Homework Assignment 6

15-463/663/862, Computational Photography, Fall 2018
Carnegie Mellon University

Due Friday, Nov. 30, at 11:59pm

The purpose of this assignment is to use structured light as a means for reconstructing the 3D shape of a surface. Rather than having to use a projector, you will rely on shadows to create a structured illumination, a technique often referred to as "weakly structured light".

In particular, you will implement the "desktop scanner" of Bouguet and Perona [2]. As shown in Figure 1, this system is composed of five primary items: a camera, a point-like light source (e.g., desk lamp), a stick, two planar surfaces, and a calibration checkerboard. By waving the stick in front of the light source, you can cast line shadows into the scene. As Bouguet and Perona demonstrate, the depth at each pixel can then be recovered using simple geometric reasoning.

In the course of completing this homework, you will need to develop a good understanding of camera calibration, Euclidean coordinate transformations, and manipulation of lines and planes. Rather than just "encouraged" to read it, to solve this assignment you will need to carefully go through Bouguet and Perona [2]. So you should read it carefully before starting the assignment, and keep a copy handy while working through solving the assignment.

1 Implementing structured-light triangulation (150 points)

For the first part of the homework, you will be using two image sequences contained in the `./data` directory of the homework ZIP archive. One is a `calib` sequence you will use to estimate the intrinsic and extrinsic calibration parameters of the camera, and consists of ten images of a checkerboard at various poses. The second is a sequence captured for the `frog` object shown in Figure 1. For each sequence we have provided both a high-resolution 1024768 sequence, as well as a low-resolution 512384 sequence for development. You should convert these color images to grayscale, e.g., using `rgb2gray`.

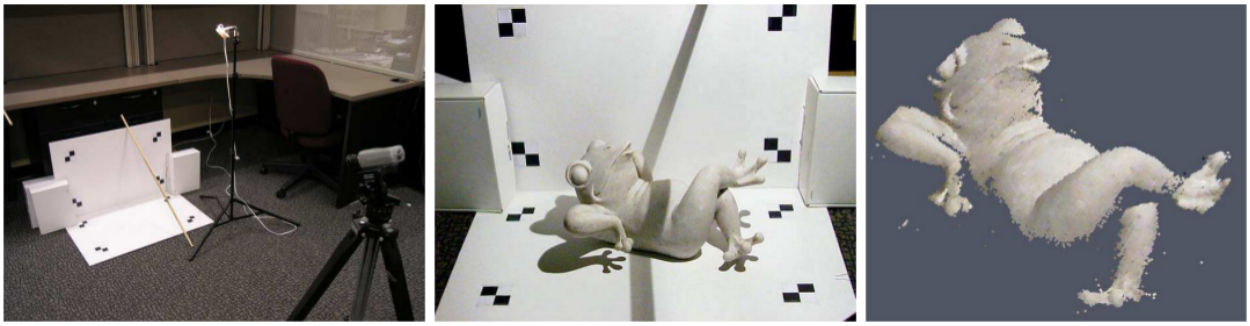


Figure 1: 3D Photography using Planar Shadows. From left to right: the capture setup, a single image from the scanning sequence, and a reconstructed object (rendered as a colored point cloud).

1.1 Video processing (30 points)

Your first task is to estimate two fundamental quantities from an input video sequence: (1) the time that a shadow enters a pixel and (2) the position of the shadow edge as a function of time. The following sections outline the basic procedures for performing these tasks. You will need to consult Section 2.4 in [2] for additional information.

Spatial shadow edge localization. In terms of Figure 2 in [4], you need to estimate the shadow lines $\lambda_h(t)$

and $\lambda_v(t)$, projected on the horizontal and vertical planar regions, respectively. In order to perform this and subsequent processing, you will utilize a spatio-temporal approach. We begin by defining the maximum and minimum intensity observed in each pixel (x, y),

$$I_{\max}(x, y) = \max_t I(x, y, t), \tag{1}$$

$$I_{\min}(x, y) = \min_t I(x, y, t). \tag{2}$$

In order to detect the shadow boundaries, we choose a per-pixel detection threshold which is the midpoint of the dynamic range observed in each pixel. As a result, the shadow edge can be localized by the *zero crossings* of the difference image

$$\Delta I(x, y, t) = I(x, y, t) - I_{\text{shadow}}(x, y), \tag{3}$$

where the shadow threshold image is defined to be

$$I_{\text{shadow}}(x, y) = \frac{I_{\max}(x, y) + I_{\min}(x, y)}{2}. \tag{4}$$

In practice, you will need to select an occlusion-free image patch for each planar region. Afterwards, you can obtain a set of sub-pixel shadow edge samples (for each row of the patch) by interpolating the position of the zero-crossings of $\Delta I(x, y, t)$. To produce a final estimate of the shadow edges $\lambda_h(t)$ and $\lambda_v(t)$, you should find the best-fit line (in the least-squares sense) to the set of shadow edge samples. The desired output of this step is illustrated in Figure 2(a), where the best-fit lines are overlaid on the original image.

