COMP37212: Computer Vision

Coursework 3

Introduction

For this practical assignment you should use Python/OpenCV to develop the code, and present your results as a formal report in **PDF format**.

You will find the self-assessment code snippets useful and you are free to use them as part of this assignment.

You should use the supplied images for your processing and include them in your report.

For ease of marking, please lay out your report in sections using the titles given in this document.

You may want to create separate Python programs for the different parts.

Intended Learning Outcomes

By the end of this assignment, you should be able to:

- Implement computer vision code using Python/OpenCV
- Choose combinations of techniques in order to solve a computer vision problem (that is, a computer vision pipeline, or workflow)
- Implement software to create a 3D model from a stereo pair of images
- Perform processing on regions of images depending on their depth into the scene

3D Demonstration

(a) (b)

Figure 1: See a 3D image. (a) Right image. (b) Left image. Look at these images with your eyes crossed. Eventually, you will be able to see a single image and should be able to see the depth into the scene.
The images are placed right-left so that, when you cross your eyes, your left eye sees the left-hand image and the right eye sees the right-hand image.

1 Stereo Imagery

You should use the supplied image pair of photography umbrellas (source: `https://vision.middlebury.edu/stereo/data/scenes2014/` – for interest, see the paper 'High-Resolution Stereo Datasets with Subpixel-Accurate Ground Truth' by Daniel Scharstein, Heiko Hirschmuller, et al).

Figure 2 shows the image pair and below is the calibration data for them.

(a) (b)

Figure 2: Stereo Pair. (a) Left image. (b) Right image.

```
cam0=[5806.559 0 1429.219; 0 5806.559 993.403; 0 0 1]
cam1=[5806.559 0 1543.51; 0 5806.559 993.403; 0 0 1]
doffs=114.291
baseline=174.019
width=2960
height=2016
```

The key to these parameters is shown in the table below.

cam0,1	camera matrices for the rectified views, in the form [f 0 cx; 0 f cy; 0 0 1], where f is focal length in pixels, cx, cy is principal point (note that cx differs between view 0 and 1)
doffs	x-difference of principal points, doffs = cx1 - cx0
baseline	camera baseline in mm
width, height	image size

However, to make the processing faster, we resized the images to 740 × 504. You might need to take that into account in your calculations below.

You have been supplied with a small program that creates and displays a disparity map (`disparity.py`). You can use this as a starting point.

1.1 Focal Length Calculation

The paper says that they used two Canon DSLR cameras (EOS 450D with 18–55 mm lens) in medium resolution (6 MP) mode.

This type of camera has a physical sensor size of 22.2 mm × 14.8 mm and in 6 MP mode, the resolution is 3088 × 2056.

Calculate the focal length in millimetres that the two cameras were set to (cam0 is the left camera).

1.2 Disparity Map

Use the supplied program `disparity.py` as a starting point. The `getDisparityMap()` function can accept the original images as greyscale or as edge detected images (0 & 255). Try both and see which produces the best result for these purposes.

The `getDisparityMap()` function returns a floating point image of the same size as the input images with the values corresponding to the disparities. This function takes two additional parameters (number of disparities and block size).

Display an image from the (normalised) disparity map for the umbrella images. Vary the parameters (with Trackbars) until you get an image that looks like the scene without too much noise. Your image might look something like Figure 3.

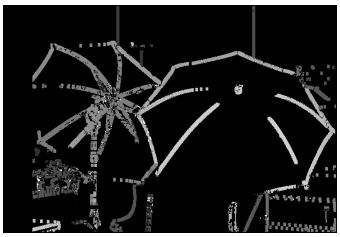


Figure 3: Example disparity map.

1.3 Views of the Scene

The depth into the scene (Z mm) can be calculated by

$$Z = baseline \frac{f}{d + doffs}$$

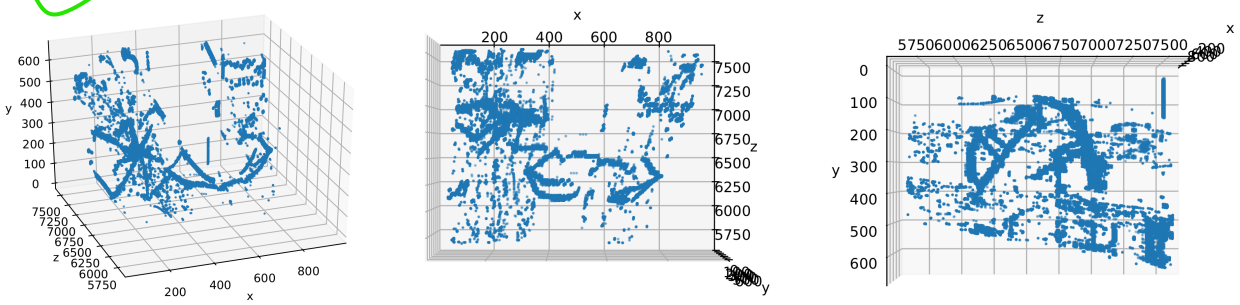
where baseline is in mm, and disparity (d), focal length (f), and $doffs$ are in pixels. (You can convert the baseline to metres if you want to get the depth in metres.)

