Assignment 2: Notices
CS4053: Computer Vision
Sean Durban (14320715)
Due: 20/11/2017

Solution Description

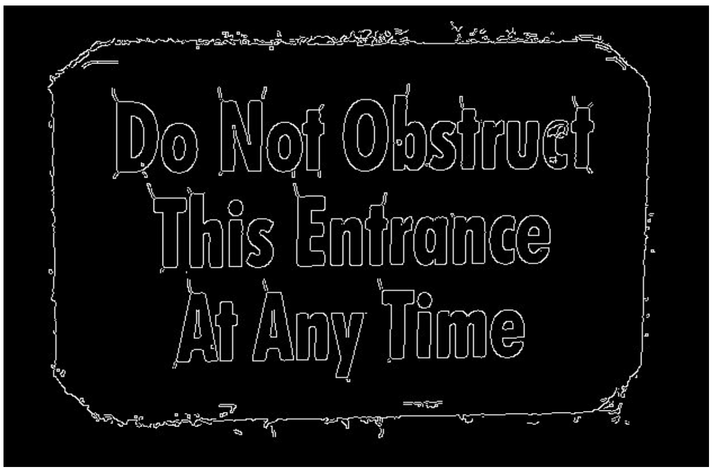
My solution involves finding the sign contour then cropping the image to perform thresholding on the sign itself in order to identify the regions of text.

The first step is to remove noise from the input image. Here mean shift segmentation is used. It allows clustering based on both colour and pixel spatial locality, we also don't need to make any assumptions on the shape or number of clusters in the image. The parameters used were found through repeat testing over all the images. Deciding upon a higher spatial window and a lower colour window radiuses. Having a higher colour window radius resulted in some of the text being augmented in a way that made it less detectable. Also the higher spatial window radius helped remove the noise from around the notice. The overall purpose of this step is to provide an improvement on the input for our purpose of recognising the sign and then the text.

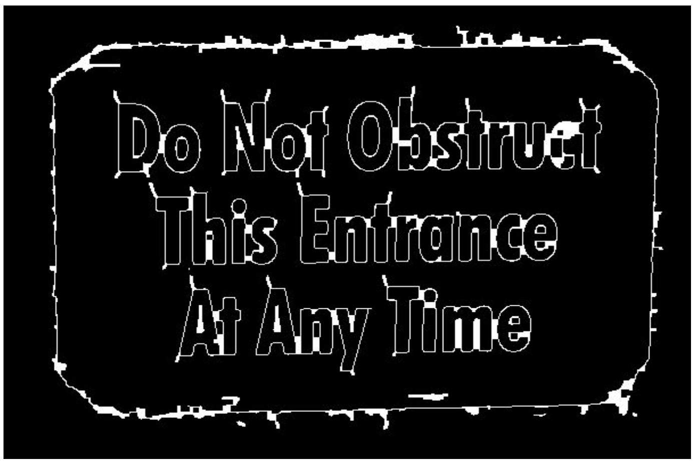
Then the resulting image of mean shift is converted to greyscale and canny edge detection is applied. Edge detection is very susceptible to noise and even though the initial step of Canny is a form of Gaussian blur I still found the mean shift to be necessary. Canny is found to attain higher accuracy vs other edge detection techniques such as Sobel in common use. I too found it to be more accurate in detecting the narrow outline of the sign. The threshold parameters were found through testing to be optimal for the test set.

Once the program identifies all the edges in the program then closing is used to ensure these edges are well defined. Closing is the act of applying dilation followed by erosion. This step is important as there's potential for noise or other objects disrupting a sign's edges. The default parameters and kernel (3x3) are applied. Below is example of where this transformation is important:

Before closing/ Canny result

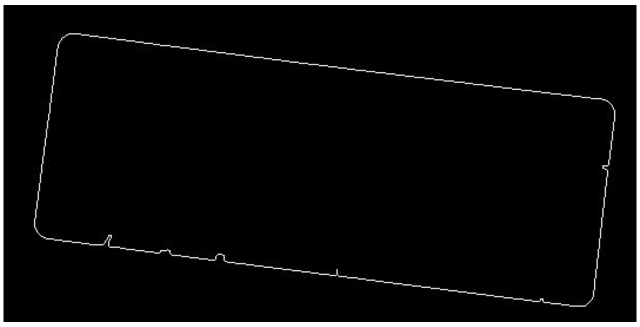


After closing



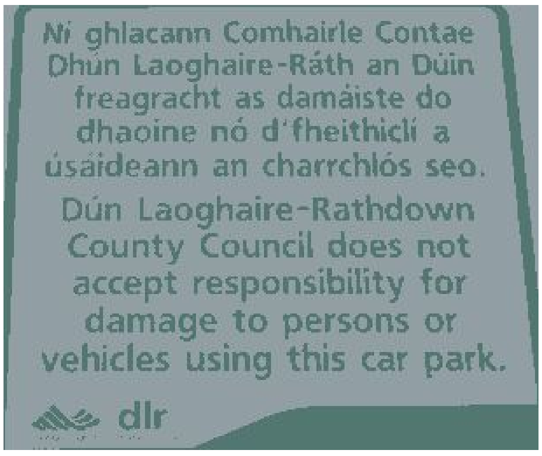
Connected components analysis is then used on the result of the closing transformation. This returns all the the contours or regions in the image and a hierarchy. The hierarchy is specifically important as it gives us the location of a contour in relation to other contours. From this we can tell if a contour is inside another contour. This is paramount to finding our sign in an image, which at a high level should be a contour with other contours (i.e the letters) inside it.

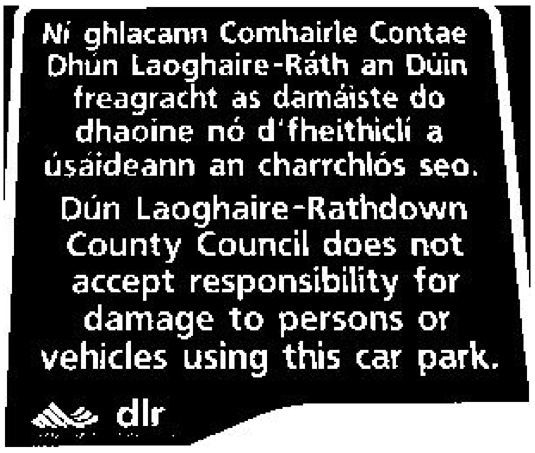
I implemented an algorithm to identify the contour which is the sign. It iterates over each contour, first checking if it's of a suitable area. This check is done to remove the likelihood of some small contour containing many other contours being considered as the sign. If it's of suitable area then it counts each child and grandchild (child of the child) of that contour. It's important to check the grandchild for finding the sign since certain letters such as 'O' are represented as 2 contours. In order to be counted a contour must be over a certain area as to not count noise and contours too small to be letters. Also the area must be below an area threshold, this is in place to avoid a large contour such as a background wall which has the sign as a child contour being able to count the sign and all it's child contours. The contour which has the largest area and over a certain threshold of valid child and grandchild contours is classified as the sign. Below is example of the sign contour being found for notice 6:



Once the sign contour is determined the image is cropped to just include the sign. This allows us to apply operations only on the sign and being able to ignore other potential objects and noise in the input image

As the sign is the main focus of the image now we apply k-means clustering to minimise the colours present in the image. The value of k is kept low (5) with the assumption that the sign background and the text colour are the main colours in the image but still allows for images where there may be different colour text or where there's another dominant colour eg) red in hazard signs. The result of this is converted to greyscale and then a binary threshold is applied. This is with the intention of separating the text from the rest of the sign, with the assumption that they should be contrasting colours. Below is the result of k-means and threshold for cropped notice 1:


Ní ghlacann Comhairle Contae
Dhún Laoghaire-Ráth an Dúin
freagracht as damáiste do
dhaoine nó d'fheithiclí a
úsáideann an charrchlós seo.
Dún Laoghaire-Rathdown
County Council does not
accept responsibility for
damage to persons or
vehicles using this car park.
dlr


Ní ghlacann Comhairle Contae
Dhún Laoghaire-Ráth an Dúin
freagracht as damáiste do
dhaoine nó d'fheithiclí a
úsáideann an charrchlós seo.
Dún Laoghaire-Rathdown
County Council does not
accept responsibility for
damage to persons or
vehicles using this car park.
dlr

Connected components is applied and rectangles are formulated around these contours. The intention is for the rectangles to incase the inner contours of the sign, which from our previous high level definition of a sign should be the letters. Again these rectangles are only recorded if they're considered valid, which checks they're consistent shape to valid letters or text regions. For example an inner contour with height above 150px is far more likely to be non text than a very large piece of text. Note: at this stage the rectangles only cover single or a handful of letters.