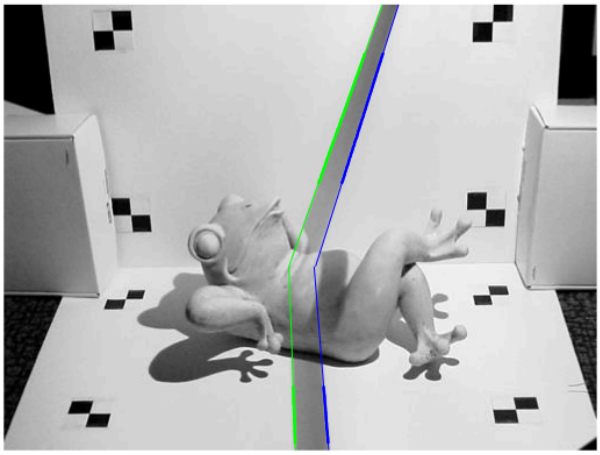
Temporal shadow edge localization. After calibrating the camera (see next section), the previous step will provide all the information necessary to recover the position and orientation of each shadow plane as a function of time. As described in Section 1.3, in order to reconstruct the object, you also need to know when each pixel entered the shadowed region. This task can be accomplished in a similar manner as spatial localization. Instead of estimating zero-crossing along each row for a fixed frame, you can assign the per-pixel shadow time using the zero crossings of the difference image $\Delta I(x, y, t)$ for each pixel (x, y) as a function of time t. The desired output of this step is illustrated in Figure 2(b), where the shadow crossing times are quantized to 32 values (with blue indicating earlier times and red indicated later ones). Note that you may want to include some additional heuristics to reduce false detections. For instance, dark regions cannot be reliably assigned a shadow time. As a result, you can eliminate pixels with very low contrast $I_{\max}(x, y) - I_{\min}(x, y)$.

In your submission, show a few examples of spatial and temporal edge localizations, analogous to those shown in Figure 2.

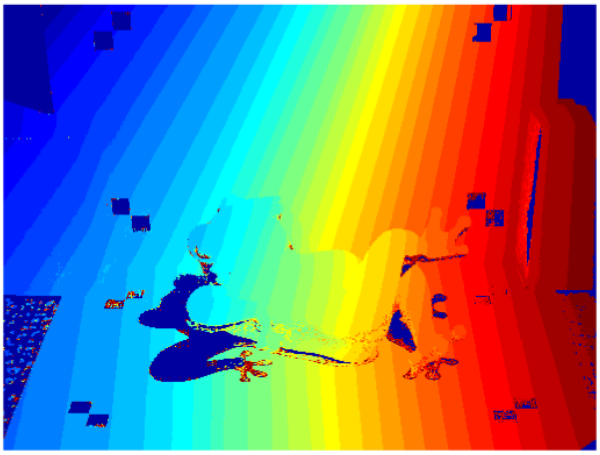
1.2 Calibration (50 points)

You will need the intrinsic and extrinsic calibration of the camera in order to transfer image measurements into the world coordinate system. You will be using the *Camera Calibration Toolbox* for Matlab, also created by Jean-Yves Bouguet. This toolbox is widely used within the computer vision community. The intrinsic and extrinsic parameters are estimated by viewing several images of a checkerboard at various poses. Before continuing, you should download the toolbox and review the documentation on the toolbox website [1]. In particular, review the first calibration example and the description of calibration parameters. After downloading the toolbox, make sure it is in your Matlab path.

Intrinsic calibration. The intrinsic parameters of the camera can be obtained using the `calib` command of the calibration toolbox: Change the current working directory to one of the calibration sequences, then Type `calib` at the Matlab prompt to start. Since you are only using a few images, select "Standard (all the images are stored in memory)" when prompted. To load the images, select "Image names" and press return, then j. Then select "Extract grid corners", pass through the prompts without entering any options, and then follow the on-screen directions. (Note that we used a calibration target with the default 30mm30mm squares. Also, always skip any prompts that appear.) Once you have finished selecting corners, choose "Calibration", which will run one pass though the calibration algorithm. Next, choose "Analyze error". Left-click on any



(a) spatial shadow edge localization



(b) temporal shadow edge localization

Figure 2: Spatial and temporal shadow edge localization. (a) The shadow edges are determined by fitting a line to the set of zero crossings, along each row in the planar regions, of the difference image $\Delta I(x, y, t)$. (b) The shadow times (quantized to 32 values here) are determined by finding the zero-crossings of the difference image $\Delta I(x, y, t)$ for each pixel (x, y) as a function of time t.

outliers you observe, then right-click to continue. Repeat the corner selection and calibration steps for any outliers. Once you have an evenly-distributed set of reprojection errors, select "Recomp. corners" and finally "Calibration". To save your intrinsic calibration, select "Save".

Include the resulting `.mat` file with the intrinsic calibration parameters in your submission.

