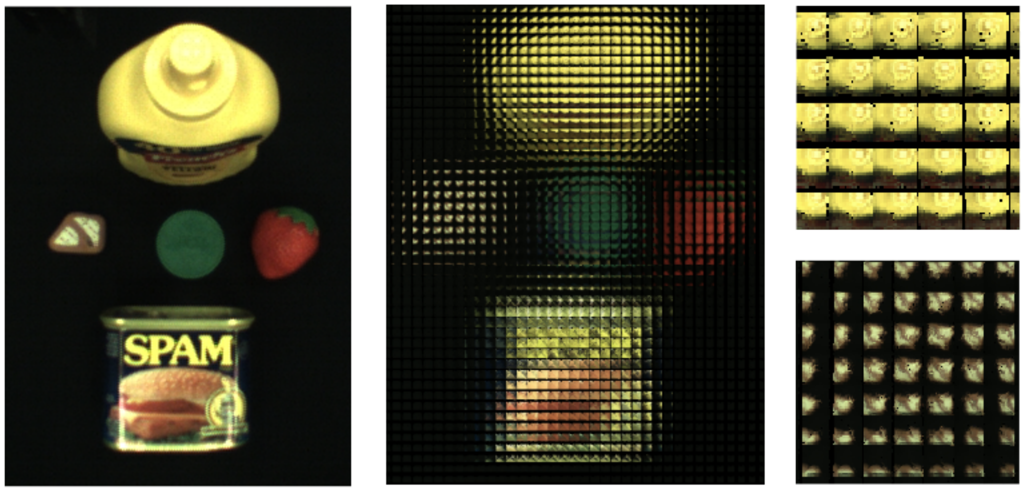
The first type of image is a 2D array of sub images which encode the 4D light field in a target focal plane. We will call this representation a *light slab* or *image array*. Light slabs allow us to capture optical effects like reflections and focal cues better than synthetic photographs because they sort rays according to both the target position and the angle of convergence on the focal plane. We parameterize the index of the sub images by the variables x and y, and the pixels within the sub images by s and t. In a normal synthetic photograph, all the light contributing to all of the pixels in a sub image (x, y) would be focused into a single pixel of the photograph. In a light slab, the light coming into an (x, y) cell is instead assigned within the (x, y) sub image to (s, t) pixels corresponding to the angle at which the light is incident in the focal plane. Conceptually, each sub image is like a pinhole image taken by an aperture in the (x, y) location in the focal plane, visualized in Figure 2.2. Light slabs are very much like the images captured by a Lytro light field camera using a main lens and a micro lens array. One way to form a 2D photograph in the plane of a light slab is by fixing x and y while summing over s and t to form one pixel in the 2D photograph per sub image in the light slab.

The second type of image is called an *epipolar plane image* (EPI). Where a photograph can be considered a 2D *projection* or integral over the (s, t) variables of the 4D light field, an EPI is a 2D *slice* across parallel dimensions of the 4D light field, either in x and s or y and t. Slices over parallel dimensions make parallax induced by sub image "motion" appear as slanted lines whose slopes correspond to the distance of those lines from the focal plane of the sliced light slab, illustrated in Figure 2.3. The geometric structure of the EPI



(a) The light field is initially represented as a collection of pose annotated images. It can be resampled as a four dimensional function of two spatial variables x and y, and two angular variables s and t. This resampling can be performed in an aribitrary plane, but will only be the true light field in that plane if no occlusions exist between the cameras and the plane. Light fields can be resampled effeciently between parallel planes using convolution [42].

Figure 2.1: A ray diagram showing different representations of one lightfield.



(a) Left: Single camera image showing a table top scene. Middle: Light slab of the scene, an array of pinhole images taken from the focal plane of the light slab. The focal plane is just above the surface of the SPAM can, which is why the sub images over the SPAM consist of nearly one color. Right: Magnified sub images showing parallax as the perspective shifts. Black dots appear where there is missing data. The mustard is magnified more to show detail while the tile is magnified less to show more parallax over a larger baseline.

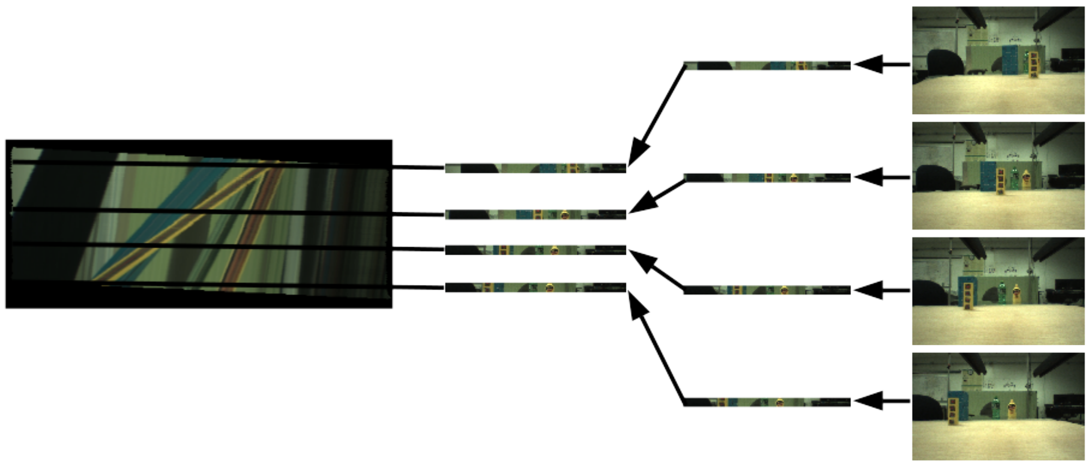
Figure 2.2: A photograph and a light slab of the same scene.

provides a means to capture refraction and parallax cues using primitive computer vision algorithms such as line detection.

Reflected light that would be averaged out over angle in an synthetic photograph remains separate in a light slab or EPI, which are themselves two different parameterizations of information contained in the 4D light field. Each parameterization emphasizes different cues and structures in the light field.

2.2 Related Work

Time-slice light field photography has precedent [67, 35], but the movement is typically constrained to a few dimensions. Fixed camera [66] and microlens [17, 42] arrays are stable once calibrated and can capture angular information from many directions simultaneously, but camera arrays are not very portable and microlens arrays do not have a very large baseline. Baxter's arm allows us to densely collect images (in sub millimeter proximity to each other) across large scales (about a meter) over a 6 DoF of pose in a 3D volume. This enables the study of light fields in a diverse variety of modes on a widely available piece of equipment, Baxter, which is relatively inexpensive for a robot arm and is already present in hundreds of robotics research labs. To our knowledge, given our approach, such an eye-in-hand robot may be the most flexible and accessible apparatus for light field research, despite the limits ultimately imposed by joint encoder quantization and a relatively inexpensive camera. Other datasets using camera gantries have been released, which include wider baselines [64]. Existing approaches perform depth estimation [63], shape estimation [59] and other computer vision tasks. However we are unaware of a robot being used as a light field capture device online. Using a



(a) The coordinates s and t are the column and row coordinates in the constituent images corresponding to angular coordinates in the light field. The coordinates x and y are the physical coordinates of the camera when the images were taken, corresponding to the spatial coordinates of the light field. In these images, the arm moves along the x axis, which is parallel to the s axis. Right: Images taken as the arm moves in the x direction. Middle: Slices in s extracted from the images, fixing t. Left: Stacked slices showing parallax as sloped lines.

Figure 2.3: Constructing an epipolar plane image.

7 DoF arm as a camera gantry allows the robot to dynamically acquire more views and integrate information from multiple views in a probabilistic setting, providing new attacks on object detection, picking, and non-Lambertian objects.

The variance of the rays contributing to a pixel in a synthetic photograph have been considered as a correspondence cue [59]. To our knowledge, though, our graphical model 3.1 and our inference techniques are a novel use of this information. The technique of rendering multiple synthetic photographs of the same scene at different focal planes to perform depth estimates is fairly standard. There are even cameras that collect multiple focal planes of the same scene simultaneously in order to perform high performance foreground background segmentation [39]. Our use of the variance information in the probabilistic graphical model for segmentation, pose estimation, and specular highlight reduction is new, to our knowledge, but this information can be expressed in different forms and so it is possible these techniques exist in another formalism.

Other work on depth reconstruction from light field data has considered shading cues to introduce global consistency and differential focus cues to improve local estimates [59]. Our graphical model would benefit from the introduction of such terms, but adding that structure would in many cases require learning the statistics of target scenes. Our simplified graphical model is parameter free for many tasks in the sense that depth inference 3.2.6 and reasoning about the nature of reflections 3.2.5 can be performed on the basis of physical laws and don't depend on learning about the domain before performing inference on it. There is more work to be done in this direction extending the graphical model for consistency. For instance, there exist well motivated models for probabilistically estimating depth from the flow of specular highlights over reflective surfaces [52].

One robotic system has used a Raytrix light field camera to provide depth estimates during a robotic suturing task with a 7 DoF arm [56]. The robot performed better than trained surgeons in some tasks. Our work differs from theirs in the types of tasks performed, and in that they used a stationary light field camera to provide depth estimates. Our work uses a moving single perspective camera to gather light field data and performs pose estimation and depth estimation online through synthetic photography.

Some work in robotic perception describes an approach for detecting transparent objects that assumes objects are rotationally symmetric [45]. Our work, by contrast, can detect transparent and specular objects by reasoning about how they distort the light rays themselves. There is prior work using a camera array to perform video stabilization using light fields [57] but it did not provide a framework for non-Lambertian objects. Kanade, Rodrigues, et al. describe a multi-light system to detect and localize transparent objects [50]; our light field approach can benefit from controlling lighting, and opens the door to lighting models for further reducing artifacts and predicting next best views.

Herbst defines a probabilistic surface-based measurement model for an RGB-D camera and uses it to segment objects from the environment [22]. A number of approaches use robots to acquire 3D structure of objects either by moving the camera or moving the object [31, 5, 25, 61, 54, 27, 40, 37, 30, 5]. Our approach, by contrast, uses a model based around light fields, incorporating both intensity and direction of light rays. This approach can be generalized to IR cameras as well, augmenting approaches such as KinectFusion [41] to handle non-Lambertian surfaces and exploiting complementary information from the IR and RGB channels. Correll et al. review entries to the Amazon Robotics Challenge [12], a variety of state-of-the-art systems and state that non-Lambertian objects pose a challenge; the winning entry had problems due to the reflective metal shelves [14].

A separate body of work automatically acquires 3D structure of the environment from RGB or RGB-D cameras. Notably, LSD Slam [13] uses a pixel-based approach and achieves efficiency using key frames and semi-dense depth maps. Our approach, instead, uses all pixel values from observed images to render a synthetic photograph, which could then be processed with gradients or other steps. More recent extensions detect, discover and track objects while mapping [62, 16, 11, 53, 55, 65], but do not explicitly model light rays in the scene. When pose is already available, as on Baxter's wrist camera, our approach enables efficient use of all information from the camera, integrating information across many images.

Many existing computer vision problems, such as object detection, are canonically solved within a single image [58] and typically trained on a large data set of photographs taken by human photographers [28, 29, 33, 20]. By contrast, the robot's ability to move its own camera enables it to dramatically constrain problems of object detection and pose estimation. Feature-based approaches such as SIFT or FAST [23, 51, 36] capture information from individual images; neural networks [19] learn to extract structure from pixels. Existing approaches to scene reconstruction, such as multi-view stereo [15] cannot handle non-Lambertian surfaces. Our work, by contrast, explicitly models the behavior of light over the surfaces of objects, enabling improved performance on challenging reflective objects. Many of these approaches can be applied to light fields rather than photographs, and we are excited about the potential improvement from having access to metric information and multiple configurations for objects

Chapter 3

A Probabilistic Model of Synthetic Photography and Object Detection

3.1 The Model

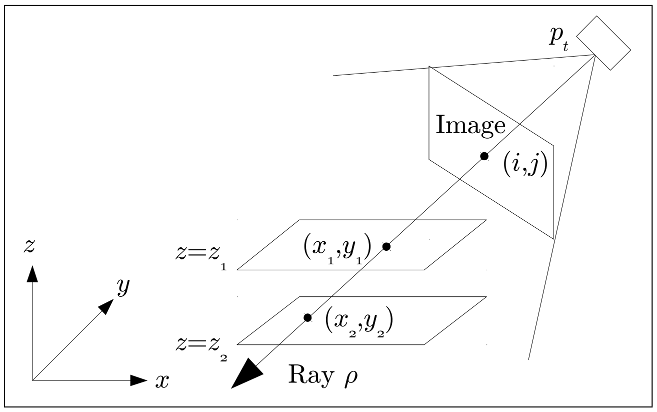
We present a probabilistic model for light field photography using a calibrated eye-in-hand camera. Inference in this model corresponds to an inverse graphics algorithm that finds a 2.5D model of the scene using a process called a *focus sweep*. During a focus sweep, we render a series of synthetic photographs with different focal distances and assign to each point the distance at which it was most in focus and the color observed at that point in the image rendered at that distance.

We assume that the robot is observing the scene from multiple perspectives using a camera, receiving a sequence of images, $Z = \{Z_1, \ldots, Z_H\}$. The robot also receives a sequence of poses, $P = \{p_1, \ldots, p_H\}$, containing camera position and orientation. We assume access to a calibration function that defines a ray for each pixel in an image: $\mathcal{C}(i, j, r_h, z) \rightarrow \{(x, y)\}$ which converts pixel i and j coordinates to an x and y in the robot's base coordinate system given a z along the ray, along with its inverse: $\mathcal{C}^{-1}(x, y, p_h, z) \rightarrow (i, j)$. Section 3.2.1 describes this function in detail, including how we estimate its parameters for the Baxter robot.

This calibration function enables us to define a set of light rays, R, where each ray contains the direction and intensity information from each pixel of each calibrated image. Formally, each $\rho \in R$ consists of an intensity value, (r, g, b)[1], as well as the pose of the image, r_h, and its pixel coordinate, (i, j). This information, combined with the calibration function, enables us to use the ray components to compute an (x, y) coordinate for any height z. Figure 3.1 shows a diagram of a scene.

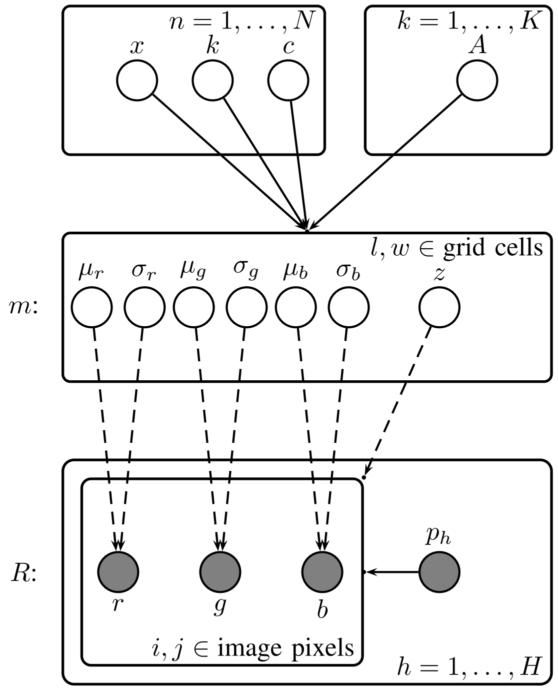
Next, we define a distribution over the light rays emitted from a scene. Using the generative model depicted in the plate diagram in Figure 3.2, we can perform inference about the underlying scene, conditioned on observed light rays, R. We define map, m, as an $L \times W$ array of cells in a plane in space. Each cell $(l, w) \in m$ has a height z and scatters light at its (x, y, z) location. For convenience, we write that the calibration function C can take either (x, y) coordinates in real space or (l, w) indexes into map cells. We

[1]We use (r, g, b) here because it is more intuitive; our implementation uses the YCbCr color space.



(a) For $z = z_1$, ray $\rho \in R_{x_1,y_1}$; for $z = z_2$, ray $\rho \in R_{x_2,y_2}$.

Figure 3.1: Diagram of a scene captured by a camera, mapping pixels in the camera to points in space.



(a) Dashed lines indicate that those edges are only present for a subset of the variables, for the bundle of rays $R_{l,w}$ that correspond to a particular grid cell, defined formally in Equation 3.4.

Figure 3.2: Graphical model for our approach.

assume each observed light ray arose from a particular cell (l, w), so that the parameters associated with each cell include its height z and a model of the intensity of light emitted from that cell. Formally, we wish to find a maximum likelihood estimate for m given the observed light rays R:

$$\underset{m}{\operatorname{argmax}}\, P(R|m). \tag{3.1}$$

We assume that the bundle of rays striking each cell is conditionally independent given the cell parameters. This assumption is violated when a cell actively illuminates neighboring cells (e.g., a lit candle), but enables us to factor the distribution over cells:

$$P(R|m) = \prod_{l,w} P(R_{l,w}|m). \tag{3.2}$$

Here $R_{l,w} \subset R$ denotes the bundle of rays that arose from map cell (l, w), which can be determined by finding all rays that intersect the cell using the calibration function. Formally:

$$R_{l,w} \equiv \{\rho \in R | C(i, j, p_h, z) = (l, w)\}. \tag{3.3}$$

We assume that each cell emits light on each channel as a Gaussian over (r, g, b) with mean $\mu_{l,w} = (\mu_r, \mu_g, \mu_b)$ and variance $\sigma^2_{l,w} = \left(\sigma^2_r, \sigma^2_g, \sigma^2_b\right)$:

$$P(R|m) = \prod_{l,w} P(R_{l,w}|\mu_{l,w}, \sigma^2_{l,w}, z_{l,w}). \tag{3.4}$$

We rewrite the distribution over the bundle of rays, $R_{l,w}$, as a product over individual rays $\rho \in R_{l,w}$:

$$P(R_{l,w}|\mu_{l,w}, \sigma^2_{l,w}, z_{l,w}) = \prod_{\rho \in R_{l,w}} P(\rho|\mu_{l,w}, \sigma^2_{l,w}, z_{l,w}). \tag{3.5}$$

Next we assume each color channel c in r is independent. We use ρ_c to denote the intensity value of ρ for channel c.

$$P(R_{l,w}|\mu_{l,w}, \sigma^2_{l,w}, z_{l,w}) = \prod_{\rho \in R_{l,w}} \prod_{c \in \{r,g,b\}} P(\rho_c|\mu_c, \sigma^2_c, z_{l,w}). \tag{3.6}$$

As a Gaussian:

$$P(\rho_c|\mu_c, \sigma^2_c, z_{l,w}) = \mathcal{N}(\rho_c, \mu_c, \sigma^2_c). \tag{3.7}$$

We can render m as an image by showing the values for $\mu_{l,w}$ as the pixel color; however variance information $\sigma^2_{l,w}$ is also stored. Figure 3.3 and Figure 3.4 show example scenes rendered using this model.

