Final Project: Stereo Correspondence

Description

For this topic you will implement two algorithms. First, you will write code for a simple sum squared difference stereo correspondence algorithm as described in the lectures. Second, you will write stereo correspondence code incorporating an advanced energy minimization or graph-cut method of your choice. You are expected to write code that takes in a pair of images and returns a disparity map.

Sample Dataset: http://vision.middlebury.edu/stereo/data/scenes2014/
Related Lectures (not exhaustive): 3A-3D, 4A-4C

Problem Overview

Methods to be used: Implement stereo correspondence methods to find disparity maps between multiple views. These methods must not contain pre compiled functions that perform Stereo Matching. Instead, you are required to apply the knowledge acquired from your research.

RULES:

- **Don't use external libraries for core functionality**
  You are encouraged to use libraries while writing code for your final report. However you will receive a low score if the main functionality of your code is provided via an external library. Don't use stereo functions available in OpenCV or any other library.
- **Don't copy code from the internet**
  The course honor code is still in effect during the final project. All of the code you submit must be your own. You may consult tutorials for libraries you are unfamiliar with, but your final project submission must be your own work. Any instance that does not follow the Honor Code and the class rules will be directly reported to the Office of Student Integrity.
- **Don't use pre-trained machine learning models**
  If you choose a topic that requires the use of machine learning techniques, you are expected to do your own training. Downloading and submitting a pre-trained model is not acceptable for any project topic.
- **Don't rely on a single source**
  We want to see that you performed research on your chosen topic and incorporated ideas from multiple sources in your final results. Your project must not be based on a single research paper and definitely must not be based on a single online tutorial.

Please do not use absolute paths in your submission code. All paths must be relative to the submission directory. Any submissions with absolute paths are in danger of receiving a penalty!

Programming Instructions

You may use the python 3 environment provided for the project. This new environment is simply a list of the versions of libraries that will be used during grading. You may install them however you wish. We recommend

conda. Include a README.md file with usage instructions that are clear for the grader to run your code. Remember to specify what version of python you are using. Notice that despite having Tensorflow and Pytorch in the environment, you are not allowed to use them.

Windows Users Warning:
Be warned that TA's grade exclusively on linux machines. Thus, it is your responsibility to make sure that your code is platform independent. This is particularly important when using paths to files. If your code doesn't run during grading due to some incompatibility you will incur a heavy penalty.

Write-up instructions

The report must be a PDF of 3-6 pages including images and references. Not following this requirement will incur in a significant penalty and the content will be graded up to page 6. **Note that the report will be graded subject to a working code.** There will be no report templates provided with the project materials.

The report must contain:

1) A clear and concise description of the algorithms you implemented. This description must include references to recently published computer vision research and show a deep understanding of your chosen topic.
2) Results from applying your algorithm to images or video. Both positive and negative results must be shown in the report and you must explain why your algorithm works on some images, but not others.

You report must be written to show off your work and demonstrate a deep understanding or your chosen topic. The discussion in your report must be technical and quantitative wherever possible.

How to submit

Similar to the class assignments, **you will submit the code and the report to Gradescope (note: there will be no autograder part).** Find the appropriate project and make your submission into the correct project.
Important: Submissions sent to Email, Piazza or anything that is not Gradescope will not be graded.

Grading

The report will be graded following the scheme below:

- Code (30%): We will verify that the methods and rules indicated above have been followed.
- Report (70%): Subject to a working code.
    - Description of existing methods published in recent computer vision research.
    - Description of the method you implemented.
    - Results obtained from applying your algorithms to images or videos.
    - Analysis on why your method works on some images and not on others.
    - References and citations.

Assignment Overview

In class, Szeliski 2010 (chapter 11), and some of Forsyth and Ponce 2012 (chapter 13) we discussed window-

based approaches to estimating dense stereo correspondence. In this project you will implement such approaches and evaluate it on some standard stereo pairs. Additionally you are expected to implement stereo correspondence methods incorporating an advanced energy minimization or graph-cut method of your choice. Your results must be represented by disparity maps that are close to the ground truth images, also included in the example links below.

1. Method Requirements

Implement the basic stereo algorithm of taking a window around every pixel in one image, and search for the best match along the same scan line in the other image. You will do this both left to right and right to left. Remember: Because of depth changes (discontinuities), some pixels visible in the left image are not in the right image and vice a versa. So you will match in both directions. Keep in mind this approach relies on epipolar constraints where it is assumed epipolar lines in both images are horizontal. This may be different when you are testing your code with your own images (if you decide to do so).

Start by implementing the simplest thing imaginable: Look for the smallest difference between the template window (source) and the proposed location window. Use the *sum of squared differences measure (SSD).* We are going to take the definitions from: https://software.intel.com/content/www/us/en/develop/home.html

SSD is defined by:

$$S_{tx}(r,\ c) = \sum_{j=0}^{tplRows-1} \sum_{i=0}^{tplCols-1} \left[t(j,\ i) - x\left(r+j-\frac{tplRows}{2},\ c+i-\frac{tplCols}{2}\right)\right]^2$$

In this section you will implement this method without using OpenCV functions that directly perform stereo correspondence operations, for example cv2.StereoBM_create and others similar to this.

Research stereo correspondence methods that incorporate advanced energy minimization or graph-cut techniques. Integrate them to your code and use them to improve your results. Notice that SSD generates a lot of artifacts near occluded edges. The main objective is to remove them such that the disparity maps are close to ground truth images. For this section you are allowed to use external libraries and functions such as networkX, minimum_cut, etc... If you are not sure whether you are allowed to use a specific function or library please ask on Piazza.

2. Input Requirements

Use images like the ones in the Middlebury Stereo Datasets to test your algorithms. You must include image pairs that belong to the same scene and present a different point of view.

3. Final Results

Use ground truth images to compare your results. In order to present your output images, you must define a

metric to quantify how close your results are the these reference images. This will involve putting enough effort in analyzing and processing your input images to achieve this objective.

3.1 Image results

Include disparity maps that demonstrate depth variations for all objects in the scene. Make sure you present all cases mentioned in the project instructions.