EECS 4422/5323 Computer Vision

Stereopsis 3

Calden Wloka

11 November, 2019

Announcements

- Assignment 2 due this Friday
- Labs this week will be devoted to Assignment 2
- Course schedule change - last four lectures will be devoted to final project demos

Final Demo Timing

$$4\,\text{lectures} \times \frac{80\,\text{minutes}}{\text{lecture}} = 320\,\text{minutes}$$

- 31 students, round to 32 slots: 10 minutes per slot

Final Demo Timing

$$4\,\text{lectures} \times \frac{80\,\text{minutes}}{\text{lecture}} = 320\,\text{minutes}$$

- 31 students, round to 32 slots: 10 minutes per slot
- 7 minute presentations, 2 minutes for questions, 1 minute buffer for next talk

Final Demo Timing

$$4\,\text{lectures} \times \frac{80\,\text{minutes}}{\text{lecture}} = 320\,\text{minutes}$$

- 31 students, round to 32 slots: 10 minutes per slot
- 7 minute presentations, 2 minutes for questions, 1 minute buffer for next talk
- You must stick to this timeline; I will cut you off after seven minutes, and direct questions to be taken up after class at 9 minutes.

Final Demo Dates

- Wednesday, November 20th
- Monday, November 25th
- Wednesday, November 27th
- Monday, December 2nd

Final Demo Dates

- Wednesday, November 20th
- Monday, November 25th
- Wednesday, November 27th
- Monday, December 2nd

Going early (especially the 20th) means you may present partially completed work. The advantage of going early is that it helps you build what can essentially serve as a partial outline for your final report.

Final Demo Presentation Tools

- HDMI connector
- My laptop (Ubuntu 18.04, PDF or Google slides)
- Classroom computer (Windows, but should be tested first)

If you are using your own computer, make sure you have the appropriate connection dongle ready and you know how to export or mirror your screen.

If you are using my laptop, you must provide your slides to me by 10:30 the morning of the class so I can set up the environment to easily switch between talks in order.

Final Demo Content

Engineering project: “describe to the class the goal and function of the implemented algorithm, as well as provide an example of its execution.”

Scientific project: “describe to the class the goal and motivation for the study, what results were obtained, and what conclusions can be drawn from those results.”

Outline

- Assignment 2 Discussion (Hints and concepts clarification)
- Rectified Images
- Disparity and Depth

Assignment Motivation

Assignment 1 was intended to get you more comfortable with pixel-level manipulation of images, and so the focus was on the implementation of image processing functions. Assignment 2, by contrast, is meant to emphasize the analysis of computer vision models (a miniature walkthrough of analysis leading up to your project completion).

Assignment Motivation

Assignment 1 was intended to get you more comfortable with pixel-level manipulation of images, and so the focus was on the implementation of image processing functions. Assignment 2, by contrast, is meant to emphasize the analysis of computer vision models (a miniature walkthrough of analysis leading up to your project completion).

Note: This assignment is computationally intensive, so it is recommended that you complete Part 1 as soon as possible so you have time to debug and begin generating results for Parts 2 and 3.

Assignment Motivation

Assignment 1 was intended to get you more comfortable with pixel-level manipulation of images, and so the focus was on the implementation of image processing functions. Assignment 2, by contrast, is meant to emphasize the analysis of computer vision models (a miniature walkthrough of analysis leading up to your project completion).

Note: This assignment is computationally intensive, so it is recommended that you complete Part 1 as soon as possible so you have time to debug and begin generating results for Parts 2 and 3.

One more note: You can complete the functions specified in Parts 2 and 3 even if you don't have Part 1 working or the analysis completed. This is worthwhile from a marks perspective.

Assignment Walkthrough

- You are starting midway through a project. You have results from several models, and now need to compare them

Assignment Walkthrough

- You are starting midway through a project. You have results from several models, and now need to compare them
- Part 1 is the implementation of a thumbnail generator. You are given a detailed API generating thumbnail ROIs

Assignment Walkthrough

- You are starting midway through a project. You have results from several models, and now need to compare them
- Part 1 is the implementation of a thumbnail generator. You are given a detailed API generating thumbnail ROIs
- Parts 2 and 3 encompass the analysis of the function developed in Part 1, and are intended to highlight how different ways of defining performance and ground-truth can affect your apparent results and conclusions

Assignment Walkthrough

- You are starting midway through a project. You have results from several models, and now need to compare them
- Part 1 is the implementation of a thumbnail generator. You are given a detailed API generating thumbnail ROIs
- Parts 2 and 3 encompass the analysis of the function developed in Part 1, and are intended to highlight how different ways of defining performance and ground-truth can affect your apparent results and conclusions
- Note: For any function with a detailed API, make sure you follow it!