Synthetic photographs observed in the past (with both μ and σ information) therefore define distributions which can be evaluated on future synthetic photographs. We can use synthetic photographs of objects in different poses to perform pose estimation for tabletop objects. We can use synthetic photographs of entire scenes to perform change detection and segmentation as the scene is altered, for instance by adding an object to a known scene in order to segment the object. The variance channel contains useful information about a scene that can be used directly: High variance regions in focused photographs usually correspond to *shiny* regions, such as water droplets , grease smears, glass, or metal. Changes in the shininess can indicate the presence of dirt, grime, or residues, even if the average color remains essentially the same. There are many choices to make when applying a model photograph to a target photograph for inference. A well behaved optimization is to evaluate the μ channels of the target under the Gaussians defined by both μ and σ variables.

Each object we wish to detect has a known appearance model A_k, which is a collection of C_k synthetic photographs (with means and variances) of the object, called *configurations*. The synthetic photographs are formed at either a fixed height z or with inferred heights for each cell (the same choice for all K objects). In practice, we use background subtraction to segment the object during scanning and take a small crop containing the object and some padding to form a configuration. Cells which are not discrepant with the background are considered to have infinite variance and are not taken into account during future inference.

Each configuration can be used to localize an object as its pose changes rigidly in the plane, allowing inference of (x, y, θ) coordinates. Most objects have a small number of gravitationally stable poses, so only a small number of configurations is necessary to model each object in the poses that it is likely to appear in. An obvious exception to this is objects which can roll and have very different suface patterns on the rolling surface, but we have succeded in localizing such objects as well.

An expert can manually set the number of configurations, changing the pose of the object and scanning a new configuration for each pose. During the operation of the Infinite Scan (see Section 5.1.4), we allow a robot to "play" with the object and try to decide whether it is seeing the object from a new view. We are still investigating ways to make that decision, and ways to model these transitions, so we leave this discussion for future work, along with the task of defining more continuous object models.

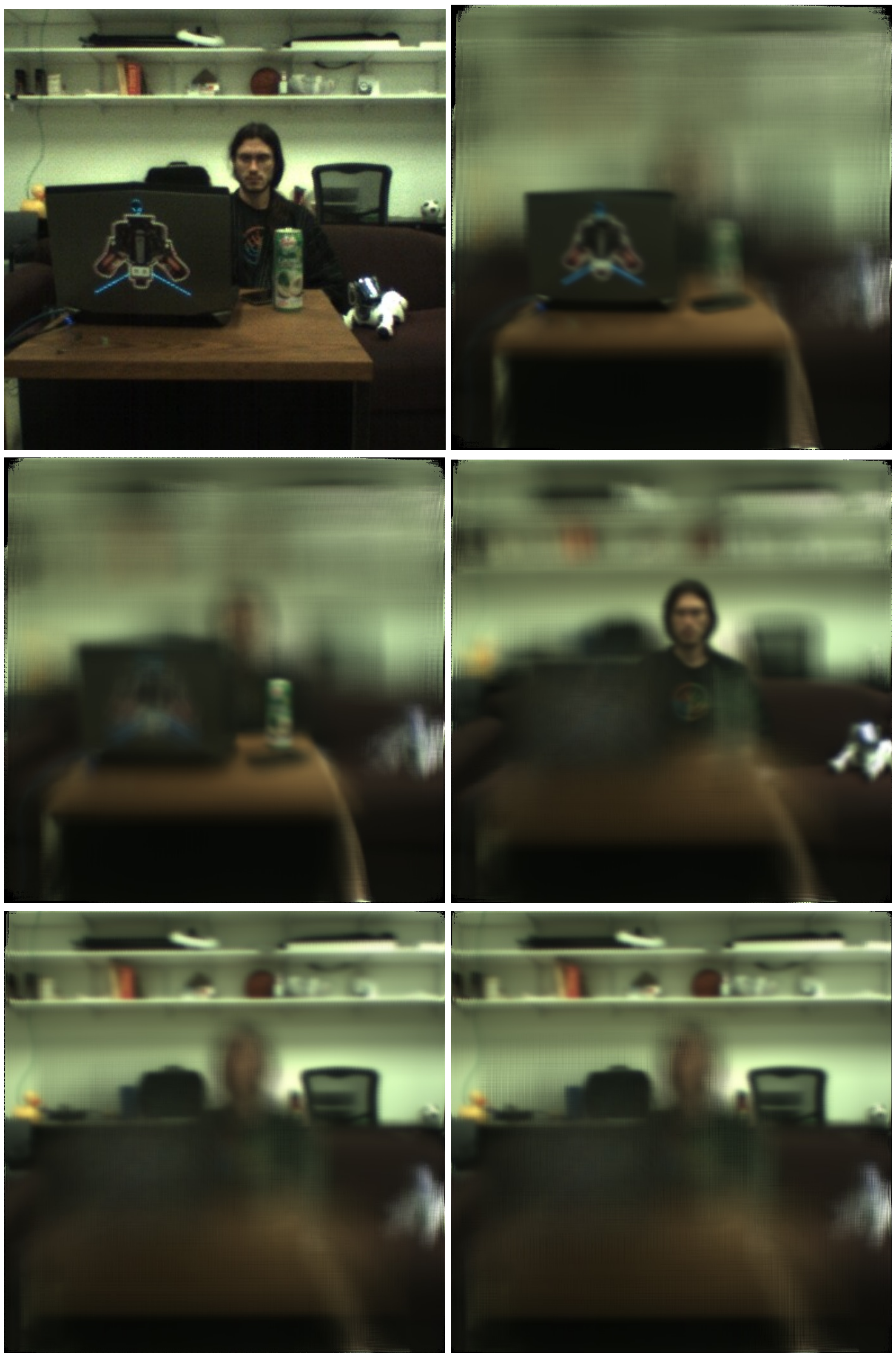
3.2 Algorithms Applying the Model

3.2.1 Camera Calibration

In order to accurately focus rays in software to synthesize sharp 2D photographs, we must be able to determine the path of a ray of light corresponding to a pixel in a camera image given the camera pose for that image, r_t, and the pixel coordinate generating that ray, (i, j). We define a calibration function of the form $\mathcal{C}(i, j, r_h, z) \to \{(x, y)\}$. Then, having specified z, the pixel (i, j) maps to the world coordinate (x, y, z). To perform calibration, we first define a model for mapping between pixel coordinates and world coordinates. Then, using Equation 3.4, we find the maximum likelihood estimates for the model parameters corresponding to the parameters of our calibration function $\mathcal{C}$, including the magnification factor in each axis and the principle point of the camera..

We build a 4×4 matrix S which encodes the affine transformation from pixel coordinates to physical coordinates for an origin centered camera. Next we compose it with the 4×4 matrix T representing the end effector transformation, obtained by forward kinematics over the joint angles supplied by the encoders.

To simplify the notation of the matrix math, we use a four dimensional vector a^p to represent a pixel $p = (i, j)$. Let $y^p = i$ and $x^p = j$. Suppose the image is w pixels wide and h pixels tall. Let $a^p = (x^p, y^p, 1, 1)$ represent a pixel in the image located at row y and column x, where x can span from 0 to $w - 1$, and y can span from 0 to $h - 1$. We specify x^p and y^p as integer pixels, assign a constant value of 1 to the z coordinate,



(a) Top Left: A single image from the wrist camera. Remaining: Refocused photographs computed with approximately 4000 wrist images and focused at 0.91, 1.11, 1.86, 3.16, and 3.36 meters.

Figure 3.3: A room scene refocused at different depths.

and augment with a fourth dimension so we can apply full affine transformations with translations. Assume that the principle point c_p of the image is $c_p = (c_{px}, c_{py}) = (\frac{w}{2}, \frac{h}{2})$. That is to say, the aperture of the camera is modeled as being at the origin of the physical coordinate frame, facing the positive z half space, collecting light that travels from that half space towards the negative z half space, and the z axis intersects c_p. In the pinhole model, only light which passes through the origin is collected, and in a real camera some finite aperture width is used. The camera in our experiments has a wide depth of field and can be modeled as a pinhole camera. We define a matrix T to correspond to the affine transform from r_t to the coordinate system centered on the camera, and S_z to correspond to the camera parameters. If we want to determine points that a ray passes through, we must specify the point on the ray we are interested in, and we do so by designating a query plane parallel to the xy-axis by its z coordinate. We can then define the calibration function as a matrix operation:

$$TS_z \begin{bmatrix} (x^p - c_{px}) \\ (y^p - c_{py}) \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ 1 \\ 1 \end{bmatrix}. \tag{3.8}$$

To determine the constants that describe pixel width and height in meters, we obtained the relative pose of the stock wrist camera from the factory measurements. Here M_x and M_y are the camera magnification parameters:

$$S_z = \begin{bmatrix} M_x \cdot z & 0 & 0 & 0 \\ 0 & M_y \cdot z & 0 & 0 \\ 0 & 0 & z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3.9}$$

We model radial distortion in the typical way by making S_z a function not only of z but also quadratic in $(x^p - c_{px})$ and $(y^p - c_{py})$.

Calibrating the camera involves finding the magnification terms M_x and M_y (though the principle points and the radial quadratic terms can be found by the same means). To find the magnification coefficients, we set a printed paper calibration target on a table of known height in front of Baxter. We collect camera images from a known distance above the table and estimate the model for the collected rays, forming a synthetic image of the calibration target. The values for M_x and M_y which maximize the likelihood of the observed rays under the estimated model in Equation 3.4 are the values which yield the correct pixel to global transform, which incidentally are also the values which form the sharpest synthetic image. We find M_x and M_y with grid search. Determining such parameters by iteratively rendering is a form of bundle adjustment, and also allows us to perform 3D reconstruction. It is repeatable and precise, and the cameras are consistent enough that we can provide a default calibration that works across different robots.

3.2.2 Inferring a Synthetic Photograph

During inference, we estimate m at each height z using grid search. Most generally, each ray should be associated with exactly one cell; however this model requires a new factorization for each setting of z at

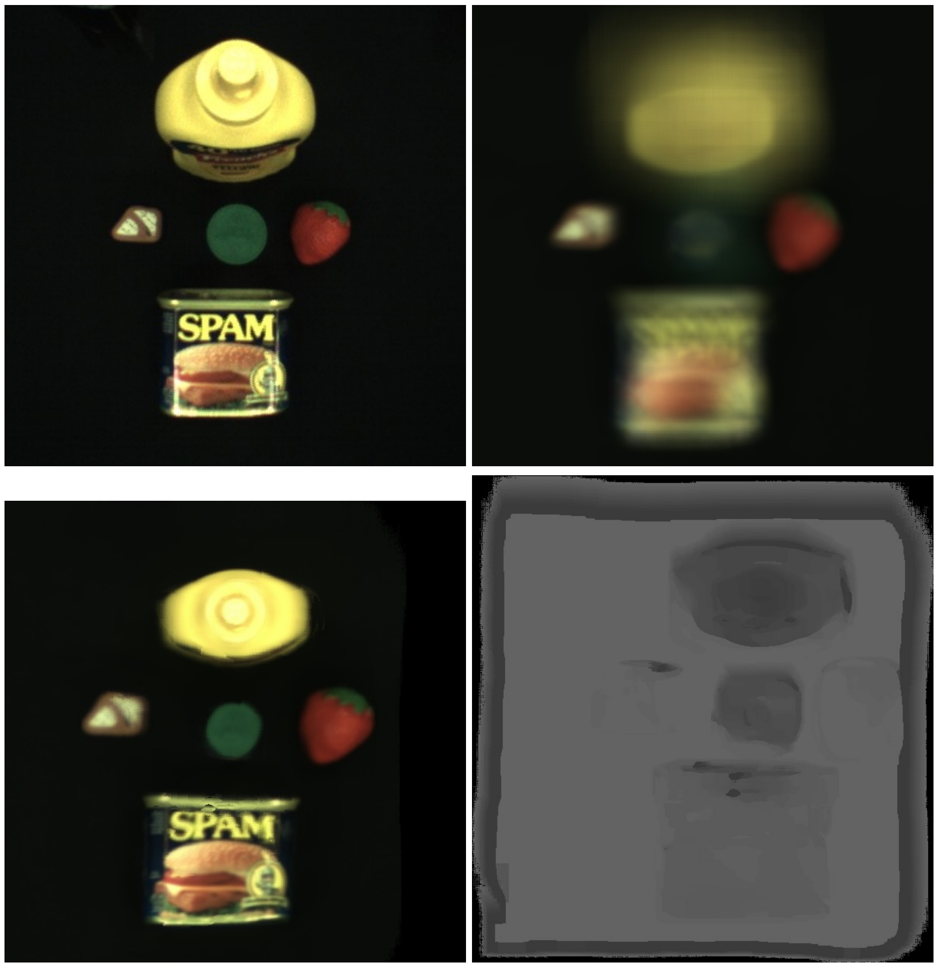


Figure 3.4: Top Left: A single image from the wrist camera, showing perspective. Top Right: Refocused image converged at table height, showing defocus on tall objects. Bottom Left: Approximate maximum likelihood RGB image, showing all objects in focus, specular reflection reduction, and perspective rectification. Bottom Right: Depth estimates for approximate maximum likelihood image.

inference time, as rays must be reassociated with cells when the height changes. In particular, the rays associated with one cell, $R_{l,w}$, might change due to the height of a neighboring cell, requiring a joint inference over all the heights, z. If a cell's neighbor is very tall, it may occlude rays from reaching it; if its neighbor is short, that occlusion will go away.

As an approximation, instead of labeling each ray with a cell, we optimize each cell separately, over counting rays and leading to some issues with occlusions. This approximation is substantially faster to compute because we can analytically compute $\mu_{l,w}$ and $\sigma^2_{l,w}$ for a particular z as the sample mean and variance of the rays at each cell. In the future we plan to explore EM approaches to perform ray labeling so that each ray is assigned to a particular cell at inference time. This approximation works well for small distances because it overcounts each cell by approximately the same amount; we expect it to have more issues estimate depths at longer distances, such as that shown in Figure 3.3.

Figure 3.4 shows the depth estimate for a tabletop scene computed using this method. The images were taken with the camera 38 cm from the table. The top of the mustard is 18 cm from the camera and very shiny, so this degree of local estimation is non-trivial. The RGB map is composed from metrically calibrated images that give a top down view that appears in focus at every depth. Such a map greatly facilitates object detection, segmentation, and other image operations. Note that fine detail such as the letters on the SPAM is visible.

3.2.3 Detecting Changes

Once the robot has found an estimate, m, for a scene, for example to create a background model, it might want to detect changes in the model after observing the scene again and detecting a ray, ρ' at (l, w). At each cell, (l, w), we define a binary random variable $d_{l,w}$ that is false if the light for that cell arose from background model m, and true if it arose from some other light emitter. Then for each cell we want to estimate:

$$P(d_{l,w}|m, \rho') = P(d_{l,w}|\mu_{l,w}, \sigma^2_{l,w}, \rho'). \tag{3.10}$$

We rewrite using Bayes' rule:

$$= \frac{P(\rho'|d_{l,w}, \mu_{l,w}, \sigma^2_{l,w}) \times P(d_{l,w}|\mu_{l,w}, \sigma^2_{l,w})}{P(\rho'|\mu_{l,w}, \sigma^2_{l,w})}. \tag{3.11}$$

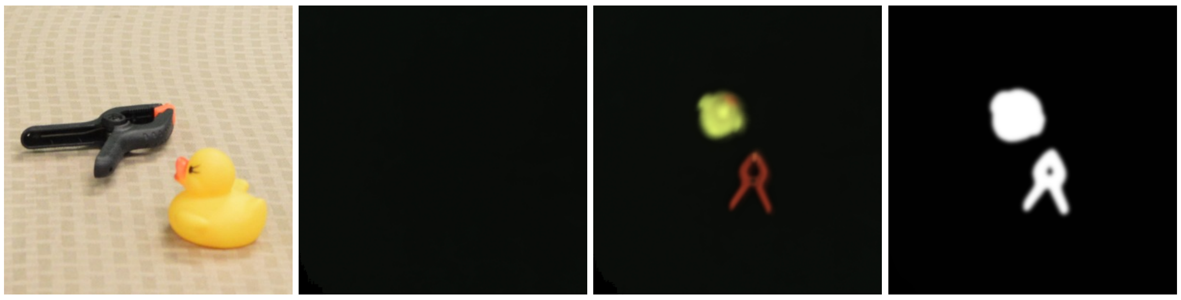
We use the joint in the denominator:

$$= \frac{P(\rho'|d_{l,w}, \mu_{l,w}, \sigma^2_{l,w}) \times P(d_{l,w}|\mu_{l,w}, \sigma^2_{l,w})}{\sum_{d_{l,w}\in\{0,1\}} P(\rho'|d_{l,w}, \mu_{l,w}, \sigma^2_{l,w}) \times P(d_{l,w}|\mu_{l,w}, \sigma^2_{l,w})}. \tag{3.12}$$

We initially tried a Naive Bayes model, where we assume each color channel is conditionally independent given $d_{l,w}$:

$$= \frac{\prod_{c\in\{r,g,b\}} P(\rho'_c|d_{l,w}, \mu_c, \sigma^2_c) \times P(d_{l,w}|\mu_c, \sigma^2_c)}{\sum_{d_{l,w}\in\{0,1\}} \prod_{c\in\{r,g,b\}} P(\rho'_c|d_{l,w}, \mu_c, \sigma^2_c) \times P(d_{l,w}|\mu_c, \sigma^2_c)}. \tag{3.13}$$

If $d_{l,w}$ is false, we use $P(\rho_c|d_{l,w} = 0, \mu_c, \sigma^2_c) = \frac{1}{255}$; otherwise we use the value from Equation 3.7. We only use one ray, which is the mean, $\mu'_{l,w}$ of the rays in $R'_{l,w}$. We use a uniform prior so that $P(d_{l,w}|\mu_c, \sigma^2_c) =$



(a) Scene. (b) Initial map. (c) Map with objects. (d) Discrepancy.

Figure 3.5: The mean values of a map m for a scene without objects, and the map after two objects have been added.

0.5. However this model assumes each color channel is independent and tends to under-estimate the probabilities, as is well-known with Naive Bayes [6]. In particular, this model tends to ignore discrepancy in any single channel, instead requiring at least two channels to be substantially different before triggering.

