Computer Vision Report

by Andrew Howell (tzjn72)

Introduction

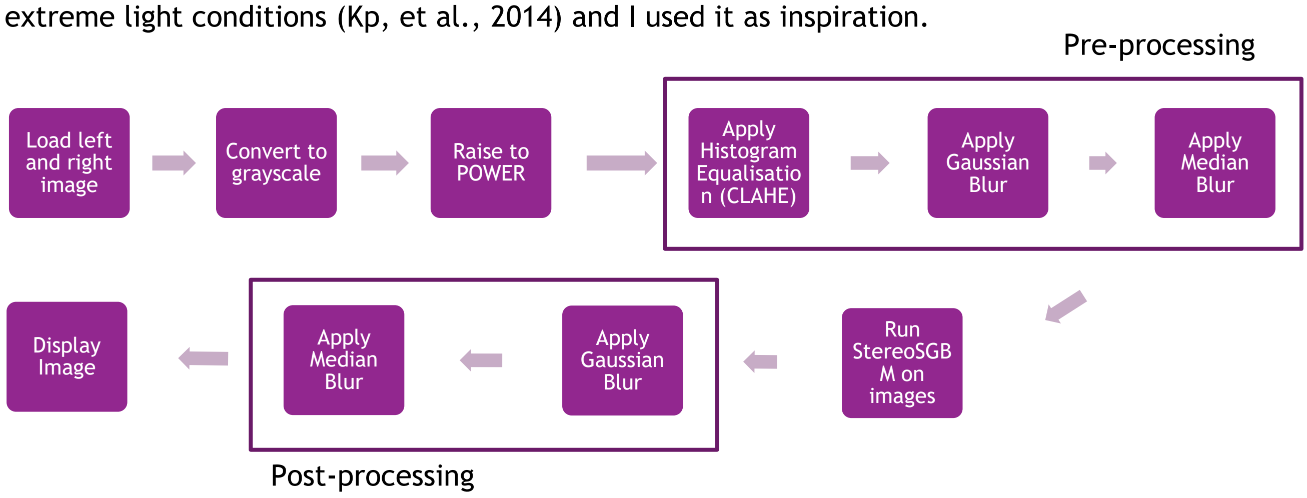
In this assignment, I used pre- and post-processing, as well as parameter manipulation in the StereoSGBM class, in order to maximise the accuracy of the generated disparity map from the two stereo images provided. I used similar methods in order to optimise the YOLO functionality.

Disparity Mapping

My disparity mapping comprised only of dense stereo matching, using open-cv's Semi-Global Block Matching algorithm.

Dense Stereo Matching Pipeline

This pipeline has been shown to improve the results of dense stereo matching under extreme light conditions (Kp, et al., 2014) and I used it as inspiration.



Pre-processing

In order to maximise the results of the SGBM algorithm, I used CLAHE to boost the contrast in the left and right images. This should improve its effectiveness in extreme lighting conditions.

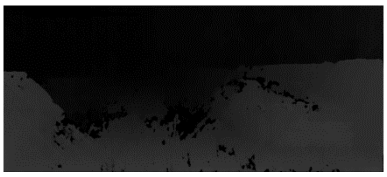
Next, a Gaussian Blur of window size (7x7), sigma 7/9 is used to blur away any gaussian noise caused by the CLAHE function. A medium blur of window size 9 is used to remove salt-and-pepper noise caused by the CLAHE algorithm.

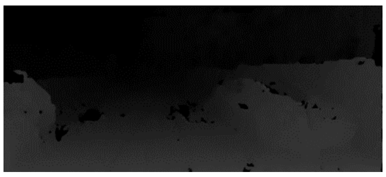
The CLAHE function used a clip limit of 5 and a tile grid size of (5,5), which struck the perfect balance of boosting contrast without degrading the image.

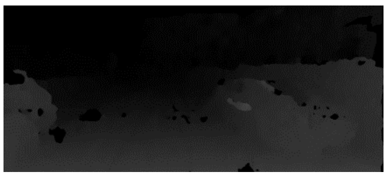
SGBM Parameters

Block size

I found that a block size of 7 created a smooth disparity map without creating lots of tiny disparity regions.


