Computer Vision Coursework Report

by vzbf32 for Software, Systems and Applications III

Integrating YOLO and Stereo Disparity

The first task was to integrate object detection and a disparity map. I started by experimenting with the provided examples and trying to work out what each file did.

It became apparent that I would essentially need to join the different files. I took the existing code and split it into my own modules which were each loaded into **app.py**, which was then responsible for producing a disparity map, using YOLO to detect objects and then drawing boxes around detected objects on the video, along with a label for what the item was and its distance from the camera (based on the disparity map).

To calculate the distance of an object, we select the middle 40% of a bounding box of an object, and then calculate the median disparity of those pixels. This works reliably and gives a decent estimate in most cases.

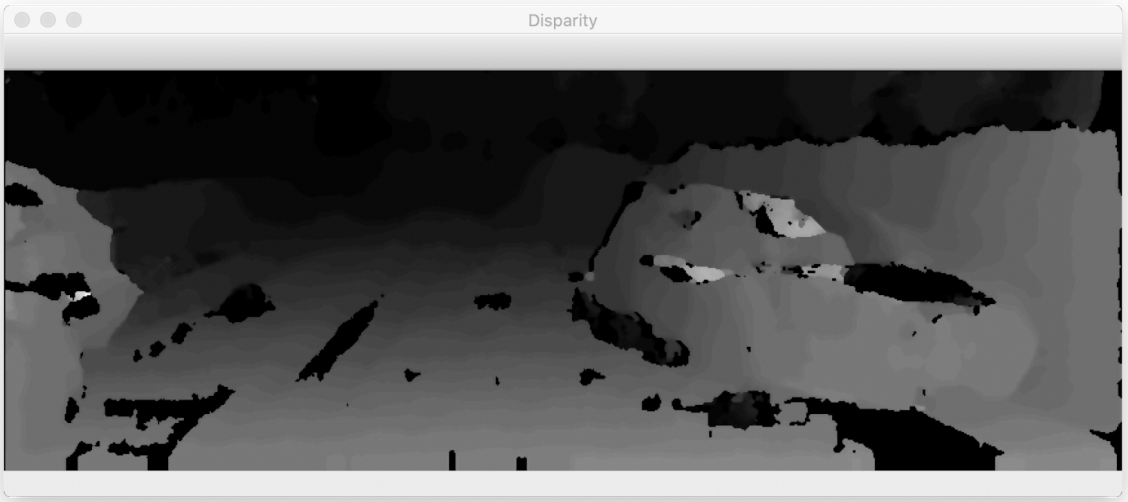
Experimentation

I varied the **max_disparity** value in the given **stereo_disparity.py** to see what effect this would have. I found that smaller values made the disparity map calculation faster, whereas higher values were slower but gave a more accurate result and a greater range of disparities.

Once the disparity map is calculated, I carry out bilateral filtering on the end result. This has the effect of softening the image but still preserves edge detail. Prior to this, I tried a simple Gaussian blur, but found that it lost too much edge detail.

