COMP417: Assignment 4
Due Monday, April 10 at 11:59pm

April 22, 2017

1 Depth from stereo disparity (6.25pts)

In this exercise you are going to solve the depth from stereo disparity problem that we discussed in class. You will be given pairs of stereo-rectified images from the KITTI and Middlebury datasets, such as the ones outlined in Fig. 1. The KITTI dataset is currently the state-of-the-art benchmarking service for various visual tasks that are relevant to self-driving cars. Its data is taken from a car equipped with the same sensors as most self-driving cars[1]. The Middlebury dataset[2] is one of the oldest computer vision benchmarking services out there, and has become very popular for its stereo evaluation suite. You are encouraged to explore these datasets to become aware of what they offer. In the meantime, we are going to use the following rectified stereo image pairs for this assignment:

For each of these image pairs you will compute the disparity map in pixels, using block matching along horizontal epipolar lines, as we saw in class. Note

[1] `http://www.cvlibs.net/datasets/kitti/setup.php`
[2] `http://vision.middlebury.edu/stereo/data/scenes2014/`

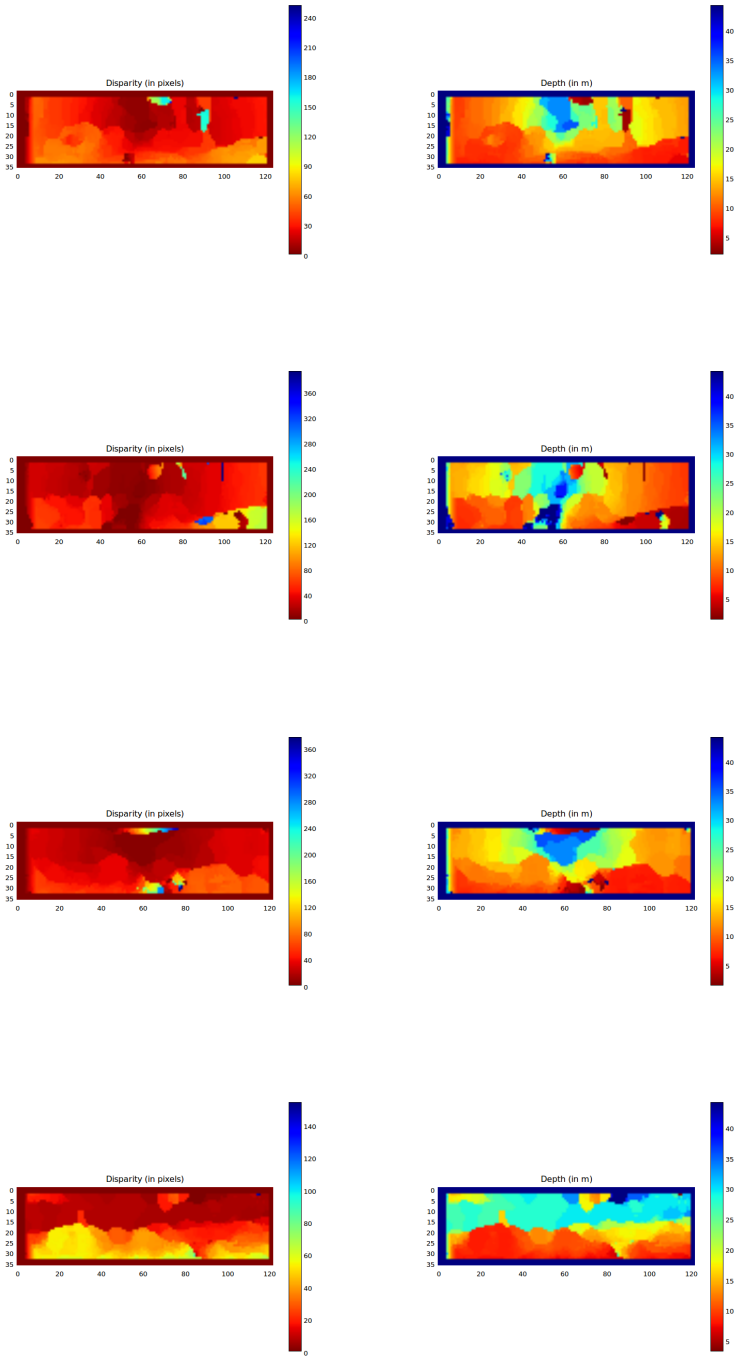
Figure 1: Stereo image pairs 1 to 6 are from the KITTI dataset. The baseline between the cameras is about 0.5 meters. The 7th image pair is from the Middlebury stereo dataset. The baseline is about 0.2 meters.

that this is definitely not the state-of-the-art algorithm for stereo disparity, and your implementation will most likely not run in real-time, but it is enough to get you thinking about the problem.