Remembering that pixels represent the light arriving from the scene from different angles, it is possible to calculate the X and Y world coordinates using similar triangles. These can be derived from the focal length, the depth into the scene and the pixel coordinates, x and y.

Loop through the disparity map and calculate the real world coordinates (X, Y, Z) for each pixel in the disparity map. You can do this in the `plot()` function in the supplied `disparity.py`, but you will need to pass in some other parameters. Then display a 3D plot of the scene.

Display the resulting data on a 2D plot viewed from above (X, Z coordinates) and the side (Y, Z). You might find the matplotlib function `ax.view_init(elev, azim)` comes in useful. (You might find it easier to get the views you want in the matplotlib 3d plot if you swap y and z in the plots.)

Examples are shown in Figure 4. Hopefully, you will getter better results than these if you vary the disparity parameters.



(a) (b) (c)

Figure 4: Example views of the scene. (a) 3-D view. (b) Top view. (c) Side view.

Alternatively, you may decide to create images of your view as in Figure 5. In that case, you do not need to supply a 3-D view, just the top and side views.

---

2 Selective Focus

Figure 6a shows a mobile phone that has a stereo camera. (For information: It only uses the stereo camera for portraits and will only take the picture if it detects a face – it cannot be used as a general stereo camera.)

The three images in Figure 6 show what you can do with the image after you have taken it. You can blur the background to varying levels and turn the background greyscale.

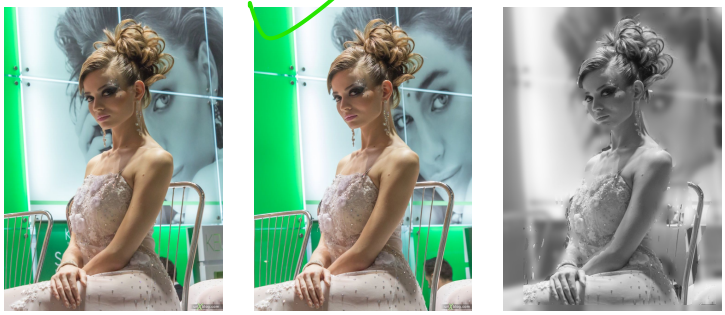
In this section, you will use the code you have written above to try this yourself (make a copy of your code used for the Stereo section so that you keep both sets of code).

Figure 5: Side view of the scene drawn into an image.

(a) (b) (c) (d)

Figure 6: Stereo camera on Moto G6. (a) Phone showing the two cameras on the back. (b) Background blurred slightly. (c) Background blurred more. (d) Background made greyscale.

You should use the supplied stereo pair, `girlL.png` and `girlR.png` as shown in Figure 7.



(a) (b) (c)

Figure 7: Stereo Pair. (a) Left image. (b) Right image. (c) Example output.

Depth can be approximated from the disparity map (without scale) by

$$depth = \frac{1}{disparity + k}$$

Use a Trackbar on your window to vary k as well as the parameters you did previously. You may find it useful to also display the calculated depth image as well as the disparity image to be able to see a possible segmentation.

The depths can then be scaled to the range [0,255] and image arithmetic performed on this depth image with the original. (How will you decide on which are object pixels and which are background pixels in the depth image?)

Your output can be either of the following:

- A greyscale image with the background heavily blurred.
- A colour image with the background heavily blurred.
- A colour image with the background changed to greyscale.

Hints: Pass greyscale images to `getDisparityMap()` rather than edge images. Use a larger block size so that more of the image is filled.

You may find that parts of the image have been incorrectly classified (object/background). Small amounts of this throughout the image are acceptable.

Marking Scheme

Total marks available: 20

1.1	Calculate focal lengths of cameras	3 marks
1.2	Create and display disparity map	3 marks
1.2	Implement trackbars (show images of the whole window with the trackbars in different positions showing the effect on the disparity map)	2 marks
1.3	3D, top and side view of umbrellas scene	4 marks
2	Perform a different type of processing on the image foreground and the image background	3 marks
2	Implement trackbars (show images of the whole window with the trackbars in different positions showing the effect on the output)	2 marks
	Discussion and conclusions	3 marks

Check-list

Have you:

- Created a well-written and well-formatted report?
- Included input, intermediate, and output images with useful labels?
- Chosen a size for your images so that the details can be seen, but not so big that the document covers too many pages (see the images in this document)?
- Included results (e.g. output images) so that report is understandable in itself?
- Included the code in the appendix?

Submit your report as a single pdf document on Blackboard.

Do not include any other files or zip it – just a pdf.