For a more sensitive test, we use a Noisy Or model. First we define variables for each channel, $d_{l,w,c}$, where each variable is a binary indicator based on the single channel c. We rewrite our distribution as a marginal over these indicator variables:

$$P(d_{l,w}|m,\rho') = \sum_{d_{l,w,r}} \sum_{d_{l,w,g}} \sum_{d_{l,w,b}} \begin{array}{c} P(d_{l,w}|d_{l,w,r}, d_{l,w,g}, d_{l,w,b}) \times \\ P(d_{l,w,r}, d_{l,w,g}, d_{l,w,b}|m,\rho') \end{array} .$$

We use a Noisy Or model [44] for the inner term:

$$\begin{aligned} &P(d_{l,w}|d_{l,w,r}, d_{l,w,g}, d_{l,w,b}, m, \rho') = \\ &\qquad 1 - \prod_{c\in\{r,g,b\}} [1 - P(d_{l,w}|d_{l,w,c} = 1 \wedge d_{l,w,c'\neq c} = 0, m, \rho')]^{d_{l,w,c}} . \end{aligned} \tag{3.14}$$

We define the inner term as the single channel estimator:

$$P(d_{l,w} = 1|d_{l,w,c} = 1 \wedge d_{l,w,c'\neq c} = 0, m, \rho') \equiv P(d_{l,w,c}|m,\rho'). \tag{3.15}$$

We define it with an analogous version of the model from Equation 3.13 with only one channel:

$$\frac{P(\rho'_c|d_{l,w,c}, \mu_c, \sigma_c^2) \times P(d_{l,w,c}|\mu_c, \sigma_c^2)}{\sum_{d_{l,w}\in\{0,1\}} P(\rho'_c|d_{l,w}, \mu_c, \sigma_c^2) \times P(d_{l,w}|\mu_c, \sigma_c^2)}. \tag{3.16}$$

Figure 3.5 shows a scene rendered with no objects, and after two objects have been added. Note the accurate segmentation of the scene obtained by the robot's ability to move objects into the scene and compare the information using an orthographic projection.

3.2.4 Classifying Objects and Detecting Poses

Our goal is to localize an object in the scene, given a map of the object. For this task we assume a single object with known map A_k.

Localizing an object means finding the most probable object pose x given the images Z, camera poses P, object map A_k and configuration c:

$$\underset{x}{\operatorname{argmax}}\, p(x|Z, A_k, c, P). \tag{3.17}$$

We factor using Bayes' rule and drop the normalizer, which is constant with respect to x. Additionally, we assume a uniform prior on x, omitting it from the optimization:

$$\underset{x}{\operatorname{argmax}}\, p(Z|x, A_k, c, P). \tag{3.18}$$

We rewrite the object pose x:

$$\underset{x,y,\theta,c}{\operatorname{argmax}}\, p(Z|x, A_k, c, P). \tag{3.19}$$

We use a sliding window approach to carry out this optimization over the object's planar position and orientation. Since computing the likelihood is expensive, we first compute candidate poses by finding highly discrepant regions. We then run the full model on the first D candidates; in our current implementation D is set to 3000. We infer entire scenes using a greedy algorithm.

When dealing with multiple object types, we add objects sequentially, adding higher probability objects first. Once an object has been added to the scene model, the discrepancy between the observed light field grid and predicted light field grid is reduced, as shown in Figure 3.5. This approach allows the system to handle clutter by incrementally adding objects to account for what it sees.

To label an object instance with its type k, we need to maximize over k given the observed information:

$$\underset{k}{\operatorname{argmax}}\, p(k|Z, A_k, P). \tag{3.20}$$

We factor using Bayes' rule, assuming a uniform prior:

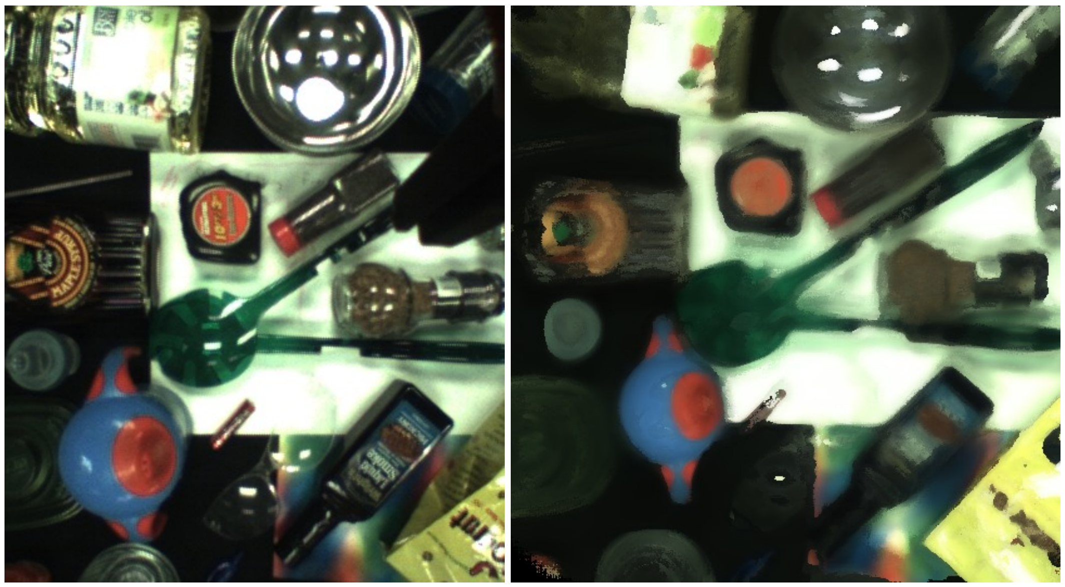
$$\underset{k}{\operatorname{argmax}}\, p(Z|k, A_k, P). \tag{3.21}$$

Here the pose x is obtained with the sliding window approach mentioned above, which assumes a constant prior on object position and configuration:

$$\underset{k}{\operatorname{argmax}}\ \max_{x,c} p(Z|k, x, c, A_k, P). \tag{3.22}$$

3.2.5 Detecting and Suppressing Reflections

Overhead lights induce specular highlights and well formed images on shiny surfaces, as well as broader and more diffuse pseudo-images on textured or wet surfaces. Vision algorithms are easily confused by the highly variable appearance of reflections, which can change locations on the surfaces of objects as relative positions of lighting sources change. We can use information contained in the light field to remove some of the reflections in an estimated map, as long as affected portions were seen without reflections from some angles.



(a) Left: An image from Baxter's wrist camera, which contains many reflections from the overhead light. Right: a synthesized orthographic photograph of the same scene with reflections suppressed.

Figure 3.6: Reflection suppression in synthetic photographs using our model.

Specular reflections on the surface of an object tend to form virtual images which are in focus behind the object. When a human looks at a shiny object or the surface of still water, they might first focus on a specular reflection formed by the object, realize it is bright and therefore not the true surface, and then look for a different color while focusing closer. To construct a map with reduced reflections, we perform an initial focus sweep to identify rays that are part of the object and in focus at one depth, and then perform a second focus sweep to identify rays that are part of a highlight and in focus at a different (deeper) depth. Specifically, we first estimate z values for the map by approximate maximum likelihood. Then, we re-estimate the z values by re-rendering at all heights while throwing out rays that are too similar to the first map and only considering heights which are closer to the observer than the original estimate. That is, the second estimate looks to form a different image that is closer to the observer than the first. The final image is formed by taking either the first or second value for each cell, whichever has the smallest variance, which is a measure of the average likelihood of the data considered by that cell. If the second estimate considered too few samples, we discard it and choose the first. Figure 3.3 shows an example image from Baxter's wrist camera showing highly reflective objects, and the same scene with reflections suppressed.

Identifying reflections using optical cues instead of colors allows us to remove spots and streaks of multiple light sources in a previously unencountered scene without destroying brightly colored Lambertian objects. The requirement that the second image form closer than the first dramatically reduces artifacts, but when considering z values over a large range some image translocation can occur, causing portions of tall objects to bleed into shorter ones. On the whole, though, when considering objects of similar height, this algorithm suppresses reflections substantially without admitting many false positives. Especially on sharply curved objects there will sometimes be a region that was covered by a reflection from all angles. If such a reflection occurs near a boundary it may aid in localizing the object. If it occurs on the inside of an object region, it will often be surrounded by a region of suppressed reflections, which are detected by the algorithm. Concave reflective regions can form reflections that are impossible to remove with this algorithm since they form complex distorted images which can project in front of the object. A notable instance of this is the metal bowl in our example.

This algorithm could be extended, but approaches which make more global use of the light field will ultimately outperform it. This is just one example of how thinking with light fields can make it easier to solve perceptual problems by forming analogies with the visual cues which humans perceive but are absent from static photographs. There are no empiracal results of this work at this time.

3.2.6 Assumptions

Our graphical model over light fields includes some simplifying assumptions that reduce the cost of inference at the expense of realism.

When performing depth inference, we allow each cell to consider all of the rays potentially intersecting it at each height without considering the paths of the rays (Section 3.2.2). This allowance is a computational approximation which we justify with the assumption that objects are not very steep and will not self occlude. This is not a fair assumption for objects in isolation, and is even less fair when objects are places close to each other. By assigning a depth label to each ray, we can make sure that it is counted in only one cell, which should lead to more realistic depth maps. Assigning labels to each ray will necessitate a different inference algorithm, and Expectation Maximization looks to be a good fit.

Another assumption that we make is that each (x, y) cell is occupied at only a single height. The approximate inference we currently perform is compatible with this assumption, but by moving to a ray labeling model, we can consider removing the single height assumption so that we can meaningfully populate voxel based maps for a true 3D reconstruction of object geometry. By reasoning about matter in 3D and progressively ignoring rays that have already been accounted for, it may also be possible to design robust segmentation and saliency algorithms that do not rely on background maps.

As noted in Equation 3.2, we are considering the cells to be independent of each other conditioned on the model parameters. This assumption is only valid for Lambertian surfaces under constant lighting and does not hold true for light sources, reflective surfaces, or on Lambertian surfaces when then global illumination changes. We can remove this assumption while continuing to use normal distributions by adding covariance terms to jointly model the cells of an object. We can do this by modeling some of the principle components of the multivariate Gaussian, which should capture global illumination changes, some structure of the shadows cast by the object, and groups of pixels with similar surface normals and reflective properties. By capturing these effects, inference should become more robust to lighting variation while becoming more expensive by a factor related to the number of principle components modeled.

When speaking of the variance of the light emitted by a cell, we are currently referring to the variance of the light as the viewing angle of that cell varies. For a single scene, this makes some sense, but if we want to capture rich information about an object, we should really capture multiple light fields for that object, register them, and calculate the statistics of synthetic photographs under those different light fields.

Chapter 4

Latent Variables in the Graphical Model

Our approach for synthetic aperture photography takes place largely in the spatial domain. Many of the interesting results in synthetic photography have been in the frequency domain. There are algorithmic advantages and insights to be had there, in a large part due to the convolution theorem. There are certain probabilistic techniques though that are straightforward to model and perform when the photography happens via ray casting rather than convolution.

4.1 Ray Labeling

When we render a synthetic photograph, we must choose a focal plane. By varying that focal plane, we can construct a stack of synthetic photographs that sweep through the space being imaged. Using the variance channel as a focus cue, we can estimate the geometry of the scene by noting the sharp pixels in the synthetic images. Each pixel in the contributing images corresponds to a ray through space. When we do a basic reconstruction, each ray contributes once to each synthetic photograph. This is fine for localization purposes, but if we start trying to assign probabilities to detections and colors, we run into overcounting issues. Thus we would really like to know the depth at which each ray originates and count it only in the synthetic photograph rendered at that depth. This would allow for a more accurate probabilistic estimate of the scene geometry.

An immediate choice presents itself, which is soft assignment versus hard assignment. Soft assignment is probably better for addressing more complex optical systems involving reflection and refraction. Hard assignment is more straightforward to investigate because soft assignment models often require additional structure, such as regularization, to avoid degeneracy. Sampling over a continous space requires more resources as well. We investigated hard assignment of depths to rays, with some perceptible improvements in both color and geometry estimation. The synthetic photographs that result from this process are *samples* drawn from the improved model where ray depth is a latent variable. We could be more accurate during inference in Lambertian cases with smooth, convex geometry by averaging the values of samples drawn from different initializations and forming the *marginal estimate* of the depth values. The marginal estimate is appropriate in the smooth Lambertian case because there is one true depth belonging to each ray. In non-Lambertian cases such as transparent and reflective objects or with geometries which include cusps, discontinuities, and non-convex regions, two rays could be arbitrarily close in position and angle but originate from different depths. It is as if one ray could take on multiple true values if sampled at different times, due to quantization and encoder noise. In the non-Lambertian and geometrically pathological cases, the marginal value could end up being very far away from any of the multiple possible true values for a ray, and so could be extremely misleading. This suggests that we really want to impose smoothness priors on geometry and keep a few samples from the model which are themselves local maxima, the *modes* of the distribution, as each would represent a solution which coherently separated one of the multiple surfaces present in the scene.

The depth to assign to a ray is the depth at which the ray is most probable according to the cells the ray passes through in the depth stack. That is, the depth at which that ray sharpens the stack the most when counted. This corresponds to a temperature 0 simulated annealing process. Better results would likely be had through full annealing.

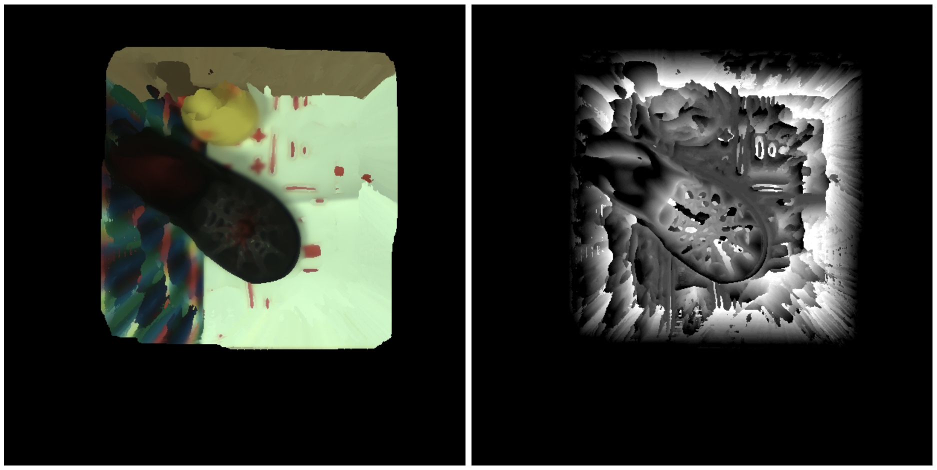
Consider Figure 4.1 and Figure 4.2. The approximate maximum likelihood estimates have large, discontinuous patches of artifacts near high image gradient regions with incorrect RGB and Z values. The marginal estimates without ray labeling are smoother in RGB and Z, but suffer from substantial blurring everywhere in RGB and hallucinate discontinuously tall borders on boundaries with high image gradients. The marginal estimates with ray labeling substantially reduce these artifacts, but still have problems in areas with less data and in monotone regions. Nonetheless, the marginal estimate with ray labeling combines some of the good properties of the other two. It captures the shape of the duck body and head, as well as the depressed region in the heel of the shoe and the spiderweb pattern on the top of the shoe, which is nearly invisible in the other estimates.

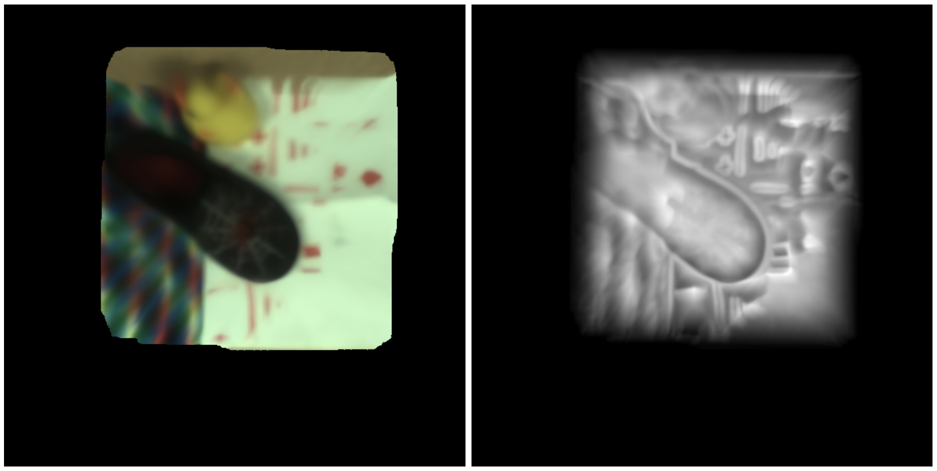
4.2 Camera Pose Re-Estimation

Baxter provides encoder values which are accurate to within $2mm$, so synthetic photography is actually useful as the *mapping* half of a SLAM algorithm. The encoders have limits, though. They reveal to us the location of the end effector relative to the robot base, but they cannot tell us if the entire robot shakes or rocks. And the robot does shake and rock, during normal operation. This is one reason why two arms mapping simultaneously produce less sharp images: two arms moving will shake the base noticeably.

Consider the following: *A first camera pose is better for its contributing image than a second camera pose if some synthetic photographs are sharper when rendered with the first pose than the second.* We can immediately see that optimizing camera poses in this fashion without constraints is ill posed. For general scenes, it is necessary to consider the underlying geometry, or else iterative adjustments will increase the sharpness by subtle means. For instance, if only a single focal plane is considered, such unconstrained optimization may move all of the camera poses toward or away from the target geometry so that more of the target comes into the considered focal plane, thereby sharpening the image. That is, if the geometry is not explicitly estimated, the geometry estimation will leak (destructively) into the camera pose estimation.

Furthermore, the quality of the geometry estimate and the relative proportions at different depths will influence the camera pose re-estimation. A safe compromise is to pick the dominant focal plane and optimize


(a) Approximate maximum likelihood estimate.


(b) Marginal estimate with no ray labeling.