Assignment Walkthrough

- You are starting midway through a project. You have results from several models, and now need to compare them
- Part 1 is the implementation of a thumbnail generator. You are given a detailed API generating thumbnail ROIs
- Parts 2 and 3 encompass the analysis of the function developed in Part 1, and are intended to highlight how different ways of defining performance and ground-truth can affect your apparent results and conclusions
- Note: For any function with a detailed API, make sure you follow it!
- You will notice that all the functions with APIs in the assignment are for specific instances (one ROI, or the scoring of a specific ROI). To complete the analysis asked of you, you will need additional code to link these functions and iterate over all the images in the dataset and settings specified in the assignment

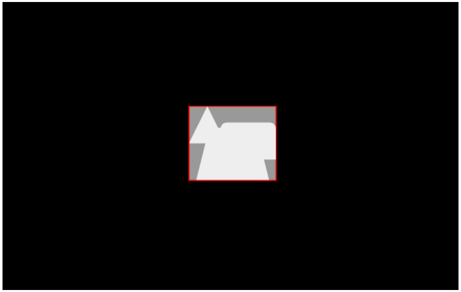
A Note on Precision-Recall Plots

In a standard precision-recall curve, recall is treated as an independent variable, such that you are computing how precision varies with recall. What I have asked for is not the same (largely driven by the constraints of computing the ROIs), and you are effectively computing both precision and recall as dependent variables on a hidden parameter (threshold).

Because of this unorthodox structure, one could argue that the standard area-under-the-curve analysis of the resultant precision-recall curve is potentially misleading, and instead all that should be reported is the average precision and average recall for each model and parameter set (*i.e.*, average across the thresholds). You may choose to report either the area under the curve or the average precision and recall values separately.

Dealing with Ground-Truth Masks

Here we have a hypothetical object mask, with minimal bounding box shown in red. True positives are white pixels included in an ROI, false negatives are white pixels not included in an ROI. False positives are black pixels included in an ROI, true negatives are black pixels not included in an ROI. Grey pixels should be ignored.



Note: If your predicted ROI *only* includes ignored pixels, you get a division by zero. For the purposes of this project, the precision in this case should be interpreted as zero (no true positives were found).

Rectified Stereo Pairs

In addition to recovering the baseline, the goal of most stereo calibration routines is to derive *rectified stereo pairs*.

A standard rectified stereo pair has the following (ideal) properties:

- All epipolar lines run horizontally across the image
- All corresponding points share the same vertical coordinate

Rectification

To accomplish rectification we must find a projective transformation which brings the two images into a common image plane (like with panoramic image matching).

Note that this involves a projection, and there are multiple approaches to this problem. Differences tend to revolve around what aspects of the projection are minimized (*e.g.*number of pixels lost or degree of image warping).

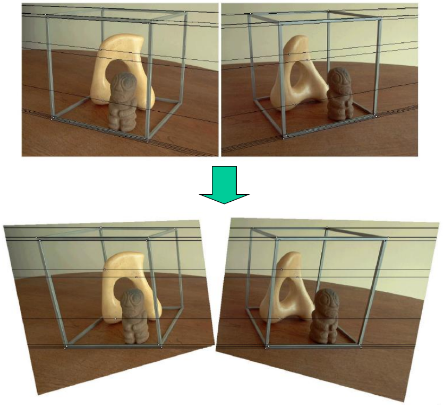


Image source: Loop and Zhang, 1999, via Szeliski, 2011

Disparity

Assuming we have rectified images, how do we compute depth?

The first step is to compute correspondence between the left and right images, which yields the *disparity*, d, of a given match.

$$x_R = x_L + d(x, y), \quad y_R = y_L + d(x, y)$$

but, since we have rectified images, this simplifies to

$$x_R = x_L + d(x), \quad y_R = y_L$$

Disparity to Depth

From a given disparity value, we can compute the depth, z, using the equation

$$z = f\frac{B}{d}$$

where f is the focal length and B is the baseline.

