Candidate evaluation – Luka Karlić
Junior Computer Vision Engineer

Introduction

This document provides description of tasks devoted to evaluation of candidates applying for the *Junior Computer Vision Engineer* position at Gideon Brothers. The candidate is expected to solve the problems described in this material independently, representing her/his expertise and skills adequate for the respective position.

The goal

Present software development skills by solving the problem of dense stereo correspondence and disparity / depth map generation.

Figure 1: A stereo pair with a ground-truth disparity map

The problem

Computer vision capabilities are of a vital significance for nearly any type of autonomous robotics application. One of the most fundamental capabilities in terms of robot's perception of the environment is depth estimation on images obtained from cameras. There are many approaches to solving the problem, but the approaches which generate the best results are extremely computationally intensive and cannot operate in real time.

The basic 2-view stereo vision approach uses two cameras with distinct horizontally (or vertically) displaced viewpoints – for the purpose of this evaluation we will use the horizontal case and define viewpoints as "Left" and "Right". Computing depth in this way involves the estimation of **disparity** – the distance between the x-coordinates of pixels in the left and right image corresponding to the same point in space. An object observed in both camera views will in the right camera view be shifted to the left (to a lesser x position) in respect to the left camera view. The number of pixels for which the object is shifted to the left is inversely proportional to the object's distance from the camera and can be used to compute the depth given the camera calibration parameters.

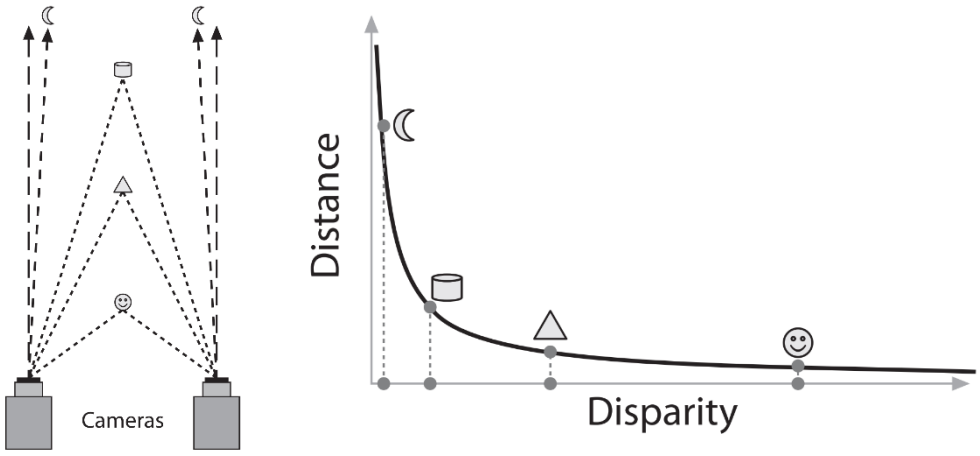

Figure 2: Stereo cameras and disparity vs distance(depth)

For the purpose of this task, we can disregard the problem of **stereo camera calibration and rectification** and assume that all input images are **rectified stereo pairs**. In rectified stereo pairs, the images have been precomputed in a way that potential lens distortions are removed and the features in both images are aligned to reside on the same image line in both images. This allows us to match features by searching only along the horizontal axis of the image, which greatly reduces complexity.

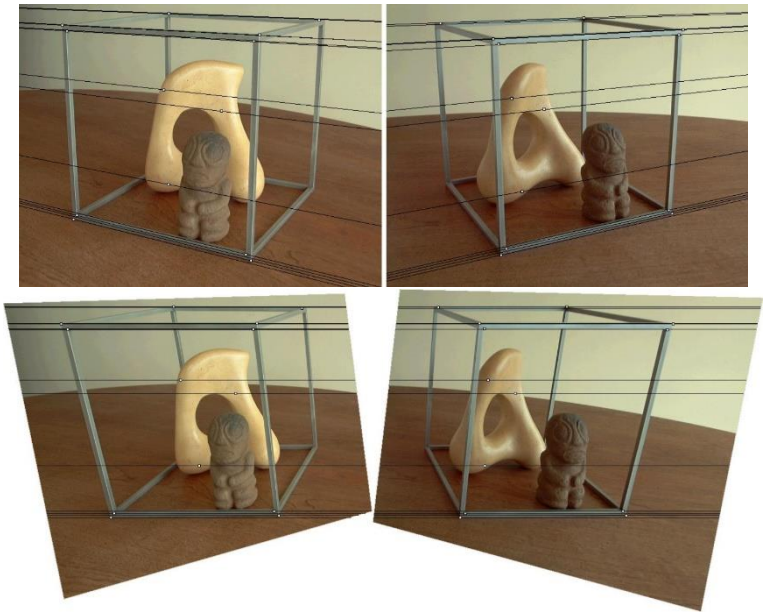

Figure 3:Rectified(bottom) vs unrectified(top) images

Task

In this task the candidate should implement stereo matching using the classic winner-take-all algorithm and produce disparity maps for input stereo pairs.

