Depth Estimation from Video with a Single RGB Camera of Known Position

Arthur Erlendsson  
Undergraduate, Social Robotics Lab  
Yale University  
New Haven, CT  
arthur.erlendsson@yale.edu

Brian Scassellati  
Social Robotics Lab  
Yale University  
New Haven, CT  
brian.scassellati@yale.edu

Abstract— Extracting depth information from a robot's environment is an often mission-critical task that demands the use of specialized equipment such as infrared sensors or dedicated stereoscopic camera rigs. Many depth estimation algorithms that bypass the use of such equipment still require extensive supervised learning on training sets of images that have been painstakingly annotated by hand. Thus, we consider the task of plug-and-play sparse depth map generation using any RGB camera and its position. We construct a structure-from-motion computer vision pipeline that matches features across frames using the scale-invariant feature transform (SIFT) algorithm and estimates objects' distances based on features' pixel disparities between frames and the known camera trajectory. The pipeline is tested on a set of images captured indoors by a medium resolution 640px x 400px camera on the right end effector of a force-compliant Baxter system. The pipeline estimates depth well on camera trajectories mostly parallel to the camera plane but functions poorly when rotating around objects in front of the camera. Due to the structure-from-motion nature of the algorithm, it is tolerant of minor amounts of image rotation, sensor noise, and motion blur. The pipeline is highly generalizable to any platform, capable of being deployed on any system that makes camera pose information available. Runtime averaged between .06 to .08 seconds per call, depending on the feature-richness of the scene, demonstrating that real-time application of the pipeline is possible. The algorithm's main drawbacks are the inaccuracy in depth measurement and the difficulty of tuning feature matching, as poor tuning results in extreme levels of noise.

Keywords—structure from motion, depth estimation, computer vision, feature matching

I. INTRODUCTION

Recovering 3D depth information from images is one of the core problems in the field of computer vision and is a vitally important task for in robotics. Often, developers resort to using expensive infrared sensors such as the Microsoft Kinect, which projects a structured light field and infers depth from distortion of a known pattern; other light ranging systems can be extremely expensive. Dedicated stereo rigs are also available, but can add extra cost or weight to robotic systems. Thus, using an RGB camera as a source of depth information reduces the complexity of the robot platform. However, producing accurate for 3D reconstructions from RGB images is a complicated task. Many estimation implementations require establishing a ground truth (e.g. photos of the background, photos of the floor) or require training datasets, and thus fail in the case of small irregularities. A naïve depth estimation algorithm is thus more flexible and may be preferable despite increased noise.

The original goal of the project was to use both position and orientation data to rectify pairs of images and use a block-matching algorithm to output a dense depth estimation across the entirety of the overlap between both frames. However, after some initial investigation, we discovered that this fell outside the scope of a semester-long project, as it effectively required a re-write of OpenCV’s calibration and rectification functions.

Thus, we attempt to create a computer vision pipeline that can distinguish levels of depth in pairs of frames, if not provide an approximate measure of depth. While the implementation is not novel, it is a demonstration that robotic platforms do not require dedicated systems to recover depth information about their surroundings.

II. DEPTH FROM STEREO IMAGES

The principle concept behind computing depth from each pair of images is the same problem that of stereo correspondence. Given images captured in different location, one can match sets of points in each image and measure the pixel disparity between the locations of the two sets.

The first approach to the correspondence problem is the correlation-based block matching algorithm, where one checks if a location in one
image matches a neighboring location in another. Correlation-based methods are highly susceptible to even small image rotations, as they can only check for large disparities in the axis of camera translation algorithms (as compared in both the both axes parallel to the image plane) without becoming too computationally expensive. Feature-based methods are another option for identifying disparities, taking an approach similar to those used in optical flow. Strong features are identified in each image and are only then matched to each other across frames. Feature-based methods are generally more tolerant to image rotation. We elect to use feature-based methods in this project due to high frequency of rotations between images.

After computing disparities between features, we can approximate depth using the focal length of the camera and the distance between the cameras (the baseline) as illustrated by the following diagram.