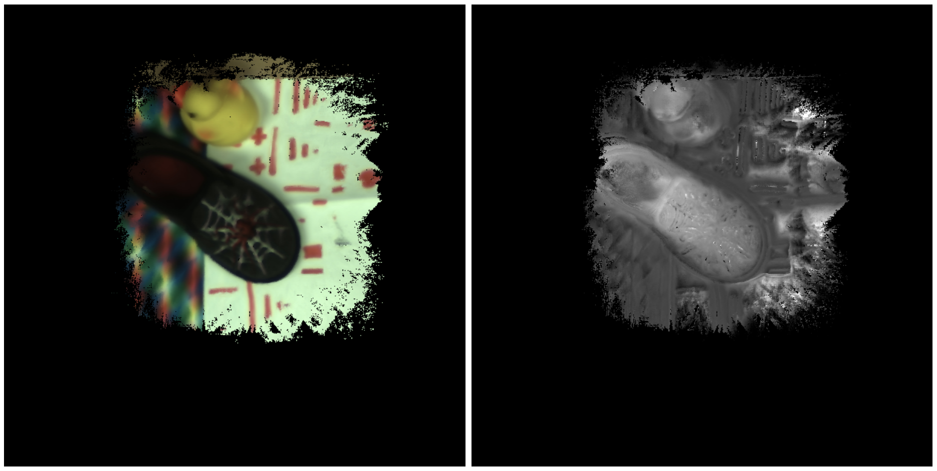
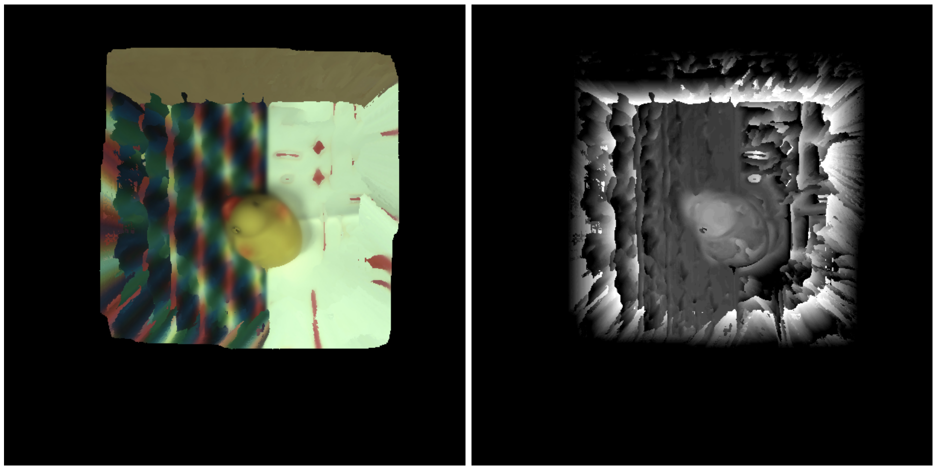
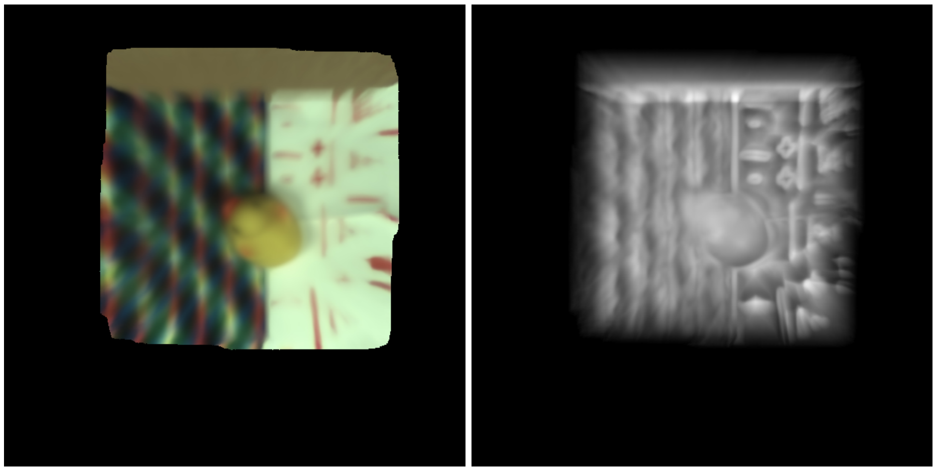

(c) Marginal estimate with ray labeling. This version is much sharper than the others and better captures the geometry of the scene.

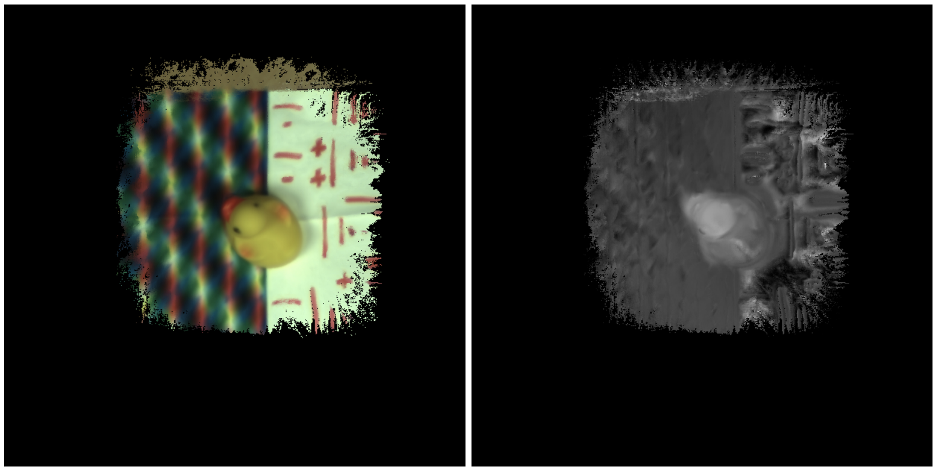
Figure 4.1: Synthetic photographs and corresponding depth maps of a tabletop scene of a rubber duck, a shoe with a spider web pattern, and several flat test patterns, rendered from the same constituent images with and without ray labeling.



(a) Approximate maximum likelihood estimate.



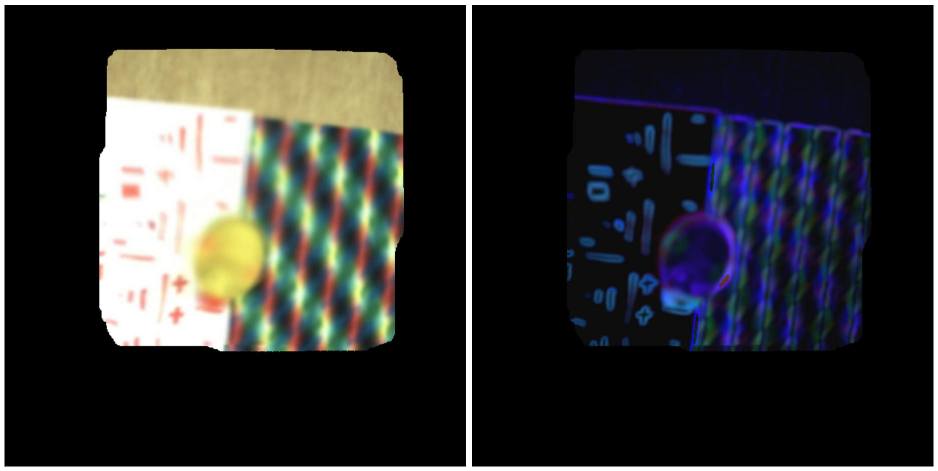
(b) Marginal estimate with no ray labeling.



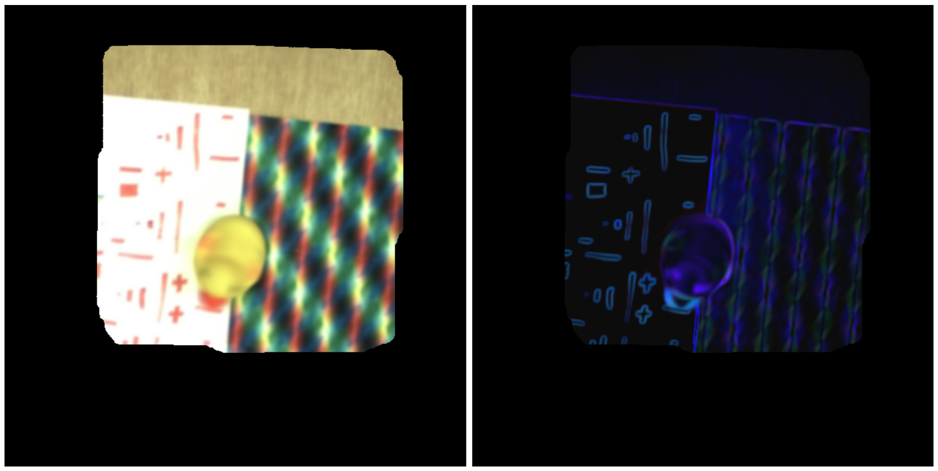
(c) Marginal estimate with ray labeling. This version is much sharper than the others and better captures the geometry of the scene.

Figure 4.2: Synthetic photographs and corresponding depth maps of a tabletop scene of a centered rubber duck and several flat test patterns, rendered from the same constituent images with and without ray labeling.

the camera poses so that synthetic cells considered at that depth are most sharp. This is like a dancer using a wall for visual guidance during a spot turn. Observe the effect of this technique in Figure 4.3, where the focal plane is set at the inferred height of the table. The image with camera pose re-estimation is much sharper. High image gradient regions are less fuzzy, the duck head and eyes are discernable despite being out of the focal plane (showing how defocus is distored substantially by incorrect camera poses), and the woodgrain and test patterns have higher frequency details exposed. This demonstration is a fair proof of concept, but a full treatment of this problem is out of the scope of this document.



(a) Synthetic photograph rendered with a single focal plane and no camera pose re-estimation.



(b) Synthetic photograph rendered with a single focal plane with camera pose re-estimation. The RGB image rendered with camera pose re-estimation is sharper and the variance of the cells is both smaller in magnitude and better confined to the areas with high image gradient, indicating that the new camera poses are more consistent.

Figure 4.3: Mean and variance synthetic photographs of a tabletop scene of a rubber duck and some flat test patterns, rendered from the same constituent images with and without camera pose re-estimation.

Part II

Planning Grasps With 2D Synthetic Photography

Chapter 5

Handling Static, Rigid Objects

5.1 Autonomous Object Manipulation

Synthetic photography allows us to create composite images which are more amenable to inference than their contributing source images due to novel viewing angles, perspectives, lighting conditions, and software lens corrections. By forming orthographic projections we can use sliding window detectors to reliably localize Lambertian objects, and our reflection suppression technique lets us cut through most specular highlights found on household goods.

The Yale-CMU-Berkeley (YCB) Object set is a physical set of objects used to benchmark performance on grasping tasks. We used these objects to benchmark our detection and grasping performance.

The YCB results were obtained with our first synthetic photography implementation. Our camera calibration procedure was less accurate and we had not yet established good practices. Our current system deserves a new evaluation, but we report these results nonetheless.

Most of these objects were shallow enough to image with a synthetic photograph rendered at table height. One object, the mustard, was very tall and so we had to render an image marginalized over focal planes spanning the table to the top of the mustard.

We evaluate our model's ability to detect, localize, and manipulate objects using Baxter. We selected a subset of YCB objects [10] that were rigid and pickable by Baxter with the grippers in the 6cm position

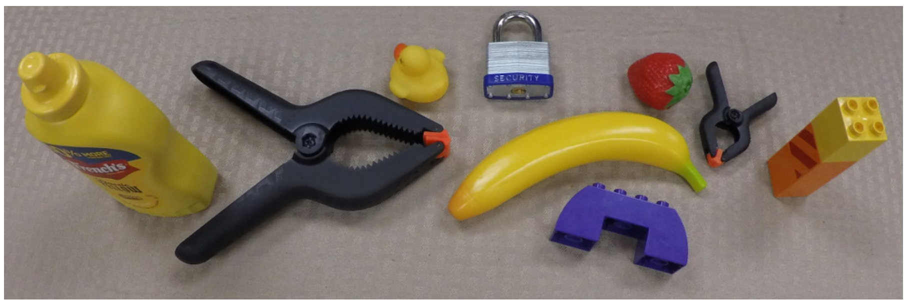
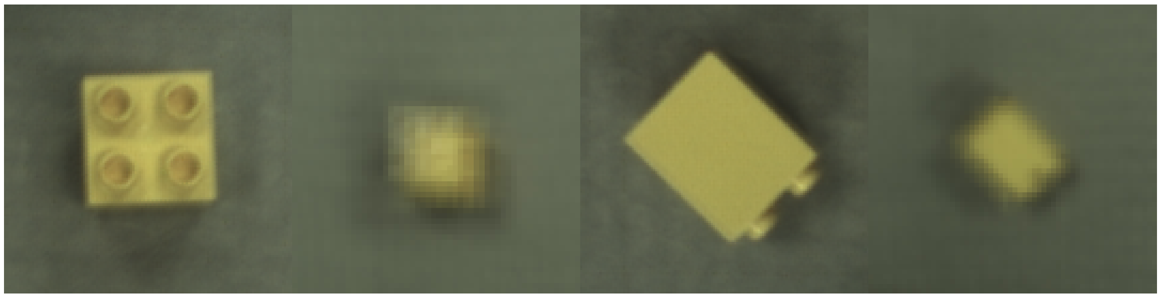


Figure 5.1: The objects used in our autonomous manipulation evaluation.



(a) Top. (b) Side.

Figure 5.2: Automatically generated thumbnails and synthetic photographs for the two configurations detected for the yellow square duplo.

as well as a standard ICRA duckie. The objects used in our evaluation set appear in Figure 5.1. In our implementation the grid cell size is $0.25\,\mathrm{cm}$, and the total size of the map was approximately $30\,\mathrm{cm} \times 30\,\mathrm{cm}$. We initialize the background variance to a higher value to account for changes in lighting and shadows.

5.1.1 Localization

To evaluate our model's ability to localize objects, we find the error of its position estimates by servoing repeatedly to the same location. For each trial, we moved the arm directly above the object, then moved to a random position and orientation within $10\,\mathrm{cm}$ of the true location of the object. Next we estimated the object's position by servoing: first we created an synthetic photograph at the arm's current location; then we used Equation 3.19 to estimate the object's position; then we moved the arm to the estimated position and repeated. We performed five trials in each location, then moved the object to a new location, for a total of 25 trials per object. We take the mean location estimated over the five trials as the object's true location, and report the mean distance from this location as well as 95% confidence intervals. This test records the repeatability of the servoing and pose estimation; if we are performing accurate pose estimation, then the system should find the object at the same place each time. Results appear in Table 5.1.

Our results show that by using synthetic photographs, the system can localize objects to within $2\,\mathrm{mm}$. We observe more error on taller objects such as the mustard, and the taller duplo structure, due to our assumption that all cells are at table height. Note that even on these objects, localization is accurate to within a centimeter, enough to pick reliably with many grippers; similarly detection accuracy is also quite high.

To assess the effect of correcting for z, we computed new models for the tall yellow square duplo using the two different z estimation approaches. We found that the error reduced to $0.0013\,\mathrm{m} \pm 2.0 \times 10^{-05}$ using the maximum likelihood estimate and to $0.0019\,\mathrm{m} \pm 1.9 \times 10^{-05}$ using a marginal estimate over z, which is a weighted sum over z values weighted by the probability of each value. Both methods demonstrate a significant improvement. The maximum likelihood estimate performs slightly better, perhaps because the sharper edges lead to more consistent performance. Computing z corrections takes significant time, so we do not use it for the rest of the evaluation; in the future we plan to use the GPU to accelerate this computation.

[1]Due to its size and weight, we used a slightly different process for this object; details in Section 5.1.2.

Table 5.1: Performance at localization, detection, and picking. Includes object height (O. Height), localization accuracy, detection accuracy (D. Accuracy), pick accuracy (P. Accuracy), and clutter pick accuracy (C.P. Accuracy).

Object (YCB ID)	O. Height	Localization Accuracy (m)	D. Accuracy	P. Accuracy	C.P. Accuracy
banana (11)	0.036 m	0.0004 m $\pm 1.2 \times 10^{-05}$	10/10	10/10	5/5
clamp (large) (46)	0.035 m	0.0011 m $\pm 1.5 \times 10^{-05}$	10/10	10/10	5/5
clamp (small) (46)	0.019 m	0.0009 m $\pm 1.2 \times 10^{-05}$	10/10	10/10	4/5
duplo (purple arch) (73)	0.031 m	0.0010 m $\pm 2.5 \times 10^{-05}$	10/10	10/10	3/5
duplo (yellow square) (73)	0.043 m	0.0019 m $\pm 2.4 \times 10^{-05}$	9/10	8/10	0/5
duplo (tall yellow square) (73)	0.120 m	0.0040 m $\pm 4.4 \times 10^{-05}$	10/10	10/10	4/5
mustard (9)	0.193 m	0.0070 m $\pm 8.2 \times 10^{-05}$	10/10	10/10[1]	0/5
padlock (39)	0.029 m	0.0013 m $\pm 3.7 \times 10^{-05}$	10/10	8/10	2/5
standard ICRA duckie	0.042 m	0.0005 m $\pm 1.4 \times 10^{-05}$	10/10	7/10	3/5
strawberry (12)	0.044 m	0.0012 m $\pm 2.1 \times 10^{-05}$	10/10	10/10	4/5
Overall		0.0019 m $\pm 5.2 \times 10^{-06}$	99/100	93/100	30/50

5.1.2 Autonomous Classification and Grasp Model Acquisition

After using our autonomous process to map our test objects, we evaluated object classification and picking performance. Due to the processing time to infer z, we used $z = table$ for this evaluation. The robot had to identify the object type, localize the object, and then grasp it. After each grasp, it placed the object in a random position and orientation. We report accuracy at labeling the object with the correct type along with its pick success rate over the ten trials in Table 5.1. The robot discovered 1 configuration for most objects, but for the yellow square duplo discovered a second configuration, shown in Figure 5.2. In the future we plan to explore more deliberate elicitation of new object configurations, by rotating the hand before dropping the object or by employing bi-manual manipulation. We report detection and pick accuracy for the 10 objects.

Our results show 98% accuracy at detection performance for these objects. The duplo yellow square was confused with the standard ICRA duckie which is similarly colored and sized. Other errors were due to taller objects. The robot mapped the tall duplo structure in a lying down position. It was unable to pick it when it was standing up to move it to its scanning workspace because of the error introduced by the height. After a few attempts it knocked it down; the lower height introduced less error, and it was able to pick and localize perfectly. The padlock is challenging because it is both heavy and reflective. Also its smallest dimension just barely fits into the robot's gripper, meaning that very small amounts of position error can cause the grasp to fail. Overall our system is able to pick this data set 84$\%$ of the time.

Our automatic process successfully mapped all objects except for the mustard. The mustard is a particularly challenging object due to its height and weight; therefore we manually created a model and annotated a grasp. Our initial experiments with this model resulted in $1/10$ picks due to noise from its height and its weight; however we were still able to use it for localization and detection. Next we created a new model using marginal z corrections and also performed z corrections at inference time. Additionally we changed to a different gripper configuration more appropriate to this very large object. After these changes, we were able to pick the mustard $10/10$ times.

5.1.3 Picking in Clutter

Finally, we assess our system's ability to pick objects in clutter. We divided the 10 evaluation objects into two groups of five, with the yellow square duplo and tall yellow square duplo always in different groups since they include the same yellow piece. Then for each object in each group, we instructed the system to pick that object. After each pick we randomized the scene, resulting in a total of five picks for each object. This task corresponds to following a command such as "Find me a banana," where the language specifies what object to pick, and the system must find the associated object amidst other distractor objects. We report pick success rate in Table 5.1. We increased our system's search depth D to $500,000$ for this task, because in clutter there are many possible distractor objects. This evaluation shows that we are successful at detecting and localizing even in the presence of distractor objects. The mustard continues to perform poorly because of its height and weight. The yellow duplo on its own performed poorly because of confusion with other yellow objects. Overall we successfully picked in clutter $30/50$ times, demonstrating the efficacy of our approach in richer environments.

5.1.4 The Million Object Challenge

The Million Object Challenge (MOC) is a pursuit designed to illustrate some concepts in the pursuit of effective domestic robotic assistants.