In the winner-take-all algorithm, a range of possible disparity hypotheses (from d_{min} to d_{max}) are tested for each pixel of the image by calculating a **matching cost** for the pixels. The disparity with the minimum overall matching cost is declared the winner and assigned to the disparity map for each pixel.

More info on the algorithm (and other general stereo topics) can be found here:

http://media.ee.ntu.edu.tw/courses/cv/18F/slides/cv2018_lec14.pdf

The **matching cost** is a measure of similarity for two pixels in different images. Many cost functions exist - whether hand-constructed or learned, but all output a value which can be used as a similarity score and has an extreme (minimum or maximum, depending on the cost function) at the point of a match between two features.

The basic matching cost is the **Absolute difference,** where the modulus of the difference in luminous intensity (grayscale value) between two pixels is calculated.

When aggregated within an area, it is usually called the **sum of absolute differences (SAD)** https://en.wikipedia.org/wiki/Sum_of_absolute_differences

The **Census** cost, for example, uses a transform of the input image to encode the relation of a local neighbourhood to the center pixel of the neighbourhood (each pixel is compared to all its neighbours and the comparison results (greater than or not greater than) are encoded into the value of the resulting pixels. The match is calculated by computing the **Hamming distance** (the number of different bits) between two pixels being compared.
https://en.wikipedia.org/wiki/Census_transform

As a single pixel provides little context for matching, usually a surrounding area, or a window, is used to aggregate the neighbouring costs and match the local context corresponding to the object. This also introduces a smoothing effect in the final disparity map. The areas can be fixed or adaptive, with constant or variable weights based on filters or segmentation. The simplest aggregation is summing the local costs within a surrounding area – as in the sum of absolute differences. **A fixed size rectangular (square) area is to be implemented for this task.**

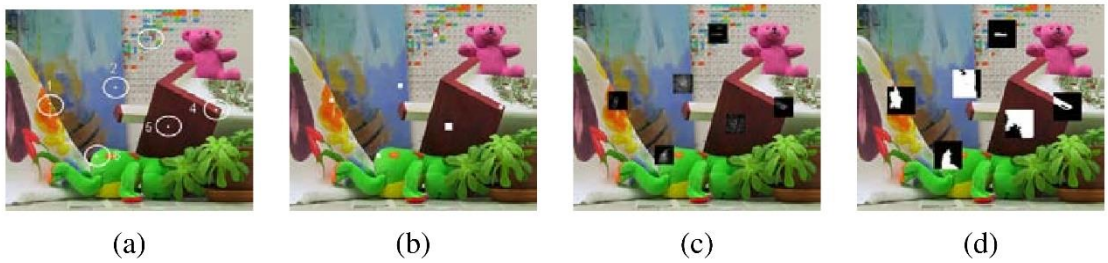

Figure 4: Example of various aggregation windows, with their depicted weights (white=1.0, black = 0.0) a) selected points b) rectangular windows (variable size) c) weighted/filtered windows d) segmentation based windows.

***Note**: A major challenge for the implementation is how to structure the processing to avoid calculating the matching cost for each pixel and disparity for more times than it is necessary, and that the aggregation within the window does not incur unnecessary performance penalties. The evaluation of the submitted solution will place special emphasis on this part.*

Evaluation and deliverables

Deliverables

The solution should load two colour or grayscale images, convert them into grayscale if required, run the stereo matching and output a disparity map as a grayscale image. The disparity map should correspond to the **left** image.

The solution should implement the following run-time configurable parameters for the algorithm:

- Cost function
  - absolute difference cost
  - Census cost with an 8-neighborhood for each pixel
- Window size - an integer radius for a square window: the area is 2*r+1 wide and tall. For example, a window with size = 2 defines a 5x5 pixel window around a center coordinate.
- Disparity range d_min...d_max - the range of hypotheses for which the disparity is evaluated

The cost function should be implemented in a way so that it is encapsulated and can be easily selected per each run of the algorithm and that more functions may be added later.

The algorithm **must** be implemented in **C++** using **OpenCV** for image processing. A "modern" C++ programming style is strongly encouraged. A **CMake** script must be provided for compilation.

Alongside the code and the compilation script, the solution must be documented with the following:

- A README file containing:
  - A personal note on the implementation reflecting on the critical implementation parts
  - Guide to compiling the program
  - Guide to running the program
  - Example results obtained with the program

The solution is to be submitted via GitHub as an open repository.

Evaluation

The Candidate should validate the algorithm on the publicly available KITTI dataset (http://www.cvlibs.net/datasets/kitti/eval_stereo.php) and the Middlebury dataset (http://vision.middlebury.edu/stereo/data/). KITTI and Middlebury have pre-rectified images from stereo camera pairs (colour and greyscale for KITTI, colour only for Middlebury) with accompanying ground truths. The candidate may choose which KITTI image(s) to use to demonstrate the solution. From the Middlebury set, however, the candidate **must** show the performance on the following samples: "Tsukuba", "Venus", "Teddy" and "Cones".

The challenge is expected to be solved using **Linux Ubuntu (**version 18.04. or later)

General information

Deadline: The task is expected to be finished within **14 days**. Partial or early submission is possible.

Note: If the candidate encounters any difficulties in the assignment solving process, he or she is expected to communicate them as quickly as possible.