Another algorithm is implemented in order to join overlapping or adjacent rectangles. We compare each non-used rectangle to every other rectangle. If they overlap or are within a predetermined range of one another then it creates a new rectangle which is the max dimensions of both rectangles, using following formula:

x= min(r1.x, r2.x)   y= min(r1.y, r2.y)
width = max(r1.x+ r1.w, r2.x+ r2.w) - x
height = max(r1.y+r1.h, r2.y+r2.h) - y
Where r1 = rectangle 1 and r2 = rectangle 2

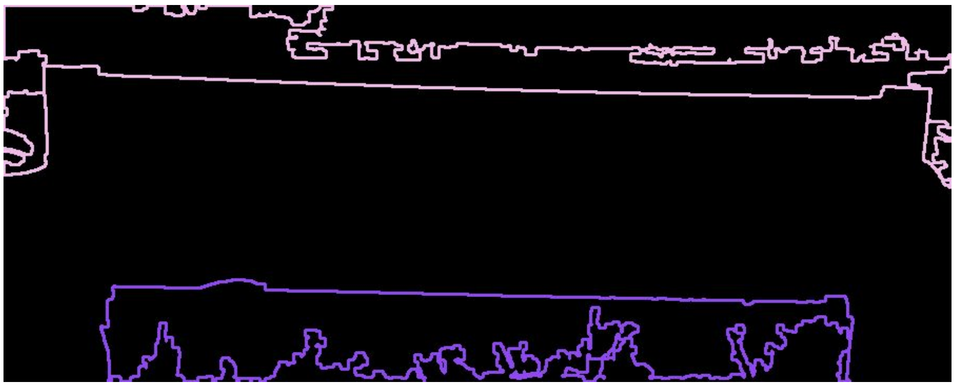
The second rectangle is marked as used as the new rectangle now represents it. This cycle is repeated until no rectangles are merged and they're all disjoint.

The resulting rectangle vector is the program's solution. These are drawn over the original input image in order to visualise the results.

Solution Discussion

Overall I think the solution has clear steps and a clear objective at each step. Much of the thresholds and operation/transformation parameters used were found through extensive testing set. But these are still 'magic numbers' and may not be optimal or even work for other images. Although the testing set is quite diverse and I'd argue that results gained from this solution means that the should perform reasonably well for any reasonable input.

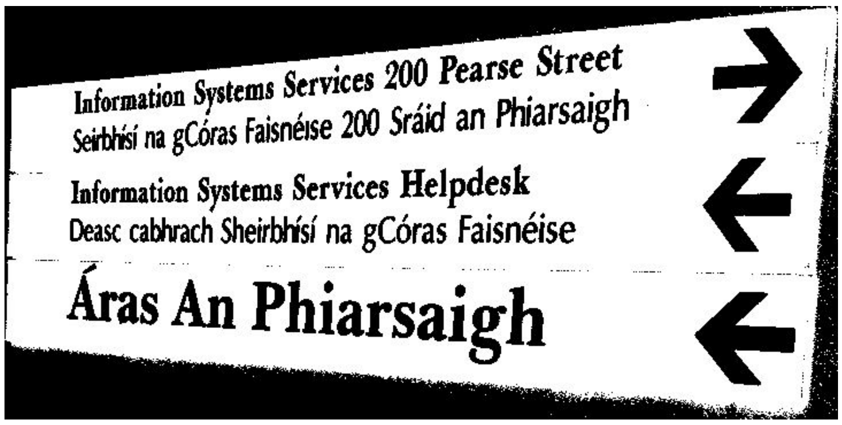
A major flaw of my solution is that it relies upon finding the sign contour correctly. If it fails in doing so (as it does for notice 8) then it will only recognise the text in the contour it has falsely assigned as the sign. In the case of notice 8 the error stems from the edge detection step, as it fails to find the vertical edges of the sign correctly and is merged with the adjacent contours of the cars. There's no way to recover from this error if it occurs.The image below shows the contours it believes could be the sign and then chooses the bottom (purple) one. The sign itself isn't even considered a contour.