The MOC seeks to quantify the rate at which data about objects can be gathered. We do so by demonstrating a robotic system which can autonomously separate objects from an input pile one-by-one, examine, manipulate, and eventually store them away. This can be thought of as a "for loop" over objects. Once an object is separated, it can be inspected by whatever learning subroutine the implementer codes. Our initial subroutines try to collect enough views of each object to allow object detection, pose estimation, and grasp proposal to succeed with high probability, enabling a system that can detect failure modes in the field and add parameters to its model to eliminate them.

We gauge our progress by our capacity to *scan* or *map* objects. The number "one million" was chosen because, given the data volumes that are necessary for state of the art algorithms, it seems likely that if one million household items were mapped, there would be enough data to create category level detectors sufficient to enable a domestic manipulator robot to handle most objects in most people's homes most of the time out of the box. The performance required to enable one million objects to be scanned in the first place would likely be fast and flexible enough to allow the robot to handle the edge cases it will inevitably encounter in situ in a reasonable amount of time.

Since our algorithmic choices and therefore specific data requirements change over time, we like to measure our progress not only by the number of objects we have scanned, but by the rate at which we can scan objects. Similarly, the definition of a scan is malleable, as a scan could include a large number of manipulations or validation steps, depending on the application. Thus we also like to consider the rate at which we can perform picks on novel objects and the number of total picks we have performed.

The number of distinct objects scanned to date is in the low hundreds, the number of scans performed is around a thousand, and the number of picks performed is in the high tens of thousands.

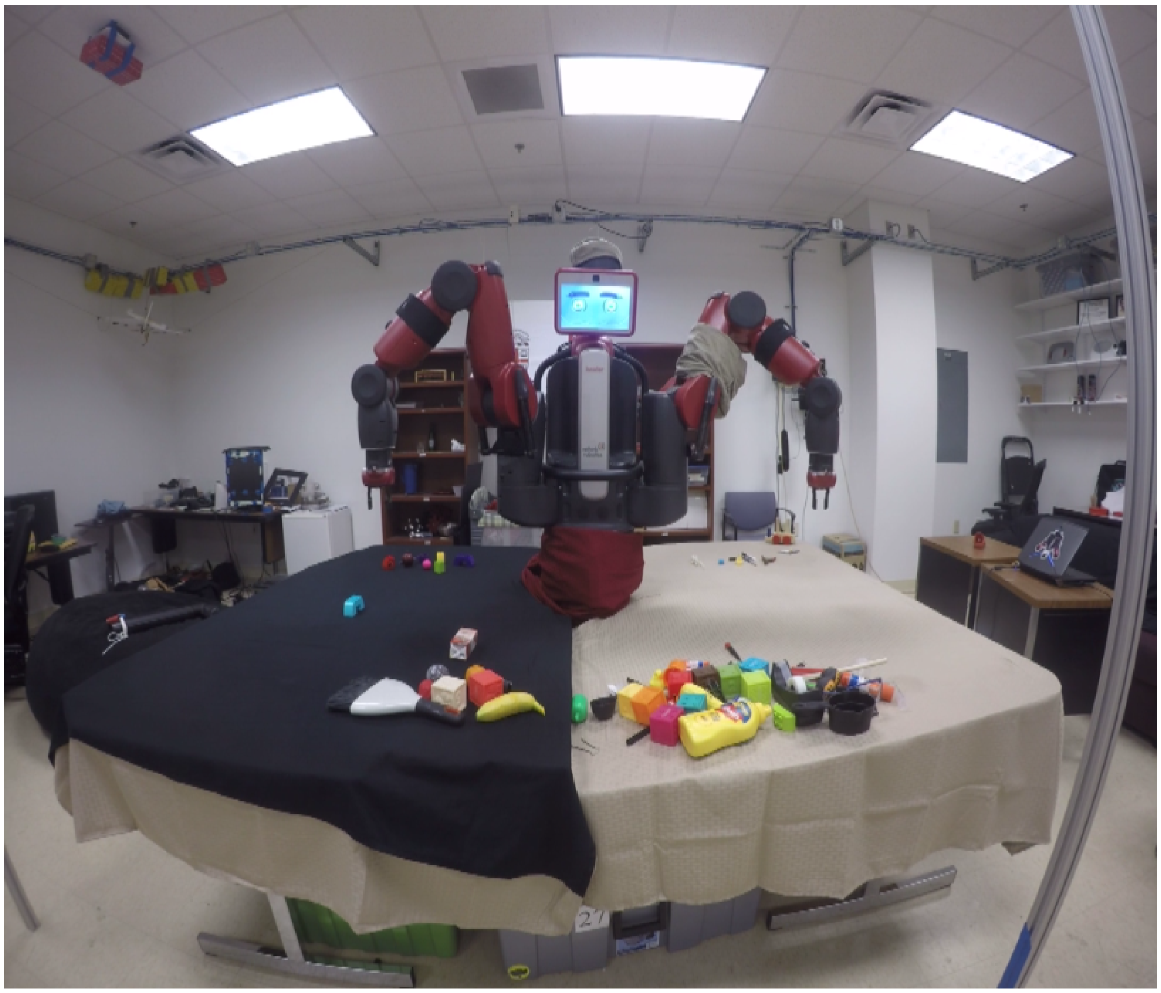
We call our automatic scanning procedure the *Infinite Scan*. The current implementation requires a structured environment of three workspaces with with known backgrounds so that background subtraction can be performed. An *input workspace* holds the objects to be examined, a *play workspace* is initially empty and serves as a place in which to isolate objects for examination, and an *output pile* acts as a repository for finished objects. A system employing depth cues rather than background subtraction would work in less structured environments like homes, offices, and factories, and allow recovery from failure in situ. Baxter executes the Infinite Scan with all workspaces visible in Figure 5.3.

Baxter is one of the robots selected to be in in a traveling exhibit called "Robots: The 500-Year Quest to Make Machines Human". "Robots" initially opened in the London Science Museum on February 7, 2017 with plans to travel to other museums for the next 5 years. There are approximately 20 different robots in the exhibit, and some of them are programmed to coordinate with a human performer during certain events. There is a specially designed table upon which Baxter conducts the Infinite Scan. The table is painted black matte, and the three workspaces on each side are demarcated by raised boundaries. Baxter stands through a hole in the center of the table, and the two are enclosed in a plexiglass cage to keep visitors from interfering.

We have set up a database which stores the data collected by the Million Object Challenge Infinite Scan program. We store calibrated camera images for each object (and each object component), including camera pose annotations, so that we can reconstruct the light field of the object and synthesize photographs appropriate for different tasks. By storing the images themselves, we maintain maximum flexibility for future applications, and need not commit to any particular representation. This allows us to apply algorithms developed in the future to data collected in the past.

As a demonstration of how the data can be used, there is a project in progress, called *Automatic Category Generator* or *ACG*, which uses data collected by the MOC to automatically generate category level object detectors from collections of labeled object instances. For the first implementation, we will use Infinite Scan to collect instance level models of around 50 wooden spoons. The ACG program will then download all of the wooden spoon models and train a detector for wooden spoons, which will incorporate one or more latent variables to account for the scale of the spoon. In a second implementation, we will incorporate multiple categories, scanning hundreds of object instances and storing them in the database. We will then create an Amazon Mechanical Turk task which will use humans in the loop to assign title and attribute labels to the scanned object instances. The second generation ACG program will accept title and attribute labels and generate a query to the database. The query will return a number of object instances, which will be used to automatically generate a category level object detector for the query, which can be used to localize and pick objects of the specified type. These queries could be issued on the fly by an intelligent system which understood spoken natural language commands and could translate those commands into database queries, creating a more complete human-robot interaction facility.

A mature version of the database, together with pre-trained stock category detectors, could accompany a domestic robot during initial deployment. The robot could then scan objects in its own environment, use those instance based models for those objects, and add those models to the database to improve the stock category level detectors.



(a) The input workspace for each arm is furthest away from the camera, the play workspace is in the middle of the table, and the output workspace is closest to the camera.

Figure 5.3: The Baxter robot executing Infinite Scan.

5.2 Domestic Experiments

A goal of our lab is to create a robot that is capable of performing useful domestic tasks in real homes. We were not aware of a framework that would allow robots to perceive and act in this setting. In this chapter, we'll cover some tasks that we contrived in order to advance the state of the art in domestic object manipulation. Our ability to autonomously iterate through a pile of diverse objects (MOC applied to the YCB objects) might convince you that we can handle Lambertian rigid bodies strewn across a carpet floor or wooden tabletop. But a real home contains shiny objects in shiny places with additional task goals and constraints beyond basic pick and place. Here we suggest that we have a handle on some of these next generation tasks and subtleties, and we hope that we have presented pieces that convince you that a path to a useful domestic robot is if not short, at least defensibly existent.

5.2.1 Picking Underwater Silverware

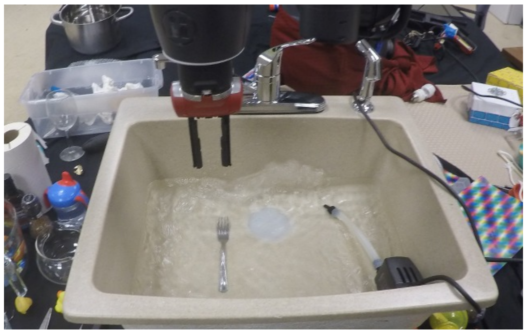
Many vision tasks could be made easier through the use of synthesized images; however, we wanted to demonstrate a task which would usually be considered difficult to perform through individual RGB camera images or with an IR depth sensor, but is rendered nearly trivial by synthetic photography: picking a metal fork from running water.

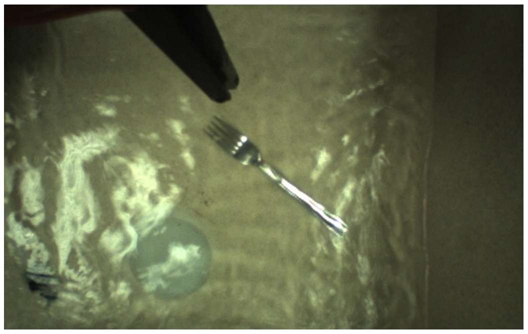
We filled a sink with several inches of water and used a pump to induce an appreciable current. The current maintained turbulence across the water surface, which generated a substantial number of shadows and reflections in the moving water and surrounding surfaces, shown in Figure 5.4a. Next we placed a metal fork in the sink under the flowing water. We used synthetic photographs focused at the bottom of the sink to successfully localize and pick the fork 24 times out of 25 attempts with Baxter's 4 cm gripper. There were no 180 degree confusions, and the single miss was pinched but not lifted. The experiment was repeated a year later with a more secure pair of gripper fingers and succeeded 50 out of 50 times.

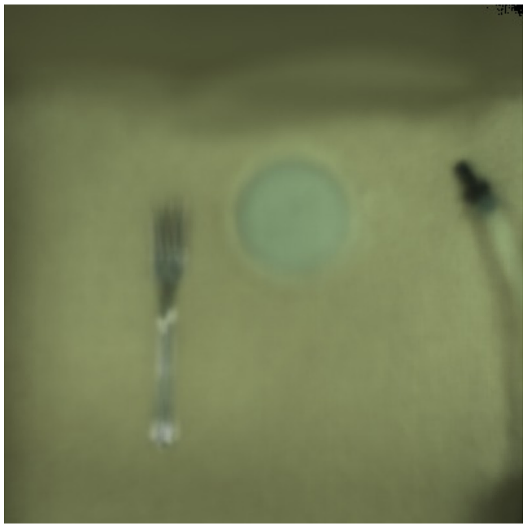
The constant presence of reflections and shadows make it challenging to pick the fork from beneath the water based on individual images. Bright edges are present at all scales, so gradient based methods will have trouble, and the average intensity of the fork is similar to the average intensity of turbulent water, so blurring the scene will significantly degrade the signal. A stronger signal of the bottom of the sink may be obtained by leaving the camera still and averaging incoming images. This may eliminate reflections and shadows, but it only provides one perspective of the fork, which is prone to noise due to fixed reflections. By synthesizing images which are focused at the bottom of the sink, we get the benefits of averaging images from a still camera combined with the robustness of multiple perspectives. Neither water nor metal is particularly conducive to imaging with IR depth cameras, making it harder to solve this problem with such sensors.

5.2.2 Screwing A Nut Onto A Bolt

As a test of our system's accuracy, we programmed Baxter to screw a nut onto a $0.25''$ bolt. The robot first localized a nut and the bolt on the table. Next it used an open-loop program to pick up the nut and place it on the bolt, given these inferred locations. The robot was able to perform this task several times, showing that


(a) Robot preparing to pick a metal fork in running water.


(b) Wrist camera view of a fork under flowing water.


(c) Rendered orthographic image of the fork; reflections from water and metal are reduced.

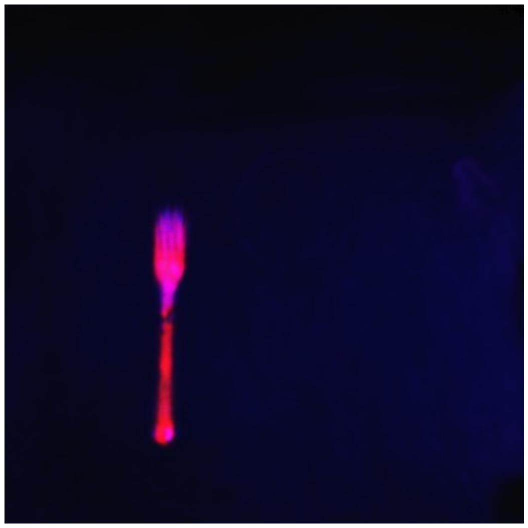
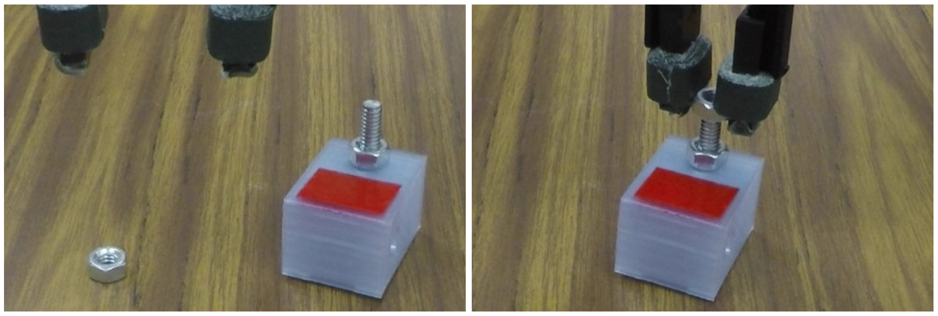

(d) Discrepancy view, showing the robot can accurately segment the fork.

Figure 5.4: Our approach allows a robot to detect, localize and manipulate under challenging non-Lambertian conditions.



(a) Initial scene. (b) Screwing the nut.

Figure 5.5: Our approach allows us to use an RGB camera to localize objects well enough to screw a nut onto a $0.25''$ bolt.

the very precise localization enabled it to robustly and accurate pick up the nut and place it in the bolt, even though both objects are very small and the accuracy required is near the limits of the Baxter robot.

5.2.3 Forks and Glasses

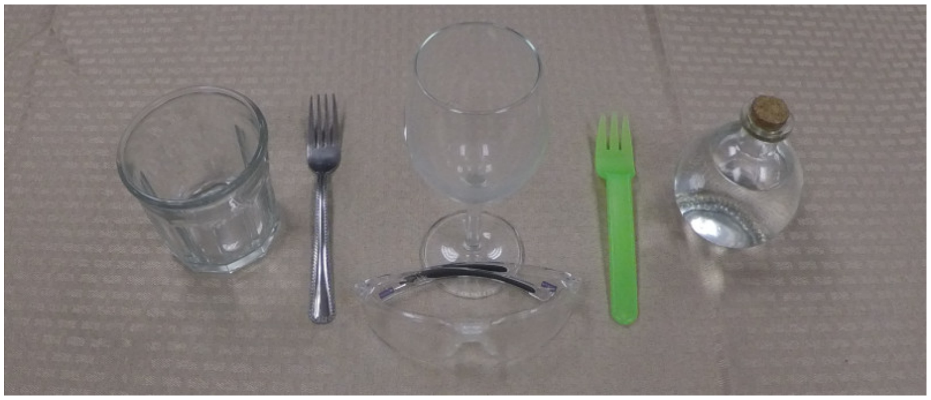
For each trial, we moved the object to a random location on the table within approximately $25\,\mathrm{cm}$ of the arm's starting location. Then we localized the object using the wrist camera and picked it. We used two different modes when localizing. For the Point Scan, the wrist camera used an average of 40 images taken at the arm's starting location. We verified that the object was always in view of the wrist camera when this image was taken, although part of it may have been occluded by the gripper. In the Line Scan, we moved the arm $28\,\mathrm{cm}$ back and forth over the workspace to make a synthetic photograph using about 140 images. Next we estimated the object's position using image matching in the synthetic photographs and grasped it. Results appear in Table 5.6d.

Notably, the Lambertian green fork was picked every time with both scans; even the Point Scan was able to successfully localize the object. However, when using the Point Scan to localize a similar reflective object, the robot frequently missed due to different appearances of the object depending on its location on the table and reflections from the overhead lights. However when the robot used our model, performance returns to $10/10$, comparable to the Lambertian object. These results demonstrate that our approach enables the robot to average away the reflections by taking into account the light field information.

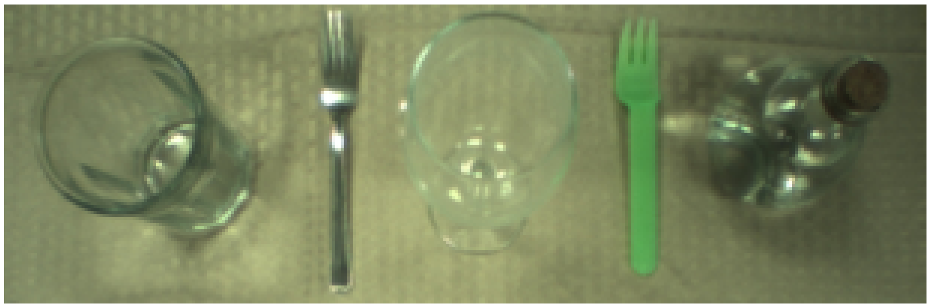
Additionally our approach is able to pick many transparent objects. For our evaluation we focused at the table height (without doing the z inference), enabling the robot to pick out the base of the object reliably, leading to more successful picks.