Sparse vs. Dense Disparity Maps

We talked in previous lectures about different correspondence approaches: keypoints (sparse, but generally more reliable matches), and area matching (dense matches, more potential for error).

Sparse vs. Dense Disparity Maps

We talked in previous lectures about different correspondence approaches: keypoints (sparse, but generally more reliable matches), and area matching (dense matches, more potential for error).

If we want to obtain a dense depth map, we typically need different approaches depending on our method of correspondence: for sparse disparity maps, we need to use *region growing*, whereas for dense disparity maps we typically concentrate on *map refinement*.

Challenges

- How do we recover from erroneous matches?

Challenges

- How do we recover from erroneous matches?
- How do we deal with noise in the disparity measurement?

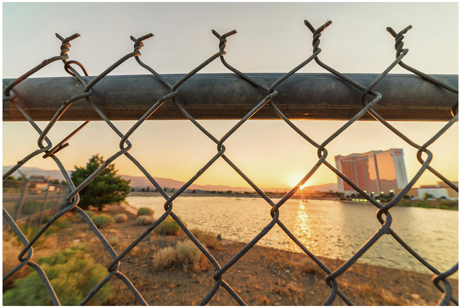
Challenges

- How do we recover from erroneous matches?
- How do we deal with noise in the disparity measurement?
    - We want to compute the depth map which best fits the disparity data

Challenges

- How do we recover from erroneous matches?
- How do we deal with noise in the disparity measurement?
  - We want to compute the depth map which best fits the disparity data
  - How do we detect when a disparity jump is erroneous or an actual part of the scene?

Some Challenging Examples



Growing Seed Points

Start with a distribution of *seed points* across your image. Each point represents the basis for a *region*.

Growing Seed Points

Start with a distribution of *seed points* across your image. Each point represents the basis for a *region*.

Iteratively grow these regions, typically based on some sort of appearance measure over the source image.

Growing Seed Points

Start with a distribution of *seed points* across your image. Each point represents the basis for a *region*.

Iteratively grow these regions, typically based on some sort of appearance measure over the source image.

Once all pixels have been assigned to a region, stop. Some methods may then refine this segmentation by merging similar neighbouring regions.

Converting Regions to a Depth Map

Not all approaches are the same, but some common steps include:

- Treat each region as a whole for the purposes of fitting depth data

Converting Regions to a Depth Map

Not all approaches are the same, but some common steps include:

- Treat each region as a whole for the purposes of fitting depth data
- Determine if region borders should be continuous (both regions belong to the same surface) or discontinuous (the region border is an object boundary)

Converting Regions to a Depth Map

Not all approaches are the same, but some common steps include:

- Treat each region as a whole for the purposes of fitting depth data
- Determine if region borders should be continuous (both regions belong to the same surface) or discontinuous (the region border is an object boundary)
- Assign depth to each region (assume smooth parametric surface or planar element) which best fits the depth information of contained keypoints and border constraints

Region Growing: Superpixels

A common and popular class of region growing models are based on *superpixels*, which falls under the class of *image oversegmentation*.

Superpixels:

- Are computationally efficient to compute
- Tend to group pixels in perceptually meaningful ways
- Usually seek approximate uniformity in distribution and size

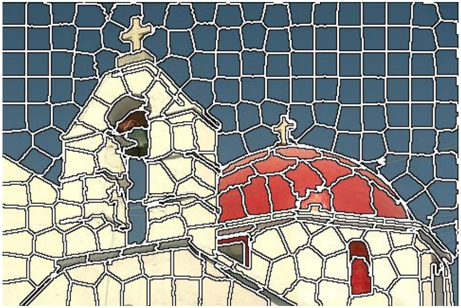
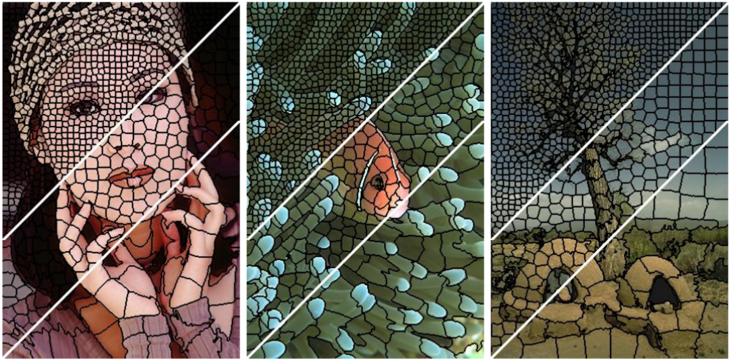


Image Source: Achanta *et al.*, 2012

SLIC Superpixels

An example of a commonly used superixel method is SLIC Superpixel (Achanta *et al.*, 2012).