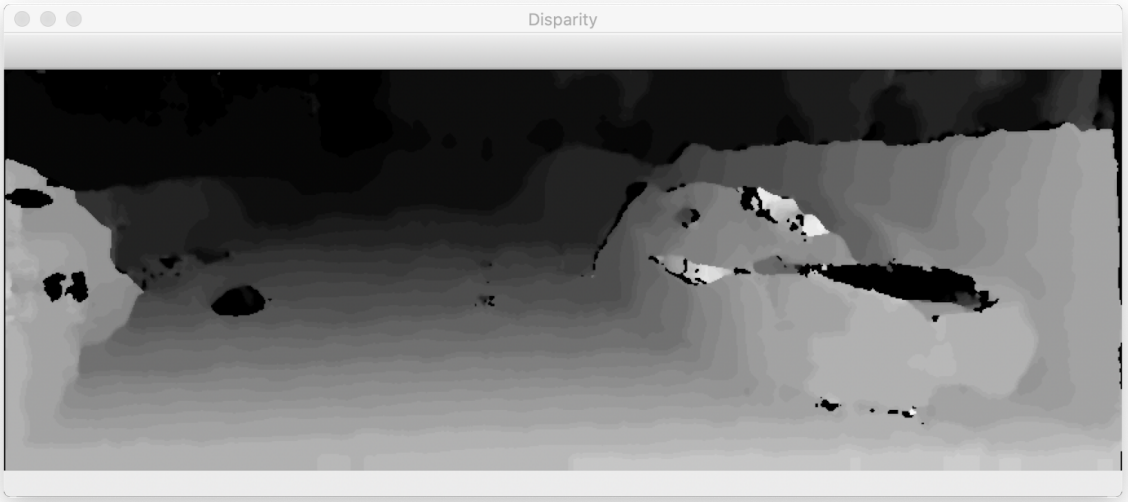
Equalising Histograms

While I was able to clearly see objects on the disparity map (like people and cars), I observed that there were a lot of holes. By applying an equalising histogram to the grey left and right images, I found that the number of holes reduced significantly.