5.2.4 50k Picks of a Toy Car

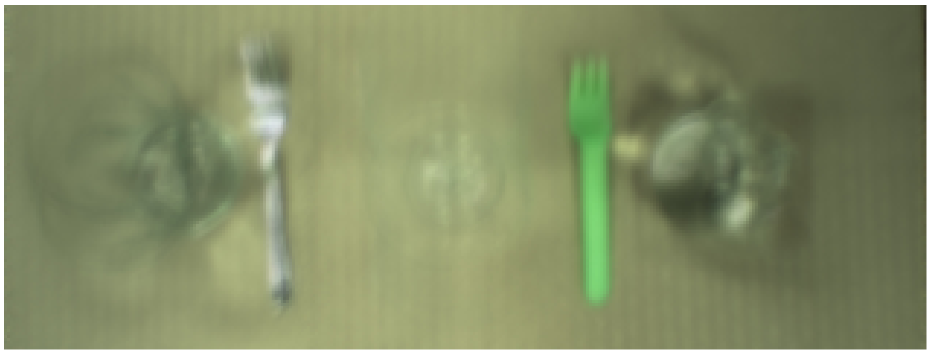
If you are going to clean a house you need to be able to pick up children's toys. Believe it or not, these are some of the weirder and more pathological objects you will find in a home. Most of the objects we use do not roll around, but kids use balls and cylinders of all sorts, and irregular objects with no obvious top or



(a) Evaluation objects from left to right: glass tumbler, metal fork, wine glass, safety glasses, green fork, and glass flask.



(b) Point scan synthetic photographs of evaluation objects using 32 wrist camera images from the same point in space. This scan shows how taller objects vary in appearance and specular highlights move dramatically as viewing angle changes, which both cause template matching with a single viewing angle to fail.

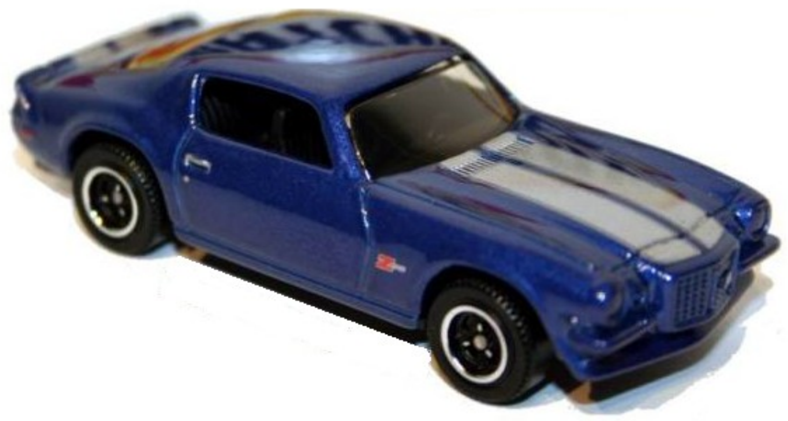


(c) Line scan synthetic photographs of evaluation objects using 143 wrist camera images taken from points uniformly distributed on a 28 cm line. The synthetic aperture is wide (the entire 640×400 image), inducing a narrow depth of field and bringing objects at table height into focus. The bases of the transparent objects appear consistently disk-like regardless of viewing angle. Specular reflections are more consistent and evenly spread across the shiny surfaces because points are viewed from many angles. Note the perspective variance from left to right.

Object	Point Scan	Line Scan
Green Fork	10/10	10/10
Metal Fork	5/10	10/10
Glass Tumbler	7/10	10/10
Wine Glass	3/10	10/10
Safety Glasses	8/10	9/10
Glass Flask	3/10	8/10

(d) Picking performance.

Figure 5.6: Evaluation objects and picking performance for point versus line scans.



(a) The wheels and non-Lambertian surfaces make this toy car difficult to pick.

Figure 5.7: Matchbox car that Ursula picked $49,500$ / $50,000$ attempts.

bottom, stuffed animals, moving parts, with so much variation in color, specularity and shape to keep them from getting bored.

Consider this Matchbox toy car. It has four functional wheels and will scoot across the table if you knock it with any force at all. The entire top is shiny, some of it is transparent, and it has planar and curved faces in multiple directions. With the wheels on the ground, it's not much narrower than the inside of the 4 cm grippers on Baxter. This is a fairly challenging object for a robot to pick, appearing in Figure 5.7.

There is a subroutine called Infinite Dribble that we use to observe the robot's picking behavior. The robot looks at the scene and tries to find the best rectangular grasp across specified dimensions, using background subtraction and silhouettes. It picks the object, places it in a random (x, y, θ) perturbation from the center of the workspace, and repeats. We configured Ursula to perform Infinite Dribble on the toy car. It took between one and two minutes for an iteration and she stayed up and running for between one and two months. She attempted more than $50,000$ picks, succeeded in more than $49,500$ of them (during a string of which a child had hidden the car from view), and had a streak of over 5000 consecutive successful picks.

Part III

Hand-Eye Coordination

Chapter 6

Ein

Programming robots to perform complex behaviors relies on powerful high-level abstractions combined with the necessity to get many low-level details correct, ranging from applying the correct coordinate transform before moving to grasp point to finding a sequence of poses to perform a behavior such as sweeping. Solutions to these problems rely on specific details of the robot hardware and the operating environment: the motions that a PR2 might use to sweep a table are quite different from that used by Baxter. To increase a programmer's speed and accuracy at getting these details right, we present a programming framework for interactively creating behaviors on a robot. Our approach uses a stack-based postfix programming paradigm combined with a read-eval-print loop to enable interactive development of complex behaviors at the command line. By abstracting algorithmic code into modular commands, taking anonymous arguments from the stack, and using defaults derived from the robot's current physical state, our approach allows the programmer to interactively try behaviors and see them immediately execute on the robot. Our approach makes the order of operations at execution time explicit, important when execution involves serial movements of a robot in physical space to manipulate objects and gather sensor information. We demonstrate that our approach enables fast and efficient creation of complex robot behaviors on the Baxter robot, ranging from a scanning procedure to automatically acquires models of objects from a pick-and-place stack for object delivery to programming the robot to perform dance movements for a song. For a few specific behaviors, we compare a program written using a conventional ROS and Python approach to a program written in our language and demonstrate that the interactive approach uses many fewer lines of code, and also takes much less time to create the behavior.

6.1 Introduction

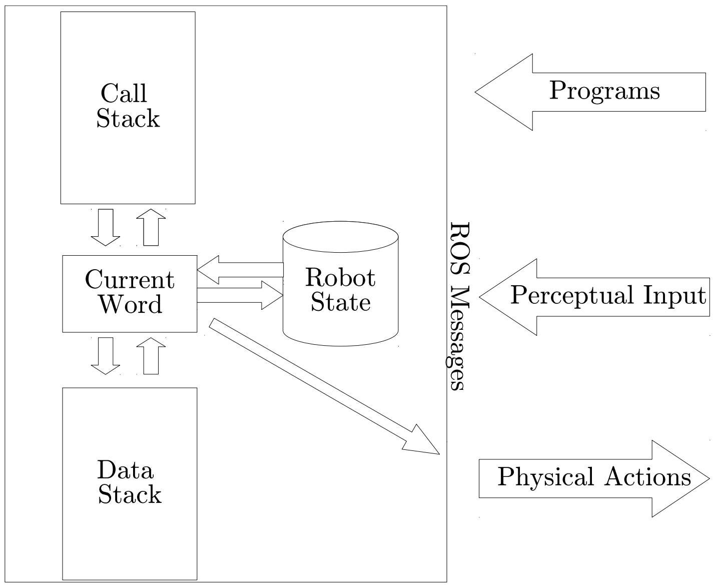
Programming robots requires the ability to create powerful abstractions combined with rigorous attention to thousands of tiny details. Getting these details right is critical to robust and reliable task completion; yet it is challenging to create a unified framework for learning and programming robots to perform this task automatically. A significant part of every robot project is long hours and late nights putting hours on the robot to program, test, and debug all of these details.

Approaches such as ROS [49] or LCM and Libbot [24] ease the programming burden by providing infrastructure for message passing, software sharing, and visualization, but still require programs to be written, usually in Python or C++. Higher-level languages such as temporal-logic [32] provide formal guarantees and very-high-level task specifications but give up control over low-level behavior. Natural language interfaces [60, 38] similarly provide high-level specifications while sacrificing control over low-level behavior. Learning from demonstration [2] provides the ability to teach behaviors but it is difficult to interleave sensing behaviors to create feedback controllers or paramaterized behaviors.

To address these issues, we propose an interactive approach to programming robot behaviors at the command line using a stack-based postfix language similar in spirit to Forth. Our approach enables a programmer to quickly create programs for different high-level behaviors, including feedback behaviors such as servoing. Because our approach relies on interactive development, it is easy to test partial behaviors on the physical robot hardware and sensors, for example by moving an arm to a particular pose, running computer vision preprocessing, and then intiating servoing with different parameters. It is easy to start in the middle of a complex sequence of behaviors, saving on development time. Tasks such as sequencing behaviors, choosing parameters and assessing the noise of a sensor at different distances all require the same sorts of interactions with the robots as a user might interact with an operating system at the command line.

Our approach enables fast and efficient programming to get the details right in the real world using a read-eval-print loop using three key ideas. First, most commands require very few arguments to perform useful behavior, making them easy to string together into more complex behaviors; we achieve this using a stack for anonymous arguments. Many commands perform useful and interpretable behavior with zero arguments. Second, we use postfix notation so that argument computation occurs before function execution. Since computing arguments to a command for a robot often requires the robot to move to a particular pose first (for example, to take an image of an object to localize it before grasping), argument execution order matters, and our approach makes this order explicit. Third, we use an interpreted approach that abstracts away the underlying publish/subscribe architecture. This approach allows robotic behaviors to be specified as linear recipes with just a few loops and branches, without requiring multithreaded programming or finite state controllers buried in Python or C++ classes. It is easy to wait until a condition is true without blocking on publishing or subscribing to new messages or to write a visual servoing loop to localize an object without requiring multithreaded programming with its inherent challenges.

We demonstrate that our system enables fast development of robotic behaviors using our language. The programmer can interactively experiment with behaviors, quickly observe them on the physical robot, and then modify the behavior, all on the command line. Our approach bears some resemblance to the G-code [3] that controls many CNC devices such as 3D printers, but contains higher level logic like that found in the URBI [4] universal robotic framework. Unlike either G-code or URBI, our system is designed to facilitate interactive tasks where the executing code may change itself or be changed by others significantly during run time. **(author?)** [8] created a system for the PR2 to execute complex behaviors using hierarchical concurrent state machines. This approach enables the specification of concurrent high-level task specific behavior Our previous work used this architecture for enabling Baxter to learn to grasp objects using a bandit-based approach but did not describe or discuss the robot programming framework [43]. **(author?)** [26] created



(a) Our system takes sensor messages from robots as well as programs as input over ROS. It outputs messages that command the robot to take specific physical actions. Program execution occurs when a word is popped off the call stack, executed, possibly modifying the call stack, data stack, and robot state, as well as commanding physical actions.

Figure 6.1: The information flow in Ein.

an abstraction for programming a robot with screen shots for untrained users. They abstract away lower-level message passing in a scripting approach, but their system is designed for end-users rather than robot developers.

We describe a programming approach that enables fast and intuitive interactive development of complex robot behaviors. Our contribution is a stack-based postfix programming framework that enables the abstraction of robot behavior and logic from lower-level event processing code. This approach makes it easy to interactively write, test, and debug new behaviors, as well as to initiate behaviors part-way through a program without running the whole thing from the beginning. We demonstrate our approach enables fast development of complex behaviors on the Baxter robot.

In the future, we aim to extend our approach to additional robots, including the PR2 as well as non-robotic agent control, for example controlling a Minecraft agent. Additionally we plan to release our framework for Baxter to enable fast development of applications for users of this popular robotic platform.

6.2 Technical Approach

Our implementation of this approach is realized as a single C++ program whose architecture appears in Figure 6.1. However the programming language and read-eval-print loop abstraction could be realized on any robotic message-passing frameworks: our contribution is a design pattern for interatively programming behaviors on robots. Perceptual information and programs are received using ROS messages (or more generally, whatever underlying message passing system is in use). These messages are processed in the background in between command execution. The only job of a low-level perceptual callback is to update the corresponding variables in the robot state, which then remain in memory for commands. The bulk of processing consists of executing commands in sequence. We use a stack-based framework, where commands to be executed are pushed onto a call stack. Each command modifies the system's physical and programmed state.

6.2.1 The Back Language

Back is a pure postfix language, in the spirit of Forth [47], Reverse Polish Lisp (the language used in HP calculators) [48] and PostScript [46]. A Back program consists of a sequence of tokens, called *words* following Forth terminology. Program execution occurs when a word is popped off the call stack, executed (which may push additional words on the call stack, change the data stack, and also read and write to the robot state representation.) Words can be either command words or data words. Data words that appear on the call stack push themselves on the data stack when executed. Back contains data word types for integers, doubles, booleans and strings, as well as robot-specific types such as eePose representing the position and orientation of the robot's end effector. Command words perform some computation when executed. For example, a short Back program is `1 1 +` . After being executed, the data word `2` will be pushed on the data stack; this execution path is illustrated step-by-step in Figure 6.2. "Hello world" in Back is ``` ``Hello world'' print ```.

Variables can be stored using the `store` word which takes the value to be stored as well as the name on the data stack. For example ``` 1 ``x'' store ``` will store the data word `1` in the variable named `x`.

Like RPL, Back contains first-class functions, which we term *compound words*, which are created using the special words `(` and `)`. The open-parentheses word puts the machine in a special compile mode, in which subsequent command are popped off the call stack until it reaches the corresponding close parentheses. At that point, the close parentheses word creates a new compound word with the corresponding contents and pushes it on the data stack. Higher-level programming structures such as `if` and `while` are implemented in terms of compound words. Additionally, new command words can be easily defined by storing compound words in variables. For example, ``` (1 +) ``inc'' store ``` defines a new word, `inc`, which increments the value at the top of the data stack by one. Using compound words, we can define new language primitives and control structures, described below.

if This word takes a Boolean condition and a compound word on the data stack. If the condition is true, it executes the compound word; if it is false it skips it. For example, ``` 1 ``One'' if ``` will leave "One" on the data stack.

ifte If/then/else takes a Boolean condition and two compound words on the data stack. If the condition is true, it executes the first compound word; if it is false it executes the second one. For example, ``` 1 ``One'' ``two'' ifte ``` will leave "One" on the data stack.

while While takes a compound word which must push a Boolean value on the data stack when executed, and a second word with no contract. It executes the second word repeatedly until the condition is satisfied.

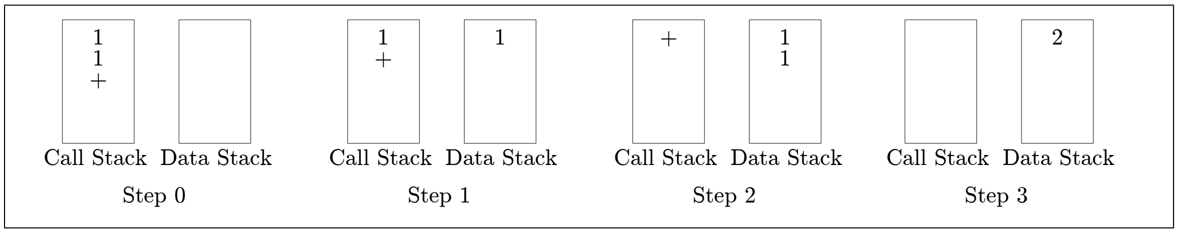


Figure 6.2: Sample execution of a simple Back program. In between each word being executed, robot messages are processed and state is updated.

```
( 1 )
( pickFromInputPile
  playWithObject
  moveToOutputPile )
while
```

Figure 6.3: Back program to pick an object from the input pile, collect data about the object, and then move it to the output pile.

A key property of Back, derived from its postfix nature, is that virtually all computation is written down in the program in the order that it occurs. This property makes scripting robot behavior very easy because the programmer simply writes down the "recipe" for the robot to follow, defining words for each step in order. For example, when the authors created a program to enable the robot to scan objects that consists of the following three words: `pickFromInputPile playWithObject moveToOutputPile` executed in an infinite loop, shown in Figure 6.3. This recipe can be written down directly without the need to define a finite controller or worry about interrupts for message processing or threading. Each of these words then goes on to implement the desired behavior in terms of other words, following traditional programming abstractions.

Back contains a number of advanced features which are beyond the scope of this document. One such feature is *multithreading*, an example of which can be found in Figure 6.4.

6.3 Baxter Case Studies

Quantitatively comparing the effectiveness of different programming approaches is challenging. It is expensive to hire programmers to implement various robotic behaviors using different paradigms, and large parts of the variation are due to differences in the programmers rather than the programming language and paradigm.

To evaluate our approach, we present a series of case studies demonstrating the design approach for various diverse capabilities. For the first case study, we describe two parallel implementations, one using ROS Python, the current most popular high-level interpreted language for robotics, and one using our approach. Additionally we describe the design and implementation of two other high-level projects using Back, demonstrating its flexibility and power to enable a variety of complex robotic tasks.

```
(
  70 30 fixCameraLightingExposureGain
  { |B
    0.1 "gazeDelta" store
    { |B
      ( 1 ) (
        blink_lightshow
      ) whileCollapsed
    }
    { |B
      (
        playWorkspace catScan2
      ) 5 replicateWord
    }
    { |B
      ( 1 ) (
        0.5 setHeadPanTargetSpeed
        drand48 20 times spinForSeconds
        endStackCollapseNoop
      ) whileCollapsed
    }
    { |B
      ( 1 ) (
        publishWristViewToFace
      ) whileCollapsed
    }
    { |B
      ( 1 ) (
        isGripperGripping (
          nod
        ) ift
      ) whileCollapsed
    }
  } ( 1 ) ( slip ) whileCollapsed
) "scan_threaded_catScan_wristFace" store
```

(a) Each `{ |B }` scope denotes a thread. Each thread is a compound word carrying its own data and call stacks. Each time the `|B` word executes, the stored thread stacks are unwound onto the real stacks, one word is executed from the call stack, and the thread stacks are rewound into the thread word, leaving the true stack contents outside of the thread unchanged. The `slip` word pops a word from the data stack and pushes it on the call stack. The outermost (master) thread scope contains the work threads, and the outermost `whileCollapsed` continually slips the master thread back onto the call stack, where it executes (in turn executing each of the inner (slave) threads) and returns to the data stack. The slave threads are stored in the master thread's data stack. This execution model sequentially executes one word in each slave thread each pass through the main loop, providing some functionality of a time sharing multithreading environment.

Figure 6.4: Back program demonstrating the use of threads.

6.3.1 Raster Scanning

As a comparison, we implemented a raster scanning behavior as a conventional ROS Python program, which we had code-reviewed by ROS experts, and compare the implementation to a Back program. The goal is to move the robot's arm in a $10cm \times 10cm$ grid pattern to collect measurements from its range sensor in the scanned region. This scan supports grasping objects, for example by running a linear filter to identify geometrically feasible grasp points in the resulting point cloud. It can also be used to collect pose annotated images to express a light field.

The Python version interleaves issues at multiple levels of abstraction: it must repeatedly stop and wait for new sensor information and for movements to complete, interleaving this logic with program logic. The Back version, in contrast, directly represents the desired behavior, looping up and down in a grid, waiting for sensor input, and then moving to the next location.