![Figure 1: An illustration of the correspondence problem.](image)

In Figure 1, X represents a point in 3D space. The variables x and x' represent the point on the 2D image plane where X is projected. The value f is the focal length, and O and O' represent camera positions. We can compute depth with the following equation:

$$\text{disparity} = x - x' = \frac{Bf}{Z}$$

Z is the depth of the 3D point projected onto each image. Computing depth on each feature in a pair of images yields a sparse depth map.

III. IMAGE PROCESSING

Each time the function is called, the following six values must be passed in –

- **img1** – by convention, the left image is passed in as the first image, although the algorithm does not differentiate between a left-right or right-left ordering
- **img2** – the second image
- **pos1** – a tuple of (x, y, z) coordinates for the location of the camera when the first image was captured, assumed to be in meters
- **pos2** – a second tuple of (x, y, z) coordinates locating the second image capture
- **K** – the intrinsic camera matrix
- **lowesratio** – this parameter determines the selectivity of ratio test described in Lowe’s paper on SIFT

The intrinsic camera matrix and Lowe’s ratio parameters will be discussed below in greater detail.

A. Rejecting disparity-free images

If two images are taken in too close proximity to one another, the difference in disparity between foreground objects and background can may be too small to observe. Thus, we run into the issue of having zero disparity values on certain features, and attempting to compute depth requires division by zero.

We do not compute a depth map on any pair of images that are less than 0.4 centimeters away from each other to avoid problem of zero disparity. This value can be tuned, as translations under this threshold can see large disparities if performing depth estimation on objects very close to the camera. When using the pipeline in a real-time application, this would indicate to the caller that the previous depth map does not require updating.

B. Rejecting too blurry images

Performing feature detection on a pair of blurry images results in poor feature matching. We reject a pair of images when either image is too blurry.

The Laplacian method set forth by Pech-Pacheco et al. convolves a 3x3 Laplacian kernel against the grayscale image and takes the variance of the response. Because the Laplacian operator measures the second derivative of the image, edges are highlighted due to fast changes in intensity. If the variance of the Laplacian is large, there must be many edges and many non-edges in the photograph, suggesting that the image contains good features; if the variance is small, then the image is likely blurred and does not contain good features [1].

Setting a high threshold for unblurred images improves feature detection, but can result in too few images being acceptable for the pipeline.
C. Feature detection and matching

Identifying strong features in each image is the core of the pipeline. By default, the function uses scale-invariant feature transform to identify features in the training image (in this case, the left image), generating a description vector for each feature. The feature matcher then computes the Euclidean distance between feature description vectors in the left image and right image, and selects pairs of neighbors by closest distance.

After selecting matches, we proceed to filter feature matches on four conditions:

1. No fixed object should be used as a feature (e.g. Baxter grippers in the test dataset).
2. No pair of features should be matched if the distance between the features is greater than a previously set threshold. This prevents noisy mismatches with unrealistic depth estimations.
3. Features must pass Lowe’s ratio test.

Lowe’s ratio test states that the ratio of the distances of the closest and next closest matches must be lower than a certain threshold. By lowering the threshold, matches that pass the test are likely to be correct. Lowe’s paper suggests that a ratio of .25 nearly guarantees that all mismatches are filtered, depending on the problem domain [2]. Tuning the ratio test is one of the most important parts of the pipeline.

D. Filtering by epiline

Using the matched features in the left and right images, we compute the fundamental matrix, a 3x3 matrix that describes the correspondence between the two images.

The fundamental matrix $F$ is defined such that for all corresponding points $x$ and $x'$, the following equation holds:

$$ x'^T F x = 0. $$

$F$ contains the translations and rotations needed to describe the coordinates of the second image in the system of the first image.

Because translations and rotations are fixed between all points, a line can be drawn from the origin of the first image to a certain point in 3D space; this line can also be projected onto the image plane of the second camera, resulting in another corresponding line called the epiline. To find the same point in the second image, one can only search along the epiline. This epipolar constraint allows us to discard matches that fail this epipolar constraint because they do not sit on a correct epiline, further increasing the accuracy of the match filter.

E. Depth estimation

After cleaning our matched feature set, computing depth is simply a matter of evaluating the disparity equation presented in Section II and solving for $Z$. The distance between features is the pixel disparity, and focal length (in pixels) and the baseline are known constants for the pair of frames.