Block Size 3


Block Size 7


Block Size 11

P1 and P2

These parameters control the smoothness of the disparity map. After tweaking, I found the values 600 and 4000 to be best as the map is smooth enough to still show details.

Max Difference

This sets the maximum allowed difference in the left-right disparity check. With a value of 20, it allows the disparity map to see up-close objects and more accurately derives the distance from the map.

Mode

I used the MODE_HH mode for the SGBM algorithm, which makes it run the full variant of the algorithm. It is a two-pass algorithm variant and moves in 8 directions instead of 5.

The issue with the HH mode is that it takes up a lot more memory: O(W * H * numDisparities) bytes (Anon., n.d.). However, I found no such issues on these images, perhaps because they are not HD images. I did not see a significant increase in processing time with the full-scale algorithm either.

Post-processing

I run another pass of filtering speckles with a max speckle size of 4000 and a max difference of 50.

In order to reduce Gaussian noise in the resultant image, I ran a Gaussian blur of window size (5,5), sigma 5/9. This would also help to blur the line between disparity regions, making for a smoother disparity map from which to calculate object distances from.

A further median blur with window size 5 was used to reduce any salt-and-pepper noise.

Note that this post-processing blurs used smaller window sizes/standard deviations as to reduce the blur effect.

Output

The image is output to the screen. If the user has chosen the cropped image, the image is cropped accordingly and only this cropped image is used for distance ranging.

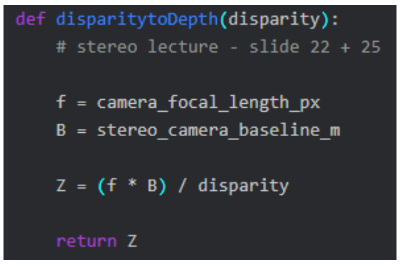
YOLO

The left (colour) image is used to form a tensor from which the YOLO algorithm will work.

The image is cropped based on the user’s choice and the image is converted to a LAB colours space in order to have it's light channel histogram equalised and reconverted to RGB.

Each box represents a subarray, I take this subarray and find the first quartile disparity value. This removes any outliers from the broad object region, as well as gives a disparity from the closer side of the object.

This disparity is converted into a distance using the function:



```
def disparitytoDepth(disparity):
    # stereo lecture - slide 22 + 25

    f = camera_focal_length_px
    B = stereo_camera_baseline_m

    Z = (f * B) / disparity

    return Z
```

Results

Qualitative Evidence

Over the first 10 frames of the video, through my own interpretation, spotted:

2 pedestrians

50 vehicles

My program spotted:

1 pedestrian

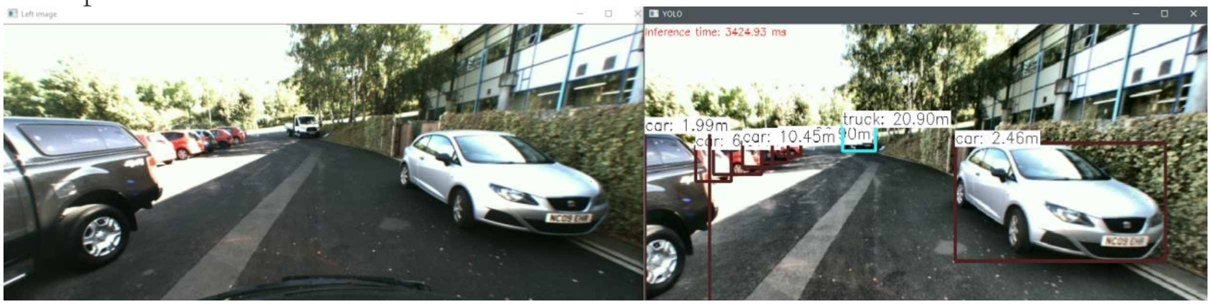
48 vehicles

39 distance estimates roughly within 2m of real distance

6 misidentified objects

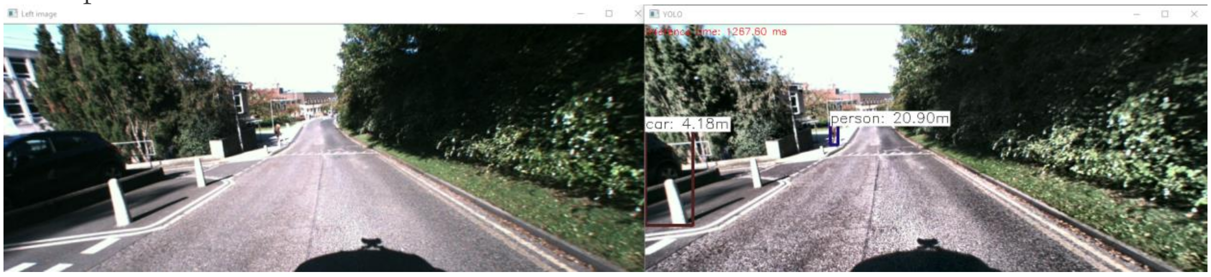
Quantitative Evidence

Example 1



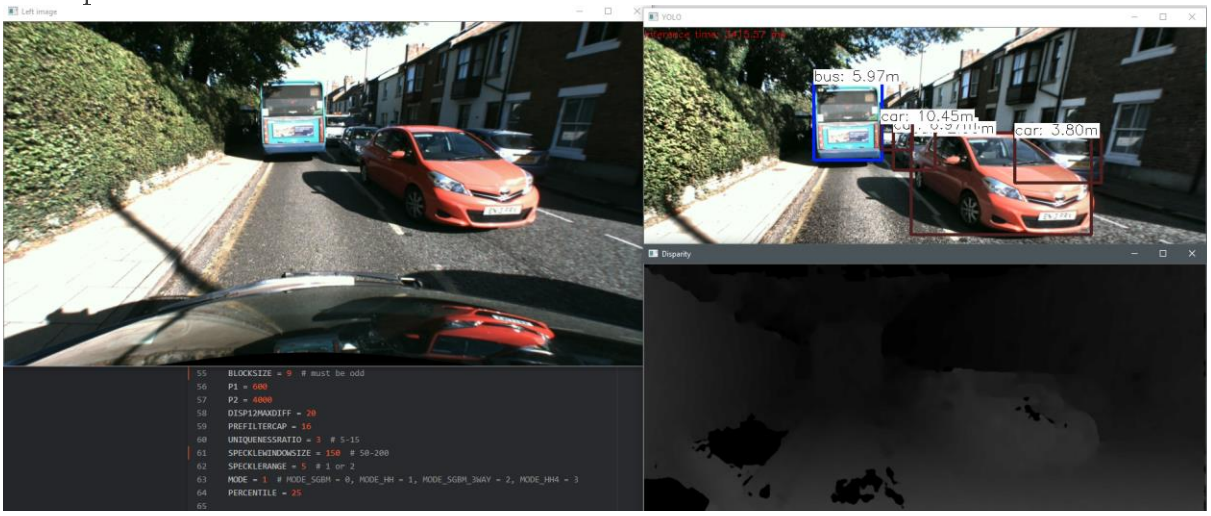
Note the strong distance ranging on this first occluded left car, as well as the overlapping cars having accurate distances.

Example 2



It sees the occluded car and the far away pedestrian easily and with accurate ranging.

Example 3



```
BLOCKSIZE = 9  # must be odd
P1 = 600
P2 = 4000
DISP12MAXDIFF = 20
PREFILTERCAP = 16
UNIQUENESSRATIO = 3  # 5-15
SPECKLEWINDOWSIZE = 150  # 50-200
SPECKLERANGE = 5  # 1 or 2
MODE = 1  # MODE_SGBM = 0, MODE_HH = 1, MODE_SGBM_3WAY = 2, MODE_HH4 = 3
PERCENTILE = 25

```

The disparity map clearly shows the vehicles, but still struggles in high lights (bottom left has 0 feature points).

References

Kp, A., Reddy, V. & R., H., 2014. *Enhancement Technique for Improving the Reliability of Disparity Map Under Low Light Condition*. Vallabh Vidyanagar, Researchgate.