Example of SLIC using different parameters for desired size.
Image Source: Achanta *et al.*, 2012

A Familiar Example

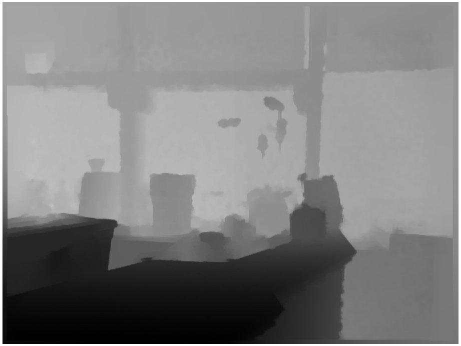
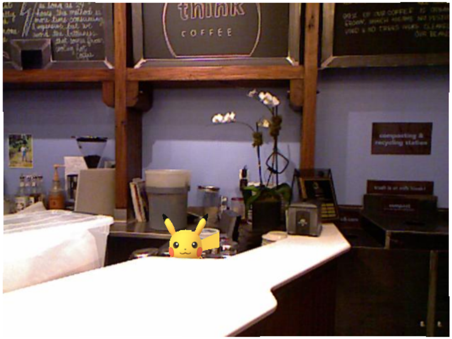
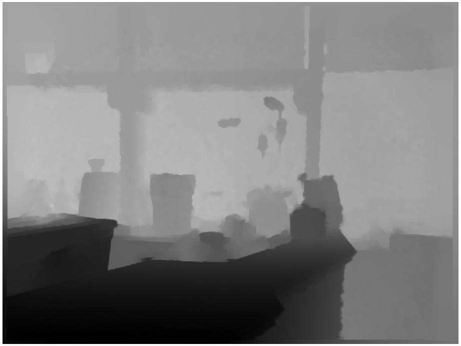


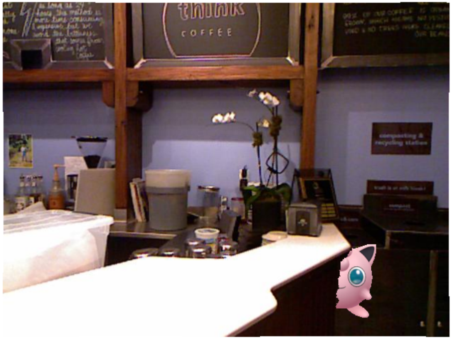
Image Source: NYU Depth Dataset V2, Silberman *et al.*, 2012

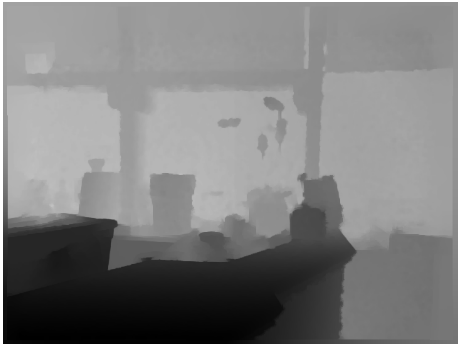
A Familiar Example - Good Insertion



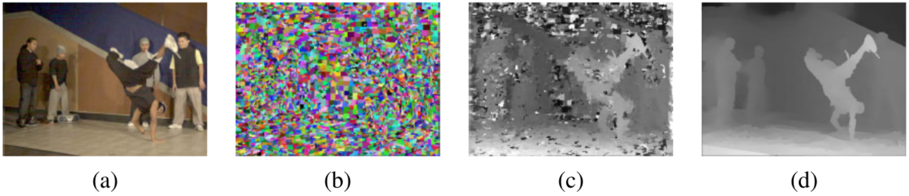


A Familiar Example - Noisy Insertion





Map Refinement



(a.) Input (b.) Colour-based segmentation
(c.) Initial disparity (d.) Refined disparity
Image Source: Zitnick *et al.*, 2004, via Szeliski, 2011