Given the stereo camera calibration and rectification parameters you will compute the depth map in metric coordinates from the pixel disparity map. For example, these disparity maps have been obtained with patches of width 64 pixels, and the sum of squared differences as the distance metric for matching patches along the horizontal epipolar lines. The most common options for a distance metric for comparing similarity between image patches are:

- sum of absolute differences (SAD), $\sum_{i,j} |L_{ij} - R_{ij}|$, where L,R are the left and right patches being examined.
- sum of squared differences (SSD), $\sum_{i,j} (L_{ij} - R_{ij})^2$
- normalized cross-correlation (NCC), $\sum_{i,j} \frac{(L_{ij} - \mu_L)}{\sqrt{\sum_{ij} L_{ij} - \mu_L}} \frac{(R_{ij} - \mu_R)}{\sqrt{\sum_{ij} R_{ij} - \mu_R}}$

The expected results you should get look approximately like this:



1.0.1 Starter code

You are given starter code in your comp417 repository under `estimation_and_vision_assignment/python/stereo_disparity_map.py`. Do `git pull` to

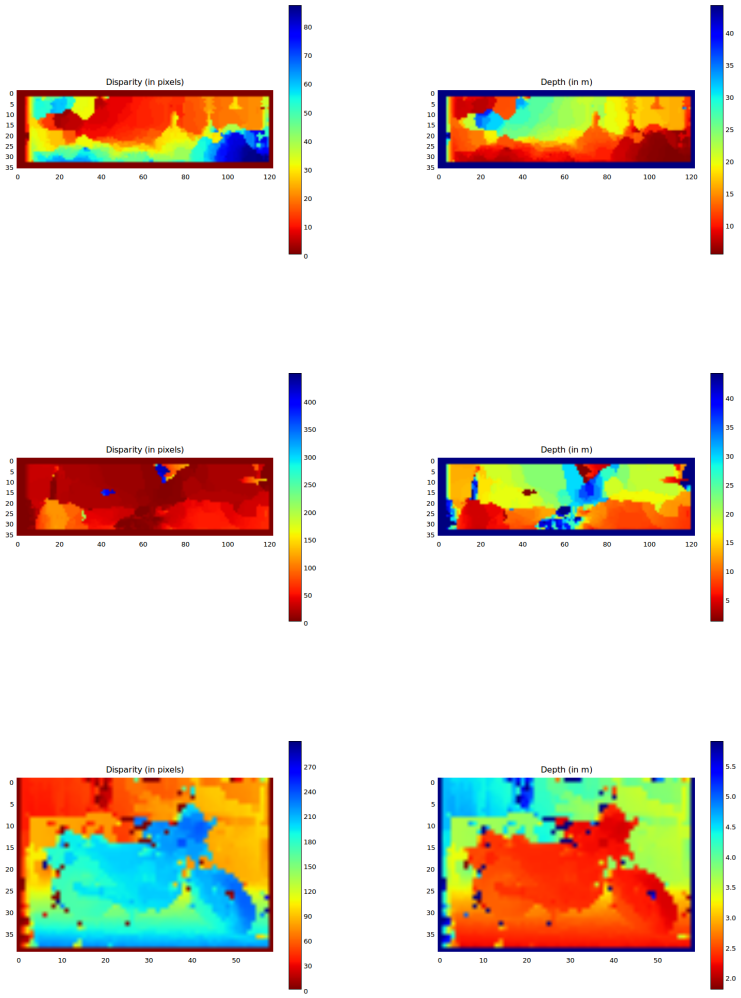


Figure 2: Example disparity and depth maps for the image pairs presented above. No post-processing has been applied to the disparity maps to encourage smoothness.

from your comp417 repository in order to receive it. The way to run this code is:

```
./stereo_disparity_map.py ../latex/stereo_images/images0/kitti_1_
left.png ../latex/stereo_images/images1/kitti_1_right.png ../resources/
kitti.yaml
```

and similarly for the motorbike photo: `./stereo_disparity_map.py .. /latex/stereo_images/images0/middlebury_1_left.png ../latex/stereo_ images/images1/middlebury_1_right.png ../resources/middlebury.yaml`

The `.yaml` files contain rectification parameters and other parameters for the two datasets.