Additionally, the Back version of the scan is defined in terms of a new high-level word which is easy to plug into even larger programs. The Python version, in contrast is buried in specific decisions about the message-passing architecture. The programmer must choose between implementing a message-passing interface over the ROS API, which requires a large amount of overhead designing and synchronizing messages, or embedding more code in the Python program, forcing them to use the same message-passing discipline.

6.3.2 Object Mapping

We originally created this framework to enable the creation of a software stack that enables Baxter to scan objects at scale. The Million Object Challenge object iteration code (as described earlier in this thesis) fulfills this goal.

The Infinite Scan is a super-routine composed of a series of actions which the robot can train and execute independently, and then string together in real time to perform methodical, closed loop handling of sequential objects. The robot checks its gripper and the workspace to verify its the results of its actions. If it drops an object in transit, it will notice and start over. If it cannot grab an object to remove it from the workspace, it will try to flick the object out of the workspace. There are many sub-routines which were developed to catch edge cases and rare failures. Each edge case is treated like a new task; a detector is built to catch the event and a *handler* is constructed, manually, by a human, to be invoked at the time of catch. These handlers rely on feedback from sensors on the robot and depend on behavior that is highly specific to the robot. It is therefor difficult to program the correct handler *from the armchair* as it were, because the interaction between the mechanics of the arm and the matter of the universe are not entirely predictable. In situations such as these, the tele-op capabilities of Ein and the direct transcription of tele-op routines to automatic programs both prove indispensible. If manual tailoring of robot algorithms could not be done so quickly and precisely, it would not be possible for one person to program such a complicated and reliable system in any reasonable time frame or perhaps at all. Baxter performs Infinite Scan in Figure 6.5.

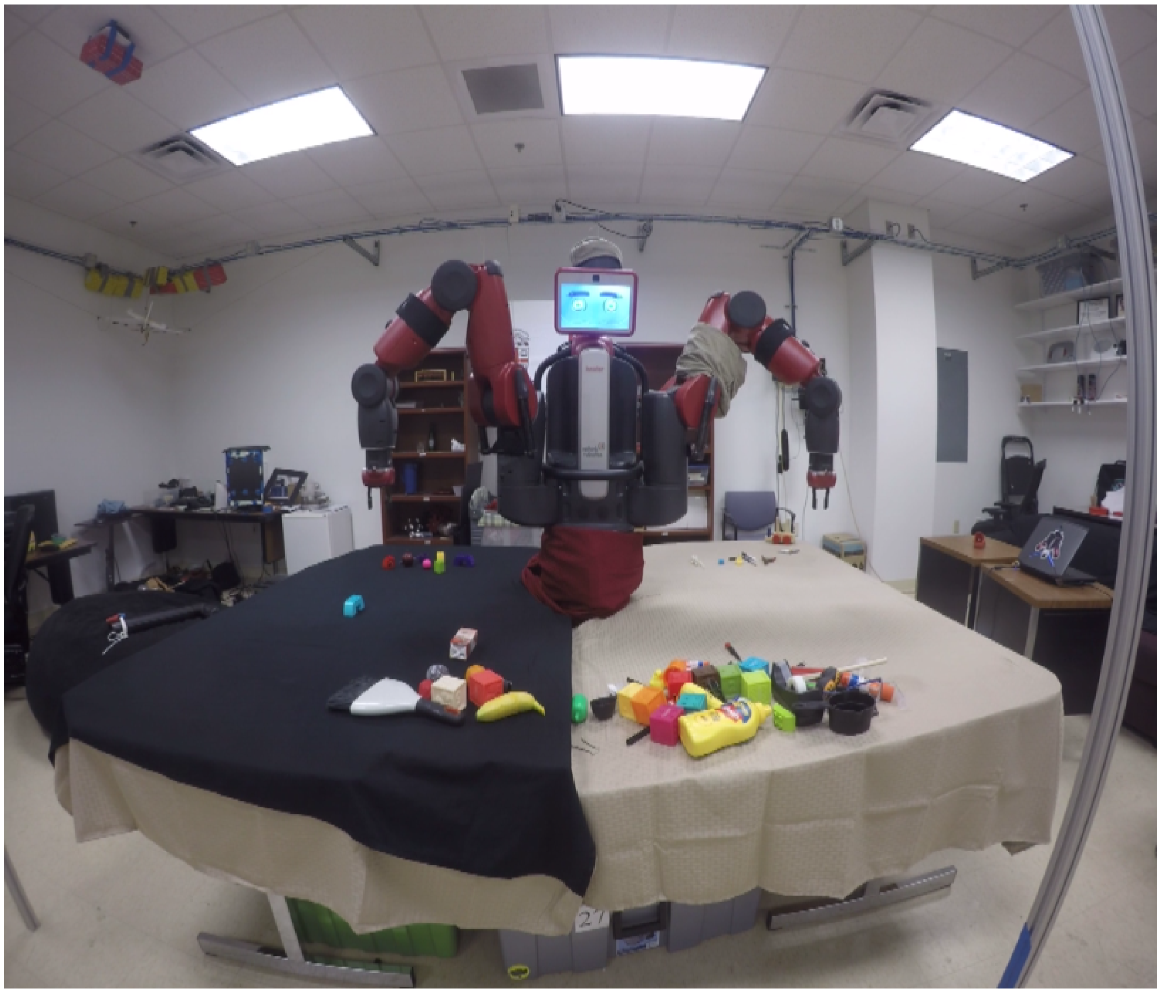


Figure 6.5: The Baxter robot executing Infinite Scan.

6.3.3 Flower Picking

Movement is an action. Detection is an action. Using Back, we can implement higher order detectors and actions from simple elementary models and movements. The movements are part of the detector. The following is a basic recipe that shows the power of simple, rational detectors.

First, form a narrow aperture orthographic projection of the scene. This is a coarse but scale accurate (perspective invariant) map of the scene in which the general location of the flower can be determined with high reliability, but in which distinct petals would be difficult to localize due to resolution. Second, use the IR range finder to determine the dominant plane of the flower. Third, move the arm so that the focal plane of the camera is parallel to the plane of the flower. Image the flower and detect the pose of a petal in the plane using vision. Use the knowledge of range to provide the missing depth coordinate, resulting in a full pose for the petal. Fourth, pick the petal.

The third step here can be further refined. Substep one is to detect the core of the daisy. Substep two is to detect all petals, sequentially. Substep three is to discard petals unattached to the core. Substep four is to sort the petal detections according to θ coordinate (with bounds between, say, 0 and 2π) and select the petal with lowest θ value to pick. If petal picks are sequentially successful and we think of petals ideally as slices of pie, this ensures that we select the petal with the most forgiveness in its pick execution, because we choose a petal whose neighbor has already been picked.

This is just one example of how to solve a problem in Ein with a moving camera which would be harder to solve with a stationary camera.

6.4 Other Robots

Ein is basically a virtual machine that can run inside of many different robot architectures. In this way it somewhat resembles URBI but differs somehow in spirit. My limited experience with URBI suggests that URBI is a protocol or a medium. Ein is the eye in the robot: not just the execution platform for the Back language, but the chosen minimum data structures which facilitate general purpose closed loop hand-eye coordination. Ein is only the controls we need to pilot a vehicle by tele-op, plus the memory and logic to make the vehicle pilot itself. As such, it can run just as well on an arm or a leg.

In general one defines a *config file* for each robot and specifies a number of config files to load at runtime. Each config file has its own stack and interacts on its own topics, so each robot can be controlled separately. By using words that publish to the appropriate channels, different robots can push instructions onto other robots' stacks. In this way, they can synchronize across collaborative tasks with and co-program eachother and human collaborators. Some staple actions for multi-robot programming in Ein include pausing until either sensory or variable conditions are met, and setting variable conditions for other robots in order to indicate task completion.

An AR Parrot quadcopter has been controlled with Ein. The onboard odometry and sensors in the Introduction to Robotics course quadcopter was designed to support Ein and to be accurate enough to enable limited synthetic photography. Each Aibo has a config file. Like arms, it is possible to run more than one

Aibo at a time, each with its own config. The Aibo can perform synthetic photography using parallax developed from the neck while the base is planted, and could perform global odometry by stitching such snapshots together. Robots on holonomic bases are a natural platform to control with Ein, and provide some good examples of robots with multiple config files. Although the Turtle Bot would have a single config file, the PR2 would have a config file for the base, a config file for the head, and a config file for each of the two arms. A base can perform odometry for its pose estimation, instead of the encoders in the arm, yielding decent image synthesis across huge baselines, the size of a room. The experience of deploying Ein on arms, quadrupeds, and drones so far suggests that it is a promising strategy for controlling robots.

Chapter 7

Conclusion

We hope you are convinced that light field photography performed with an eye in hand camera can enable a robot to perceive and manipulate most rigid household objects, even those with pathological non-Lambertian surfaces. We supported this position with three contributions. In Part I of the thesis we showed the theory and application of synthetic photography which we use on our robots, allowing them to collect light fields and to synthesize 2D photographs from those light fields. In Part II we showed the deployment of our perceptual system to enable robots to perform robust pick and place in challenging environments, achieving state-of-the-art results. In Part III we described the reactive programming environment Ein, which coordinates the hand and eye movements of the robot to control its behavior through alternate and simultaneous perception and movement.

We started Part I with an introduction to light fields. Next we talked about collecting light fields with our robot Baxter, as well as calibrating the camera and basic synthetic photography. We moved on to our graphical model for change detection and object pose estimation, which can also be used for tasks such as specular reflection suppression. Finally, we showed some results on extensions of the graphical model where ray origin depth and camera poses are latent variables. Having showed the reliability of the basic methods on common and challenging household objects, we hope you are convinced thatthe advanced, latent methods would bring even more reliability in even more challenging settings, at the cost only of some (readily attainable) computation.

We began Part II with a discussion on autonomous object manipulation, and our project, the Million Object Challenge, which is featured in a museum exhibit. Next we showed results in some challenging domestic problems, where our work enabled a robot to pick silverware from under running water, manipulate transparent household glassware, screw a nut onto a bolt, and robustly manipulate a toy car in a small, unfenced workspace. Having completed some challenging tasks with Baxter, we hope you are convinced that the faster, more precise robotic arms available even today would be able to manipulate a wide variety of (perhaps even most) rigid household objects, regardless of surface properties such as reflectivity and translucence.

We started Part III with a discussion on some of the common problems that robot programmers face, as well as the specific manifestations of those problems in our lab. Next, we introduced our reactive programming environment Ein and its corresponding programming language Back. Back is stack based and programs are first-class data, meaning that robots can program other robots, and that programs are easily interruptable, resumable, and combinable. Next we talked about some applications where Ein facilitated the project completion. Finally, we talked about the present and future applications of Ein to different types of robots. We hope that this part convinces you that programming a robot to accomplish useful tasks in the home can be relatively straightforward in the right framework.

7.1 Future Work

It is worth repeating that the work presented so far targets static, rigid household objects. The time-slice nature of the light field collection process means that moving objects do not develop sharp images. The graphical model will need to be amended to address mobile and deformable targets. Furthermore, we do not take full advantage of the moving arm in that we should be able to detect occlusions and incomplete objects and move the camera to collect missing data. In this section we address mitigation factors for some of these limitations.

7.1.1 Inverse Synthetic Aperture Photography

Inverse synthetic aperture photography (ISAP) is synthetic photography where the parallax is derived from the relative motion of the target and the observer (camera) as opposed to the absolute motion of the observer. An ISAP image requires a target object and the ability to at least coarsely localize the target in constituent frames. Once the relative pose of the object is known, computing an ISAP image is exactly the same as computing a SAP image, provided we substitute the relative pose of the object (known in ISAP) for the camera pose (known in SAP). The ISAP image is centered around the target object. The focal plane can then be varied within the frame of the object.

There is a RADAR equivalent to ISAP called ISAR. In ISAR, usually Doppler effects or other cues give the motion information unambiguously. In the absence of a velocimeter, the relative motion of the target is not well known but can be estimated up to a constant factor in the following way. A stationary observer can make fair use of background subtraction to segment and track the target in the original camera images, from which a center of mass can be derived and used to provide the image pose during aperture synthesis. Translation in the camera plane comes from the coordinates in the original image, and distance from the camera can be estimated up to a constant by comparing the aspect ratios of all the images to the aspect ratio of the first frame (this is bound to be ill posed and so there must be considerations for that). This still leaves the matter of rotation, and a basic solution is to assume that the object does not rotate as it moves past the observer. For certain moving objects this is valid much of the time, such as cars, planes, and people. Otherwise, the rotation component of the pose can be allowed to vary during camera pose re-estimation. Although the non-rotation assumption will probably be a decent initialization when poses smoothly vary, a feature based method could be used to initialize the poses and account for erratic motion. Nonetheless, vanilla ISAP will perform quite well on cars passing a single stationary camera on a road with no turns.

It may be that there is more than one moving object in a scene. We can develop an ISAP image for each moving object, called its *ISAP frame*. Objects which are sharp in eachothers frames move with similar velocity. With that in mind, here is a potentially beneficial ISAP frame initialization algorithm. The first step is to find a small patch with which to initialize a frame. The second step is to develop an ISAP image over this patch, including any iterative fixup procedures and depth (z) inference. Only consider enough synthetic pixels to cover the patch plus a small amount of padding. Third, expand the area of the ISAP frame to cover a much larger area of synthetic pixels. As usual, the variance channel here provides a measure of sharpness. More pixels than the original patch may be in focus. Any synthetic pixels which can be made sharp using the poses derived from the small patch are considered to be *rigid* with respect to the patch. This algorithm also happens to segment the synthetic and constituent photographs according to the motion of the objects in the scene.

You can execute ISAP with a monocular camera, but an array of cameras or lenses with simultaneous capture can improve the situation and resolve some inherent ambiguity. The binocular case is the most interesting. Suppose now we have two cameras in a typical stereo configuration, side-by-side with parallel focal planes and a large area of overlap, but a decent disparity nonetheless. When we perform ISAP with one camera, we are forced to make an assumption about scale, and perform our calculations relative to that chosen scale. With simultaneous capture from two co-calibrated cameras, however, this ambiguity is resolved either implicitly, as only the true scale will cause focus to be consistent across the two cameras' sets of images, or explicitly, by extending the bounding box heuristic and finding approximate scale from the stereo correspondence in each frame. The latter would be useful when periodicity in surface patterns of the target lead to ambiguity from focal cues alone. Similarly, binocular ISAP can help us remove the constant-orientation assumption, but that matter is a little more subtle. The idea here is that if an adversarial target were rotating so to keep a constant profile as it moved relative to the camera, it could make it seem to a naive monocular ISAP algorithm that its shape was, for instance, a full sphere, while in fact hiding arbitrary structure on the backend. A second camera makes such an attack much harder.

7.1.2 SCARLET

There is a well known problem called SLAM, for "Simultaneous Localization and Mapping". A robot is moving about the world and wants to use information from its sensors to navigate, but it does not know what its surrounding look like according to its sensors. The robot can build a map by taking sensor measurements as it moves about the space. If the movements are perfectly known, then the map will be good and can be used for future navigation. But the movements are known only up to a certain level of accuracy, that is, most odometry is noisy. If the map is coherent, it can be used to correct global drift in the underlying odometry system. This map-corrected odometry will presumably build a better map than the uncorrected odometry. And that better map will build a better odometry even still. So a sound approach seems to be trying to Simultaneously Localize [yourself] and Map [the space]. And there are several canonical solutions to SLAM that make some guarantees beyond probably.

In our version of light field SLAM, the map is the synthetic photograph and the localization refers to finding the camera poses. By fixing up the camera poses, we begin to actually localize ourselves (the camera) according to the scene geometry. And by estimating the terminus of each ray, we are mapping the scene to a greater degree than a basic synthetic photograph or focus sweep. Thus by combining Ray Labeling and

Camera Pose Re-Estimation, we have a rudimentary form of Light Field SLAM.

Consider now the technique of ISAP. Assign to each detection its own ISAP frame, tracked through time, and allow ray label assignment through these frames. The resulting graphical model allows for camera poses, ray assignments, and object / terrain (entity) poses to be jointly estimated through likelihood maximization. That is, *Simultaneous Camera Adjustment, Ray Labeling, and Entity Tracking (SCARLET)*.

7.1.3 Sufficiency Estimation and Next-Best View

It takes time to collect images and more computations to process them, and so it is advantageous to minimize the number of images necessary for a given task. Any dense collection pattern has its limits, and so a stopping criterion is desirable even in that case, in order to have knowledge regarding whether enough data has been collected to complete a given task. By synthesizing the required photographs online as the source images are still being collected and analyzing these intermediate synthetic photos, we can stop collecting source images when we have enough to make a good decision.

Many tasks require answering questions about any of three basic scenarios: regarding a work space, regarding an object, and regarding a grasp. Each of these scenarios has various questions associated with it. Have we seen the entire workspace? Is this workspace clean or has something changed? Is this the object we are looking for? Is this object broken? Can we pick the matter in this space? Can we leave something in this space? Grasp inference can be rephrased as object inference, and many questions about objects and workspaces are basically the same. So we will look at some canonical questions about workspaces and objects and see how they can be extended.

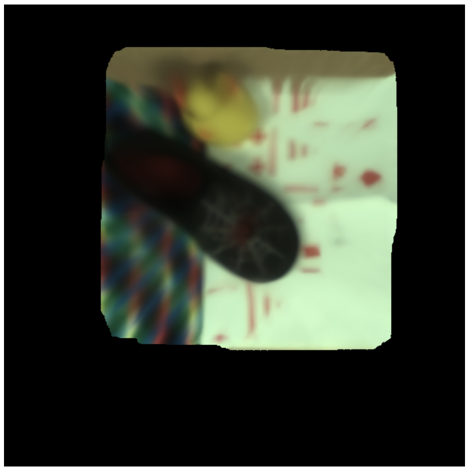
We can answer some of these questions by integrating quantities over synthetic photographs as they are being composed and stopping when those quantities cross thresholds.

For any given task, there is a set of pixels, the *support*, which contain the data necessary to complete the task. Sometimes the support, or a good approximation, is known ahead of time, but other times the identity of the support is latent. Consider surveying a workspace to determine how many objects are present. The support is the whole workspace, since there could be any number of objects, and it is known ahead of time. Now, consider the task of localizing a specific machine part in a workspace. Then the support is the synthetic pixels which contain the object, which is not known ahead of time, since once we know where the object is we need know nothing about the other pixels to localize the object. If we knew the support and object pose ahead of time we could choose the optimal set of images to cover it. Since we can't, we can develop intermediate synthetic photographs as we collect constituent images and check the intermediate photographs for partial detections. We can then collect the images which would complete the best guess for the object in the synthetic photograph, recompute, and compute a new best estimate. If the new best estimate is complete, we can stop. Similarly, tasks such as grasp detection, which can rely on filters, can define a support.