Extrinsic calibration. From the previous step you now have an estimate of how pixels can be converted into normalized coordinates (and subsequently rays in world coordinates, originating at the camera center). In order to assist you with your implementation, we have provided a Matlab script called `extrinsicDemo`. This demo will allow you to select four corners on the "horizontal" plane to determine the Euclidean transformation from this ground plane to the camera reference frame. (Always start by selecting the corner in the bottom-left and proceed in a counter-clockwise order. For your reference, the corners define a 558.8mm x 303.2125mm rectangle.) In addition, observe that the final section of `extrinsicDemo` uses the included function `pixel2ray` to determine the optical rays (in camera coordinates), given a set of user-selected pixels.

1.3 Reconstruction (70 points)

At this point you have estimated all the parameters required to recover the depth of each pixel in the image (or at least those where the shadow could be observed). In terms of Figure 2 in [2], you can use the camera calibration to obtain a parametrization of the ray defined by a true object point P and the camera center O_c. Given the shadow time for the associated pixel $\bar{x}_c = (x, y)$, you can lookup (and potentially interpolate) the position of the shadow plane at this time. The resulting ray-plane intersection will provide an estimate of the 3D position of the surface point. Repeating this procedure for every pixel will produce a 3D reconstruction. For more details on the reconstruction process, please consult Sections 2.5 and 2.6 in [2].

Now that you have recovered a 3D point cloud, you will need to visualize the result. You can use Matlab's `pointCloud` command to convert the reconstructed 3D points into a point-cloud structure. You can then display this structure using the command `pcshow`. To give you some expectation of reconstruction quality, Figure 3 shows the results we obtained with our reference implementation. Note that there are several choices you can make in your implementation; some of these may allow you to obtain additional points on the surface or increase the reconstruction accuracy. Please document the methods you used to optimize your reconstruction.

Figure 3: Reconstruction results for the frog sequence.

2 Building your own 3D scanner (100 points)

You will now build your own version of the weakly-structured light 3D scanner. You can replicate the setup of Figure 1, using a desk lamp, and the camera and tripod you borrowed from the class.

You will additionally need to print a checkerboard for performing your own calibration. We recommend using the same checkerboard configuration (in terms of number of boxes and their dimensions) as in the data sequence provided with the homework.

Finally, in setting up the scanner, you will need to create the configuration of the two planes. You should use appropriate holders (e.g., thick books) to ensure that the vertical plane is as close to orthogonal to the floor as possible. You should also mark the corners of a rectangle of known dimensions on each plane, to simplify calibration.

Use your 3D scanning setup to scan at least one object, and include images of the scanned images and the final reconstruction. Additionally, include a photograph of the setup you built.

Deliverables

As described on the course website, solutions are submitted through Canvas. Your solution should be an archive (e.g., a ZIP file) that includes the following:

- A PDF report explaining what you did for each problem, including the various visualizations of albedoes, normals, and surfaces, as well as renderings of images, that are requested throughout problems 1 and 2, as well as answers to all questions asked throughout both problems. The report should include any figures and intermediate results that you think may help. Make sure to include explanations of any issues that you may have run into that prevented you from fully solving the assignment, as this will help us determine partial credit.
- All of your Matlab code, as well as a `README` file explaining how to use the code.

Please organize your solution submission using the following file structure:

```
.zip
├─ .pdf ..... The PDF report.
├─ src/ ..... Contains all Matlab M-files and the README file explaining how to use the code.
└─ data/ ..... Contains all image, video, and other data files.
```

Hints and Information

- When building your own version of the 3D scanner, you should note some practical issues associated with this approach.

First, it is important that every pixel be shadowed at some point in the sequence. As a result, you must wave the stick slow enough to ensure that this condition holds.

In addition, the reconstruction method requires reliable estimates of the plane defined by the light source and the edge of the stick. Ambient illumination must be reduced so that a single planar shadow is cast by each edge of the stick, otherwise your shadow estimates will be off.

The light source you use must be sufficiently bright to allow the camera to operate with reasonable exposures and minimal gain, otherwise sensor noise will corrupt the final reconstruction. It is best to use a small lamp, such as a desk lamp or similar. This ensures that the light source is sufficiently point-like to produce abrupt shadow boundaries. Otherwise, the estimate of the shadow plane will not be reliable.

When calibrating your own camera, it is important to ensure planarity of the checkerboard pattern. We recommend that you stick the pattern on a flat surface (e.g., a wooden panel). Additionally, it is important that you capture a sufficient number of images, spanning a large variety of checkerboard poses everywhere in the field of view of your camera. The calibration sequence we provide in the homework should give you a sense of what sort of images you need.

In a departure from previous homeworks, here it is not necessary to use RAW images. Given that you will be capturing video sequences, it is probably easier to work with PNG images.

Finally, you should set the focal length, focus, and aperture settings of your lens appropriately, so that all of the scanning setup is within your field of view and sharply in focus. Blurry regions will result in poor shadow estimates, and therefore inaccurate reconstruction.

Credits

This homework was directly adapted from the 3D photography class offered by Gabriel Taubin at Brown. This includes the write-up, figures, and data.

References

[1] J.-Y. Bouguet. Camera calibration toolbox for Matlab, 2010. http://www.vision.caltech.edu/bouguetj/calib_doc/.

[2] J.-Y. Bouguet and P. Perona. 3d photography using shadows in dual-space geometry. *International Journal of Computer Vision*, 35(2):129–149, 1999.