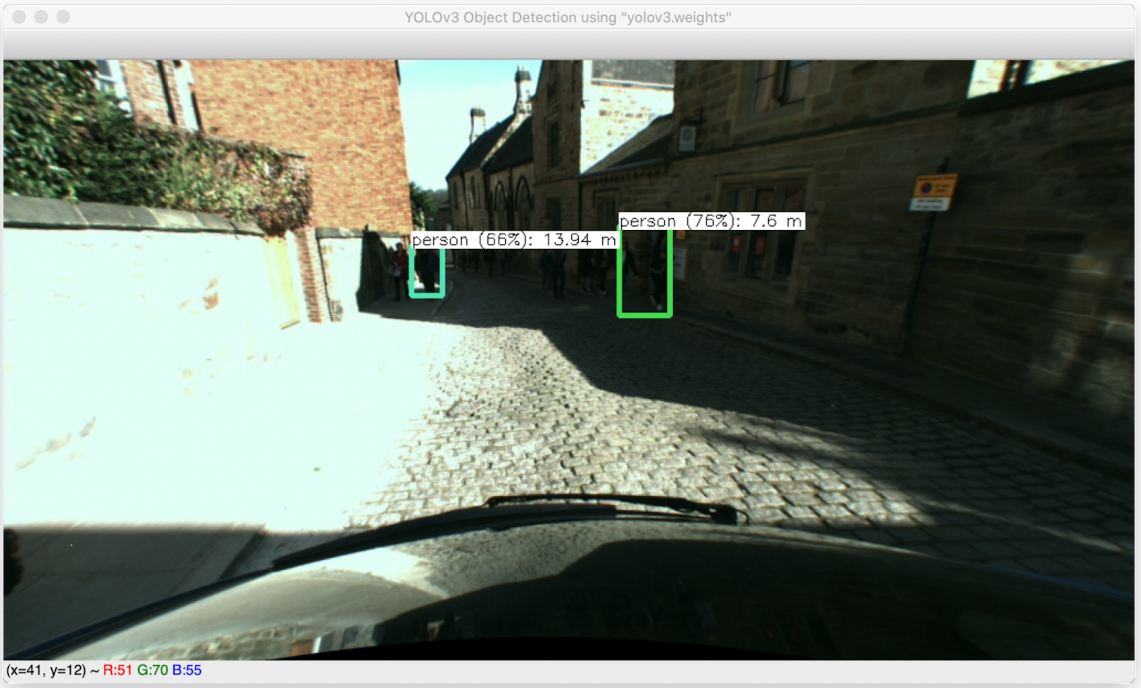


Above: *Disparity map before equalising histogram of image*

Below: *Disparity map after equalising histogram of image*

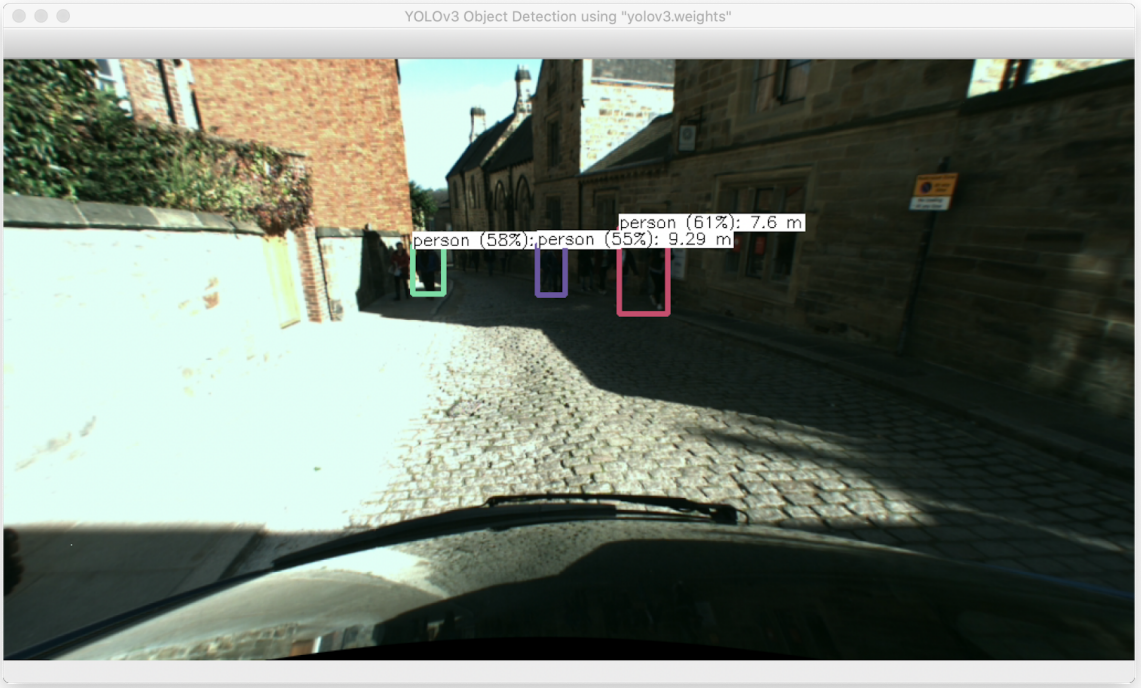


The built in **cv2.equalizeHist()** function only applies to greyscale images and so I was unable to apply this to the left colour image without losing colour information. Instead, I applied CLAHE (contrast limited adaptive histogram equalisation) on the lightness channel (converting the image from RGB to LAB and then back to RGB after applying CLAHE) before attempting object detection, and I found that this made object detection much more successful, particularly in very bright / dark areas of the image where contrast was lacking.



Above: *Object detection before applying CLAHE to assist with contrast. Observe how few people are detected in the scene.*

Below: *Object detection after applying CLAHE to assist with contrast. Observe how many more people are detected in the scene.*



Cropping

One easy way to improve performance of the application was to crop out parts of the image before attempting processing. I explicitly did not calculate a disparity map, or attempt object detection, in the lower portion of the image (where the car bonnet is, with all its reflections), or indeed in the upper portion of the image (for the sky).

I noticed issues with the disparity map struggling to match regions (or even incorrectly matching regions) in the sky, especially when the image was blown out, and as this was skewing the range of disparities I found it best to avoid performing computation on these parts of the images entirely.

Above: *Disparity map with incorrectly identified region of sky, before cropping out the sky.*

One possible improvement for the lower part of the image would have been to have applied a mask in the shape of the car bonnet rather than explicitly cutting off the entire bottom portion of the image. I found that occasionally a person would walk in the space between the bonnet and the left/right of the image but as this was part of the cropped-off region they would not be identified. However, this tradeoff is acceptable as it prevents YOLO from detecting any objects in the reflection from the car bonnet.

Conclusion

It is clear that the pre-processing measures assist with object detection and stereo disparity. Applying CLAHE clearly helps, as demonstrated above with the people in shadow, and also in the following shot where we clearly detect a number of different objects in the scene:

Above: *Detecting numerous cars in the scene through YOLO. Note the mistakenly identified 'bus' at the back, although buses and trucks are similar so this isn't a serious issue.*

Video

The included video is sped up significantly. The actual recording speed is ~0.9 frames per second.