Due: 10th March 2017 11:55 PM Total Points: **25**

1 Depth from Binocular Stereo

One of the features our brains use to discern scene depth is stereo disparity, i.e. how the images seen by both our eyes differ. Relative to the field-of-view of each of our eyes, objects which are close to us will appear to be in drastically different locations, whereas objects which are far will appear almost identically positioned. This effect occurs whenever two or more cameras are used to image a scene, so long as they are separated by some finite distance from one another. Under the right circumstances this object disparity can be used as a surprisingly robust estimator for scene depth[1].

Your task is therefore: given a stereo image pair, create a third image which contains at every pixel the depth of the scene. The scene depth is directly related to object disparity by geometry, but to make that conversion, you must know the separation between the cameras. Since this separation is unknown, you can present the disparity at every pixel as a proxy for scene depth.

Two images are given, `tsukuba_l.ppm` and `tsukuba_r.ppm`, taken by left and right cameras respectively. The images are already rectified, meaning for each camera each row of pixels lies on exactly one epipolar plane (and the planes are the same for both cameras). That is, if some object feature is imaged to a point on k^{th} row of the left image, the same object feature will be imaged to a point on the k^{th} row of the right image (assuming it is visible in both images). This is convenient, as literally counting the pixel separation between the relative locations of such a point yields its disparity (and therefore its depth). But, for any scene other than point light sources surrounded by darkness, actually finding these corresponding points is non-trivial[2].

One way to find which pixels correspond to which in the images' rows is to consider a small patch centered on each pixel in question. For a wide variety of scenes, it is reasonable to assume that this patch would be "similar" in both the images, despite possibly appearing in different locations. The extent of this similarity yields a quantifiable metric to find the corresponding scene points. Various measures of similarity can be used, such as the pixel-wise sum of squared distances or the patches' normalized cross-correlation.

[1] In fact, this is how many "3D" cinematography cameras work.

[2] And when something is difficult, researchers give it a name, in this case, "The Correspondence Problem."

1.1 Solving Correspondence

10 points

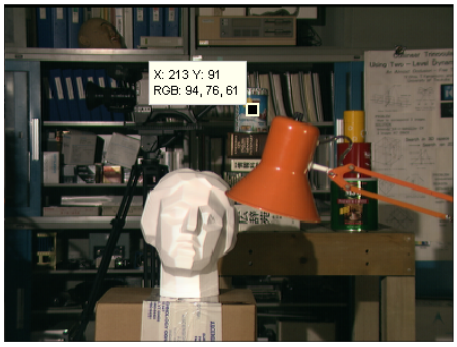
1. Choose a patch size and a similarity metric, and provide a rationale for why you chose what you did.

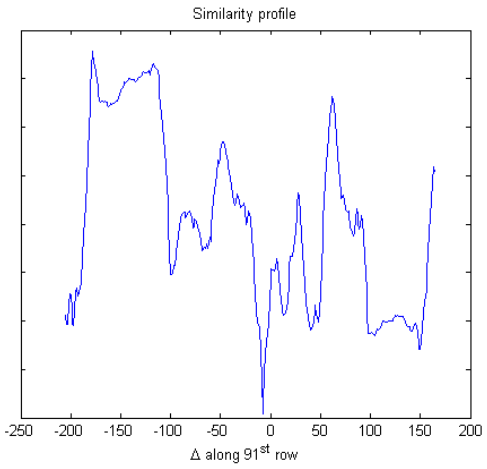
2. Consider each of the following pixels $\mathbf{p}_{i=1,2,3,4,5,6}(k,m)$ at $(row = k,\ col = m)$ of the left image and their associated patches:

   (a) $\mathbf{p}_1(136, 83)$
   (b) $\mathbf{p}_2(203, 304)$
   (c) $\mathbf{p}_3(182, 119)$
   (d) $\mathbf{p}_4(186, 160)$
   (e) $\mathbf{p}_5(123, 224)$
   (f) $\mathbf{p}_6(153, 338)$

   For each pixel $\mathbf{p}_i(k,m)$ of left image, calculate and plot the output (profile) of your similarity metric with respect to diparity Δ to all the pixels along the k^{th} row of the right image. Interpret and give reasons for the profiles observed. Does your plot clearly have a reliable extremum? If so, why? If not, why not?

   An example profile is shown below (plotted for pixel $\mathbf{p}(91, 213)$ of left image using arbitrary similarity metric):





   (Where, in the plot, $\Delta = m_2 - m_1$ for similarity measure between pixel $\mathbf{p}(k, m_1)$ of left image and pixel $\mathbf{q}(k, m_2)$ of right image)

3. From your observations of the similarity profiles you just calculated, devise a strategy to estimate the most similar patch. Justify your choices. Include a test to reject patches which have no good matches, and denote such problem-pixels as undefined.