The notion of an estimate being complete, though, has some ambiguity. The probabilistic model includes a notion of certainty in the variance estimate of the color channels, and also in the relative probabilities of the considered heights in a depth map. When such rigor is less important, it may suffice simply to ensure that each cell in the support has been observed a certain number of times or from a certain distribution of angles. But how do we know whether a cell has been observed? From a single camera image it is impossible to tell if the underlying geometry is flat and parallel to the image plane, or whether it contains non-trivial shadow casting geometry which cause parts of the scene to be unobservable from a single angle. Therefore one is tempted to say that in order to determine whether a cell has been observed, we must either determine or make assumptions about the underlying geometry. This is basically true if we use a wide synthetic aperture, and the most complete choice is to use SCARLET for total reconstruction. Secondary choices are to search through focal planes or to know *a priori* which plane contains the target. With a narrow aperture, though, a near orthographic view is produced and scale plays less of a part. Thus a target may be sought with no regard to depth or distance from the camera. Furthermore by comparing wide aperture photographs of different depths to narrow aperture photos, we can effectively search through focal planes to obtain depth estimates without resulting to full scale ray labeling. Matching an orthographic image to a wide aperture image at different depths is a sort of bulk ray labeling procedure that understands that the orthographic rays have special significance. They are truer, in a sense, because they are already in focus and cannot lie, but they are scarcer. If we can use the orthographic rays to inform the surplus of wide angle but ambiguous rays, we may be able to gain some accuracy at the cost of covering less area. A soft version of this technique, that gives more priority to rays which are closer to orthographic, could be folded into SCARLET as a prior. You can imagine learning weights which describe this trust, according to the statistics of which rays end up being true most of the time. These priors would be domain specific but would also serve as descriptors for the bulk geometry of the scene.

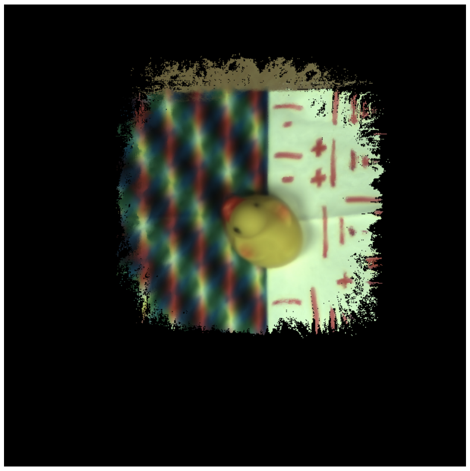
Collection strategies can be organized according to the ability of the robot to change the scale of the images relative to the target it seeks. A fixed wing aircraft, a typical pedestrian car, and a robot arm held high above a workspace can all change the scale of their images significantly by approaching their targets. A drone constrained to fly low over a large area, a robot dog searching for a bone, and a robot base performing odometry cannot change their scale as dramatically and so must rely on a different set of heuristics. Scalable search can employ coarse to fine strategies, where non-scalable search is more sequential. Either way, though, it is clear that some problems call the Travelling Salesman to mind and thus that NP-completeness is likely lurking. This means that exact solutions to such problems probably don't exist. Fortunately, for many tasks, the target will be small relative to the area observable with a single view, so that even if the target needs to be viewed from multiple angles, the problem space will not get large enough to explode, and heuristic approaches and approximations will do quite well. A thorough treatment of this problem is beyond the scope of this document. A good start to such a study would be to implement a greedy algorithm that chose the next view at each step in order to collect the most rays intersecting the geometry that is still unknown at that step.

Consider Figure 7.1b, an image from Figure 4.1 in which the RGB and Z values have been jointly estimated using ray labeling. The isolated black cells represent as of yet unobserved locations where none of the labeled rays terminate. The corresponding marginal estimate without ray labeling, seen in Figure 7.1a, is made from the same constituent images and attributes data to the entire target region. Most of the unobserved cells around the edge are on flat terrain and so were actually observed, but the algorithm falsely attributes those rays to neighboring cells due to i.i.d. noise in the camera and the fact that considering individual rays by themselves is ill posed. These artifacts might be eliminated by considering groups of contiguous rays together, much as super-pixels in traditional 2D segmentation problems. In the upper left corner, though, we have an example of the algorithm behaving as expected. The yellow rubber duck depicted in this corner is several centimeters high and occludes the cells further to the top left. Because the duck is on the edge of the imaged area, it is observed from one side only and the locations on the opposite side remain under-observed and are considered unobserved. That is, the algorithm correctly considers the patches behind the duck as unobserved due to occlusion. If the duck were placed in the middle of the target area, as in Figure 7.1c, the algorithm would see the points behind the duck from more perspectives and they would be considered observed. A next-best view algorithm would consider the geometry already known in the scene, determine a perspective from which the unobserved cells could be viewed unoccluded, collect the missing data, re-infer the photograph and geometry, and repeat.



(a) Marginal estimate with no ray labeling.

(b) Marginal estimate after ray labeling, showing missing data due to artifacts on monotone regions and occlusion behind the duck.



(c) Marginal estimate after ray labeling of a scene with the duck in the center, showing all sides of the duck unoccluded.

Figure 7.1: Images repeated from Figure 4.1 and Figure 4.2, demonstrating a situation where a next-best view system could fill in missing data.

Bibliography

[1] Adelson, E. H., Bergen, J. R., et al. The plenoptic function and the elements of early vision.

[2] Argall, B. D., Chernova, S., Veloso, M., and Browning, B. A survey of robot learning from demonstration. *Robotics and Autonomous Systems 57*, 5 (2009), 469–483.

[3] Association, E. I., et al. Eia standard eia-274-d interchangeable variable block data format for positioning, contouring, and contouring/positioning numerically controlled machines, 1979.

[4] Baillie, J.-C. Urbi: Towards a universal robotic body interface. In *Humanoid Robots, 2004 4th IEEE/RAS International Conference on* (2004), vol. 1, IEEE, pp. 33–51.

[5] Banta, J. E., Wong, L., Dumont, C., and Abidi, M. A. A next-best-view system for autonomous 3-D object reconstruction. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on 30*, 5 (2000), 589–598.

[6] Bennett, P. N. Assessing the calibration of naive bayes posterior estimates. Tech. rep., DTIC Document, 2000.

[7] Benton, S. A. Survey of holographic stereograms. In *Processing and Display of Three-Dimensional Data* (1983), vol. 367, International Society for Optics and Photonics, pp. 15–20.

[8] Bohren, J., Rusu, R. B., Jones, E. G., Marder-Eppstein, E., Pantofaru, C., Wise, M., Mösenlechner, L., Meeussen, W., and Holzer, S. Towards autonomous robotic butlers: Lessons learned with the PR2. In *2011 IEEE International Conference on Robotics and Automation* (2011), IEEE, pp. 5568–5575.

[9] Bolles, R. C., Baker, H. H., and Marimont, D. H. Epipolar-plane image analysis: An approach to determining structure from motion. *International journal of computer vision 1*, 1 (1987), 7–55.

[10] Calli, B., Singh, A., Walsman, A., Srinivasa, S., Abbeel, P., and Dollar, A. M. The YCB object and model set: Towards common benchmarks for manipulation research. In *2015 International Conference on Advanced Robotics* (2015), IEEE, pp. 510–517.

[11] Collet, A., Xiong, B., Gurau, C., Hebert, M., and Srinivasa, S. S. Herbdisc: Towards lifelong robotic object discovery. *The International Journal of Robotics Research* (2014).

[12] Correll, N., Bekris, K. E., Berenson, D., Brock, O., Causo, A., Hauser, K., Okada, K., Rodriguez, A., Romano, J. M., and Wurman, P. R. Lessons from the Amazon picking challenge. *arXiv preprint arXiv:1601.05484* (2016).

[13] Engel, J., Schöps, T., and Cremers, D. LSD-SLAM: Large-scale direct monocular SLAM. In *European Conference on Computer Vision*. Springer, 2014, pp. 834–849.

[14] Eppner, C., Höfer, S., Jonschkowski, R., Martín-Martín, R., Sieverling, A., Wall, V., and Brock, O. Lessons from the Amazon picking challenge: Four aspects of building robotic systems. In *Robotics: Science and Systems XII* (AnnArbor, Michigan, June 2016).

[15] Furukawa, Y., Hernández, C., et al. Multi-view stereo: A tutorial. *Foundations and Trends® in Computer Graphics and Vision 9*, 1-2 (2015), 1–148.

[16] Gallagher, G., Srinivasa, S. S., Bagnell, J. A., and Ferguson, D. Gatmo: A generalized approach to tracking movable objects. In *2009 IEEE International Conference on Robotics and Automation* (2009), IEEE, pp. 2043–2048.

[17] Georgiev, T., Yu, Z., Lumsdaine, A., and Goma, S. Lytro camera technology: theory, algorithms, performance analysis. In *Multimedia Content and Mobile Devices* (2013), vol. 8667, International Society for Optics and Photonics, p. 86671J.

[18] Gershun, A. The light field. *Studies in Applied Mathematics 18*, 1-4 (1939), 51–151.

[19] Girshick, R., Donahue, J., Darrell, T., and Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition* (2014), IEEE, pp. 580–587.

[20] Goldfeder, C., Ciocarlie, M., Dang, H., and Allen, P. K. The Columbia grasp database. In *2009 IEEE International Conference on Robotics and Automation* (2009), IEEE, pp. 1710–1716.

[21] Gortler, S. J., Grzeszczuk, R., Szeliski, R., and Cohen, M. F. The lumigraph. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* (1996), ACM, pp. 43–54.

[22] Herbst, E., Henry, P., Ren, X., and Fox, D. Toward object discovery and modeling via 3-D scene comparison. In *2011 IEEE International Conference on Robotics and Automation* (2011), IEEE, pp. 2623–2629.

[23] Huang, A. S., Bachrach, A., Henry, P., Krainin, M., Maturana, D., Fox, D., and Roy, N. Visual odometry and mapping for autonomous flight using an RGB-D camera. In *International Symposium on Robotics Research* (2011), pp. 1–16.

[24] Huang, A. S., Tellex, S., Bachrach, A., Kollar, T., Roy, D., and Roy, N. Natural language command of an autonomous micro-air vehicle. In *2010 Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems* (Taipei, Taiwan, Oct. 2010).

[25] Kasper, A., Xue, Z., and Dillmann, R. The KIT object models database: An object model database for object recognition, localization and manipulation in service robotics. *The International Journal of Robotics Research 31*, 8 (2012), 927–934.

[26] Kasper, M., Correll, N., and Yeh, T.-H. Abstracting perception and manipulation in end-user robot programming using sikuli. In *2014 IEEE International Conference on Technologies for Practical Robot Applications* (2014), IEEE, pp. 1–6.

[27] Katz, D., and Brock, O. Manipulating articulated objects with interactive perception. In *2008 IEEE International Conference on Robotics and Automation* (2008), IEEE, pp. 272–277.

[28] Kent, D., Behrooz, M., and Chernova, S. Crowdsourcing the construction of a 3d object recognition database for robotic grasping. In *2014 IEEE International Conference on Robotics and Automation* (2014), IEEE, pp. 4526–4531.

[29] Kent, D., and Chernova, S. Construction of an object manipulation database from grasp demonstrations. In *2014 Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems* (2014), IEEE, pp. 3347–3352.

[30] Kraft, D., Detry, R., Pugeault, N., Baseski, E., Guerin, F., Piater, J. H., and Kruger, N. Development of object and grasping knowledge by robot exploration. *Autonomous Mental Development, IEEE Transactions on 2*, 4 (2010), 368–383.

[31] Krainin, M., Henry, P., Ren, X., and Fox, D. Manipulator and object tracking for in-hand 3d object modeling. *The International Journal of Robotics Research 30*, 11 (2011), 1311–1327.

[32] Kress-Gazit, H., Fainekos, G. E., and Pappas, G. J. Temporal-logic-based reactive mission and motion planning. *Robotics, IEEE Transactions on 25*, 6 (2009), 1370–1381.

[33] Lai, K., Bo, L., Ren, X., and Fox, D. A scalable tree-based approach for joint object and pose recognition. In *Twenty-Fifth Conference on Artificial Intelligence* (August 2011).

[34] Leith, E. N., and Upatnieks, J. Reconstructed wavefronts and communication theory. *JOSA 52*, 10 (1962), 1123–1130.

[35] Levoy, M., and Hanrahan, P. Light field rendering. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* (1996), ACM, pp. 31–42.

[36] Lowe, D. G. Distinctive image features from scale-invariant keypoints. *International journal of computer vision 60*, 2 (2004), 91–110.

[37] Lyubova, N., Filliat, D., and Ivaldi, S. Improving object learning through manipulation and robot self-identification. In *2013 IEEE International Conference on Robotics and Biomimetics* (2013), IEEE, pp. 1365–1370.

[38] MATUSZEK, C., FOX, D., AND KOSCHER, K. Following directions using statistical machine translation. In *Proceedings of the 5th ACM/IEEE international conference on Human-robot interaction* (New York, NY, USA, 2010), HRI '10, ACM, pp. 251–258.

[39] MCGUIRE, M., MATUSIK, W., PFISTER, H., HUGHES, J. F., AND DURAND, F. Defocus video matting. *ACM Trans. Graph. 24*, 3 (2005), 567–576.

[40] MODAYIL, J., AND KUIPERS, B. The initial development of object knowledge by a learning robot. *Robotics and autonomous systems 56*, 11 (2008), 879–890.

[41] NEWCOMBE, R. A., IZADI, S., HILLIGES, O., MOLYNEAUX, D., KIM, D., DAVISON, A. J., KOHI, P., SHOTTON, J., HODGES, S., AND FITZGIBBON, A. Kinectfusion: Real-time dense surface mapping and tracking. In *2011 10th IEEE international symposium on Mixed and augmented reality* (2011), IEEE, pp. 127–136.

[42] NG, R. *Digital light field photography*. PhD thesis, stanford university, 2006.

[43] OBERLIN, J., AND TELLEX, S. Autonomously acquiring instance-based object models from experience. In *International Symposium on Robotics Research* (2015).

[44] PEARL, J. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, 2014.

[45] PHILLIPS, C. J., LECCE, M., AND DANIILIDIS, K. Seeing glassware: from edge detection to pose estimation and shape recovery. In *Robotics: Science and Systems XII* (2016).

[46] POSTSCRIPT. Postscript — Wikipedia, the free encyclopedia, 2015. [Online; accessed 15-October-2015].

[47] (PROGRAMMING LANGUAGE), F. Forth (programming language) — Wikipedia, the free encyclopedia, 2015. [Online; accessed 15-October-2015].

[48] (PROGRAMMING LANGUAGE), R. Rpl (programming language) — Wikipedia, the free encyclopedia, 2015. [Online; accessed 15-October-2015].

[49] QUIGLEY, M., CONLEY, K., GERKEY, B., FAUST, J., FOOTE, T., LEIBS, J., WHEELER, R., AND NG, A. Y. Ros: an open-source robot operating system. In *ICRA workshop on open source software* (2009), vol. 3, p. 5.

[50] RODRIGUES, J. J., KIM, J.-S., FURUKAWA, M., XAVIER, J., AGUIAR, P., AND KANADE, T. 6D pose estimation of textureless shiny objects using random ferns for bin-picking. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems* (2012), IEEE, pp. 3334–3341.

[51] ROSTEN, E., AND DRUMMOND, T. Machine learning for high-speed corner detection. In *European Conference on Computer Vision*. Springer, 2006, pp. 430–443.

[52] ROTH, S., AND BLACK, M. J. Specular flow and the recovery of surface structure. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on* (2006), vol. 2, IEEE, pp. 1869–1876.

[53] SALAS-MORENO, R. F., NEWCOMBE, R. A., STRASDAT, H., KELLY, P. H., AND DAVISON, A. J. SLAM++: Simultaneous localisation and mapping at the level of objects. In *2013 IEEE Conference on Computer Vision and Pattern Recognition* (2013), IEEE, pp. 1352–1359.

[54] SCHIEBENER, D., MORIMOTO, J., ASFOUR, T., AND UDE, A. Integrating visual perception and manipulation for autonomous learning of object representations. *Adaptive Behavior 21*, 5 (2013), 328–345.

[55] SELVATICI, A. H., AND COSTA, A. H. Object-based visual slam: How object identity informs geometry.

[56] SHADEMAN, A., DECKER, R. S., OPFERMANN, J., LEONARD, S., KIM, P. C., AND KRIEGER, A. Plenoptic cameras in surgical robotics: Calibration, registration, and evaluation. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on* (2016), IEEE, pp. 708–714.

[57] SMITH, B. M., ZHANG, L., JIN, H., AND AGARWALA, A. Light field video stabilization. In *2009 IEEE 12th international conference on computer vision* (2009), IEEE, pp. 341–348.

[58] SZELISKI, R. *Computer vision: algorithms and applications*. Springer, 2010.

[59] TAO, M. W., SRINIVASAN, P. P., HADAP, S., RUSINKIEWICZ, S., MALIK, J., AND RAMAMOORTHI, R. Shape estimation from shading, defocus, and correspondence using light-field angular coherence.

[60] TELLEX, S., KOLLAR, T., DICKERSON, S., WALTER, M., BANERJEE, A., TELLER, S., AND ROY, N. Understanding natural language commands for robotic navigation and mobile manipulation. In *Proc. AAAI* (2011).

[61] UDE, A., SCHIEBENER, D., SUGIMOTO, N., AND MORIMOTO, J. Integrating surface-based hypotheses and manipulation for autonomous segmentation and learning of object representations. In *2012 IEEE International Conference on Robotics and Automation* (2012), IEEE, pp. 1709–1715.

[62] WANG, C.-C., THORPE, C., THRUN, S., HEBERT, M., AND DURRANT-WHYTE, H. Simultaneous localization, mapping and moving object tracking. *The International Journal of Robotics Research 26*, 9 (2007), 889–916.

[63] WANG, T.-C., ZHU, J.-Y., HIROAKI, E., CHANDRAKER, M., EFROS, A. A., AND RAMAMOORTHI, R. A 4D light-field dataset and CNN architectures for material recognition. *arXiv preprint arXiv:1608.06985* (2016).

[64] WANNER, S., MEISTER, S., AND GOLDLUECKE, B. Datasets and benchmarks for densely sampled 4d light fields. In *International Symposium on Vision, Modeling and Visualization* (2013), Citeseer, pp. 225–226.

[65] Wong, L. L., Kaelbling, L. P., and Lozano-Pérez, T. Data association for semantic world modeling from partial views. *International Journal of Robotics Research 34*, 7 (2015), 1064–1082.

[66] Yang, J. C., Everett, M., Buehler, C., and McMillan, L. A real-time distributed light field camera. *Rendering Techniques 2002* (2002), 77–86.

[67] Zobel, M., Fritz, M., and Scholz, I. Object tracking and pose estimation using light-field object models. In *International Symposium on Vision, Modeling and Visualization* (2002), pp. 371–378.