1.0.2 What to submit

Your implementation of disparity and depth estimation in `estimation_and_vision_assignment/python/stereo_disparity_map.py` using sum of squared differences for matching. Also, the following 14 images: 7 disparity maps and 7 depth maps for each of the image pairs mentioned above. Your files should be named `kitti_[1-6]_disparity.png`, `kitti_[1-6]_depth.png`, `middlebury_1_disparity.png`, `middlebury_1_depth.png`

1.1 Bonus (2 pts)

Try to use GPU acceleration to make your stereo block matching code run closer to real-time. For the size of these image pairs for example, it could be made to run closer to 1 sec with a good implementation and on a modern GPU.

2 Monte Carlo Localization (6.25pts)

In this exercise you are going to implement Monte Carlo Localization (i.e. localization in a known occupancy grid map, using particle filters), as discussed in class. Your robot is going to start by being completely lost in the environment, so particles are going to be spread out uniformly at random in the world. Then, after many measurements, its position is going to be constrained and the particles are going to converge to a small cluster. The main mechanism for survival of the fittest among particles will be: which particles are more likely to have generated the laser scans that the robot is observing now?

2.0.1 Starter code

Do `git pull` under your comp417 repository to get the starter code. The functionality that you need to implement is marked using comments in the file `estimation_and_vision_assignment/python/monte_carlo_localization_v2.py`. To run your code, `cd path/to/comp417/estimation_and_vision_assignment/` and execute the following commands on three different terminals:

```
rosbag play -r 0.5 laser_controls_and_odometry.bag
roslaunch estimation_and_vision_assignment monte_carlo_localization_v2.launch
rosrun rviz rviz
```

Note that the -r argument to rosbag reduces the playback rate by half, so that measurements are sent more slowly to your program. When rviz initializes, go to `File` > `OpenConfig` and then load the configuration file in `estimation_and_vision_assignment/resources/comp417.rviz` which is going to start the visualization of laser scan messages, frames of reference, and particles. Save this configuration file as the default in your `/home/username/.rviz/default.rviz`, so you won't have to do this every time you restart rviz. What you will see initially, before the robot makes any measurements is the following:

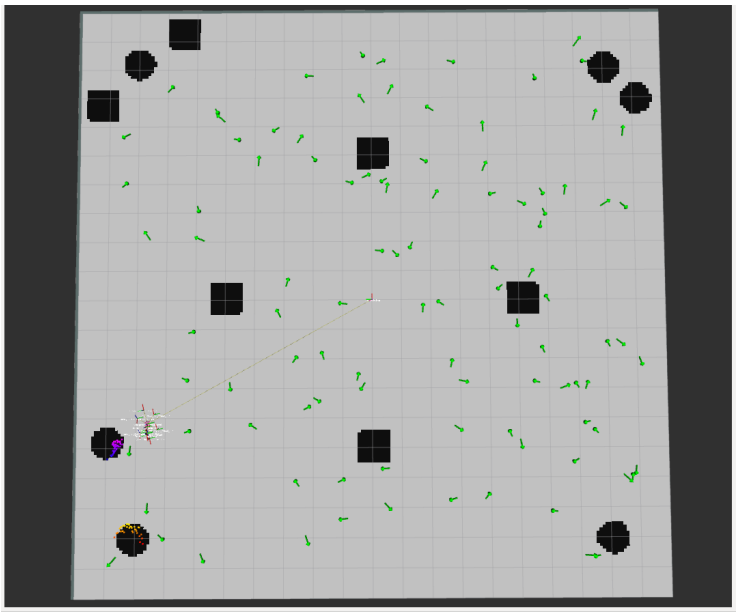


Figure 3: Uniform initialization of particles in Monte Carlo Localization, within the boundaries of a workspace. Your task is to make the particles cluster around the robot.

What you need to submit: in addition to your code, a video recording of the rviz visualization demonstrating your particles converging from beginning to end. Your video should be named `mcl_firstname_lastname.mp4/avi/ogg`.

3 Grading scheme

3.1 Q1

- (3 pts) Finding the best match along the epipolar line
- (2 pts) Computing the depth from the disparity map
- (1.25 pts) for the required images

3.2 Q2

- (2 pts) Particle filter resampling
- (2 pts) Laser scan simulation for a given particle
- (2 pts) Particle prediction error and weight updates
- (0.25 pts) Video

Note in particular that for Q2, you will be able to get almost full marks even if your particle filter does not converge, as long as you implement the resampling, the laser scan simulation, and the weight update steps correctly.

4 How to submit

Submit all your work in a file called `estimation_and_vision_assignment.zip` that contains your extensions to the provided starter code, as well as the 14 images and the video. Submissions will be done on myCourses.

5 A note on indentation

Please make sure your editor uses spaces instead of tab characters, otherwise your TAs will have a hard time reading your code on their editors, which might be different from yours.