IV. RESULTS

The following results were computed on images taken on a 640px x 400px camera on the right end effector of a force-compliant Baxter system. All photos are taken indoors, with three different views.

Depth values are colored into three quantiles: white is closest; teal is the middle range; dark blue is farthest. This coloring carries no significance, only done to illustrate the pipeline’s ability to discriminate between depth levels in the frames.

The first view was a top-down perspective of building blocks on a table. The leftmost stack and bottommost stack are one block high, and the middle and rightmost stacks are two and three blocks high respectively. The camera moved 1.14 cm, mostly horizontally; the ratio test value chosen was $\text{lowesratio} = 0.18$ after tuning. Figure 2 is the frame captured in the right position.

![Figure 2: Top-down depth estimation.](image-url)
moved between the first and second sets of captures. The camera moved 1.83 cm, mostly horizontally; the ratio test value chosen was $\text{lowesratio} = 0.15$ after tuning. Figure 3 is the frame captured in the right position.

Figure 3: Sideways depth estimation, highly selective in matching.

Again, the color quantiles make it clear that levels in the image are distinguishable: each level of blocks is separable; the back of the table and the shelves are close in value; and the back wall increases in distance as it slants away from the camera. However, values are still underestimated – the depth values on the back wall absurdly suggest that the entire scene is contained within only two meters of space in front of the camera.

Repeating the same pair of images with a more lenient ratio test shows a higher yield of depth values of lower quality. Figure 4 was processed with $\text{lowesratio} = 0.3$, which, according to Lowe, would still yield a mismatch rate of far under 5%.

Figure 4: Sideways depth estimation, lower selectivity in matching.

Figure 4 demonstrates that lowering our match selectivity still extracts a significant amount of depth information in the scene. Some noisy points do stand out, such as the depth value supplied at the top of the three-tall stack of green blocks.

Intuitively, when a camera rotates significantly, the scene backdrop is displaced far more than the foreground. Thus, the assumptions of translational correspondence break down.

The function was called on every third image in the entire top-down image set, averaging a runtime of 0.067 seconds. The function was also called on every third image in the sideways image set, averaging a runtime of 0.076 seconds, likely requiring more time per call due to increased scene complexity.

V. DISCUSSION

The structure-from-motion pipeline is clearly capable of providing a general understanding of the topography of a scene when the camera is performing a translation parallel to the image plane. While depth values themselves may not be accurate, it is possible to infer depth and recognize objects that stand out from a scene’s backdrop.

As the difference between Figure 3 and Figure 4 indicates, tuning can have an immense impact on the outputted depth map, affecting point yield and quality. While Figure 3 is an extremely sanitized, mismatch-free view of the scene, Figure 4 provides a holistic understanding of range from the camera. Selectivity must be tuned by the experimenter on an application-by-application basis. One can also modify the match-filter approach, sorting all matches by distance from the next best match and only selecting the strongest matches.
The pipeline runs in nearly real time on a camera with a refresh rate of thirty frames per second, requiring approximately for .07 seconds per call. Computation time can be decreased by switching from a brute-force feature matcher to FLANN, with a noticeable drop in accuracy. Because the algorithm runs on a stream of images, depth maps can be denoised by assuming temporal consistency, referencing maps from previous frames and removing extreme changes in depth values.

While the algorithm is not accurate enough to totally replace stereoscopic camera rigs or infrared depth sensors, it can be used for applications that only require low-fidelity depth information. The top-down image set demonstrates that a single camera can be used to determine and differentiate object sizes without user input. For example, in a collaborative manufacturing setting, a robot could use a preprogrammed camera path to determine the number of blocks left of each color without asking an experiment participant or tracking each block’s individual removal.

VI. CONCLUSION

3D reconstruction at low cost is a heavily researched task with most solutions appearing in the form of dedicated sensors. Due to the inherent complexity in processing images, it is likely that the for best solutions do not lie in the domain of monocular RGB vision. However, the ubiquity of RGB pinhole cameras makes primitive algorithms for depth estimation valuable to the robotics community.

I would like to specially thank Brian Scassellati for both advising me on this senior project and sharing his vast knowledge of every robotics-related field over the last four years during my time at Yale. I would also like to thank Alessandro Roncone and Olivier Mangin for their invaluable guidance over the duration of this project.

REFERENCES