4. Repeat this calculation for all left image pixels. Find the point of maximal similarity in each case using your estimation strategy. The position of this estimate relative to the position of the left image pixel under consideration is the estimated disparity value, Δ_{est}. Accumulate all these disparities (one for each left image pixel) into a matrix and display the matrix as a heatmap. This is the disparity map. Be sure to mark the undefined locations clearly in some way.

5. To fill in the undefined values, use an interpolation technique. There are a variety of possible options, including nearest-neighbor, linear, and cubic[3]. Choose one, describe how it works in your own words, and use it to estimate the unknown disparities. Provide a rationale for why you chose what you did. Display the resulting disparity map.

6. The technique described above attempts to find each pixel correspondence independently. It uses a hard constraint of epipolar geometry and a soft constraint of patch similarity in order to find the correspondence. But, in addition, we can also have other soft constraints like smoothness, uniqueness and ordering of correspondences. These constraints help to relate between pixels on the same row (epipolar line). Briefly explain the three constraints (smoothness, uniqueness and ordering) and mention the cases when they would fail.

1.2 Scanline stereo

10 points

1. Read the included paper[4], "Stereo by intra- and inter-scanline search using dynamic programming," until section 3.2. What are the key points of this portion of the paper? With their approach in mind, how could smoothness, uniqueness and ordering constraints be incorporated to improve the quality of correspondences? Note: although this paper discusses edge correspondences, many of the concepts can be applied to patch correspondences also.

2. Implement dynamic programming to solve the patch correspondence problem. They refer to this as the "inter-scanline search" in the paper.

[3]This is such a common operation that Matlab has many built-in functions which can do this for you. See `griddata, triscatterinterp.`

[4]Ohta, Yuichi, and Takeo Kanade. "Stereo by intra-and inter-scanline search using dynamic programming." Pattern Analysis and Machine Intelligence, IEEE Transactions on 2 (1985): 139-154.

2 *Grad Credits:* Active 3D Cameras

5 points

A stereo camera system falls under the category of passive 3D capture system. That is, the system doesn't include an actively illuminating light source. The light captured by the system comes from only the world. Another example of a passive 3D capture system is a light field camera. On the other end, there is a class of active 3D capture systems. These systems include an illumination source that is designed along with cameras to estimate depth. There are two such technologies, viz, structured light (SL) and Time-of-Flight (ToF). Commercial examples of these systems are Kinect for Xbox 360 (uses SL) and Kinect for Xbox One (uses ToF). You don't see the light emanating from these systems in plain sight because they are in the infrared (IR) wavelength, the wavelength invisible to our eyes.

For this section of the assignment, you are going to compare and contrast the depth technologies of the two Kinect sensors by reading the attached paper[5], "Kinect range sensing: Structured-light versus Time-of-Flight Kinect". Briefly explain the two technologies: SL and ToF. What are the advantages and disadvatanges of each of them? What are their advantages and disadvantages against passive 3D systems? What are the source of errors in each system?

References

- Stereo evaluation challenge and dataset reference: `http://vision.middlebury.edu/stereo/`
  `http://vision.middlebury.edu/stereo/data/scenes2001/`
- `http://en.wikipedia.org/wiki/Binocular_disparity`
- Chapter on Stereo correspondence from "Computer Vision: Algorithms and Applications" by Richard Szeliski, `http://cronos.rutgers.edu/~meer/TEACHTOO/PAPERS/Szeliskistereodraft.pdf`

[5]Hamed Sarbolandi, Damien Lefloch, Andreas Kolb, "Kinect range sensing: Structured-light versus Time-of-Flight Kinect", Computer Vision and Image Understanding, Volume 139, October 2015, Pages 1-20

Submission Instructions

Every student must submit following 2 files:

- An organized report submitted as a PDF document. The report should describe the implementation, issues (problems encountered, surprises), and an analysis of the test results (interpretation of effects of varying parameters, different image results). Intermediate and final results must be provided.
- A ZIP file containing the necessary codes.

The heading of the PDF file should contain the assignment number and topic. Also, attach a photo of yourself at top-left of the PDF along with your name and department.

Late Submission Policy

Assignments are expected to be submitted on the due date. Each student gets a total of 3 late days that can be used however you wish. For examples, all 3 days can be used towards 1 assignment or 1 day late for 3 assignments or other combinations. Late submissions beyond that will be penalized as below:

- One day late will be penalized 25% of the credit.
- Two Days late will be penalized 50%.
- Submissions more than 2 days late will not be considered for credit.

I will be ruthless in enforcing this policy. There will be no exceptions

Collaboration Policy

I encourage collaboration both inside and outside class. You may talk to other students for general ideas and concepts but the programming must be done independently. For mid-term and final examination there will be no collaboration permitted.

Plagiarism

Plagiarism of any form will not be tolerated. You are expected to credit all sources explicitly. If you have any doubts regarding what is and is not plagiarism, talk to me.