Notice 8 is quite noisy and the sign is not the only obvious object in the image. I was able to gain better results by being more aggressive with my noise reduction but this had negative effects to other image results. A solution for notice 8 and similar issues would be to increase the amount of noise reduction by increasing the radius of the colour and spatial window parameters of the mean shift operation.

There's also many cases of checking if contours or rectangles are valid. As these use hard coded values the solution would have issues with either smaller images or much larger images. A solution for this would be to get samples of valid text segment sizes at a certain image size and build a valid range. Then scale this range to the size of the input image.

Another improvement would be to change the approach after cropping the image. Currently the solution uses thresholding, but this is not ideal. A simple way to join adjacent letters and words would to be dilate the regions found in a binary region. But I could not do this due to the text being both white and black on separate signs. The threshold is currently used to separate darker segments from lighter segments. Therefore this works in clearly distinguishing black text from lighter backgrounds and it also works the opposite way in that a darker background is distinguished from lighter text. But for this reason I could not apply dilation as it would result in losing the lighter text. Example of this is notice 5 which has white text and a blue background. The resulting threshold is shown below:

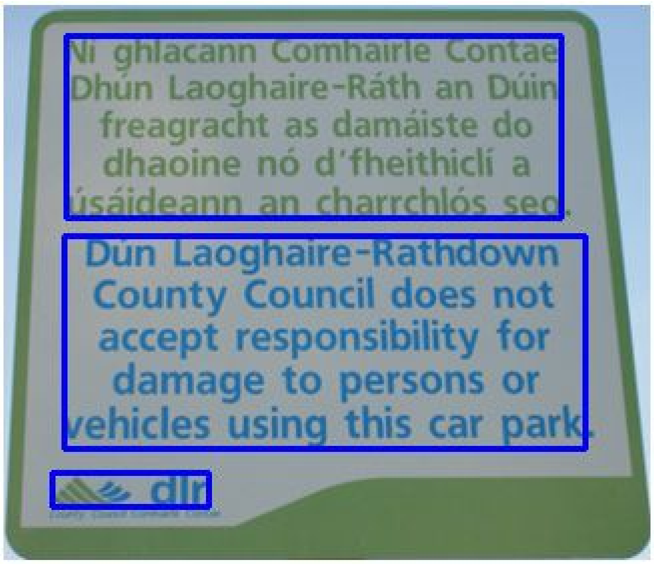

Information Systems Services 200 Pearse Street
Seirbhísí na gCóras Faisnéise 200 Sráid an Phiarsaigh
Information Systems Services Helpdesk
Deasc cabhrach Sheirbhísí na gCóras Faisnéise
Áras An Phiarsaigh

The alternative approach could be to potentially use edge detection again which wouldn't have any issues with text colour. Or determine the threshold to use based on the background of the text region. Which if successful would in turn allow dilation.

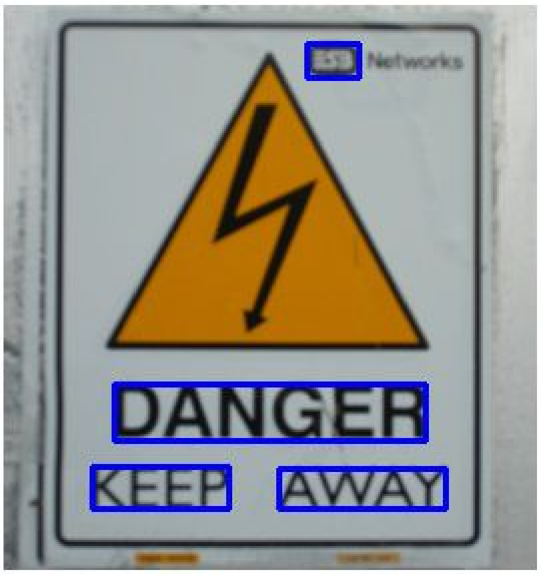
Results and DICE Coefficients

The following are the output images showing the text regions identified within bounding rectangles on the original input:

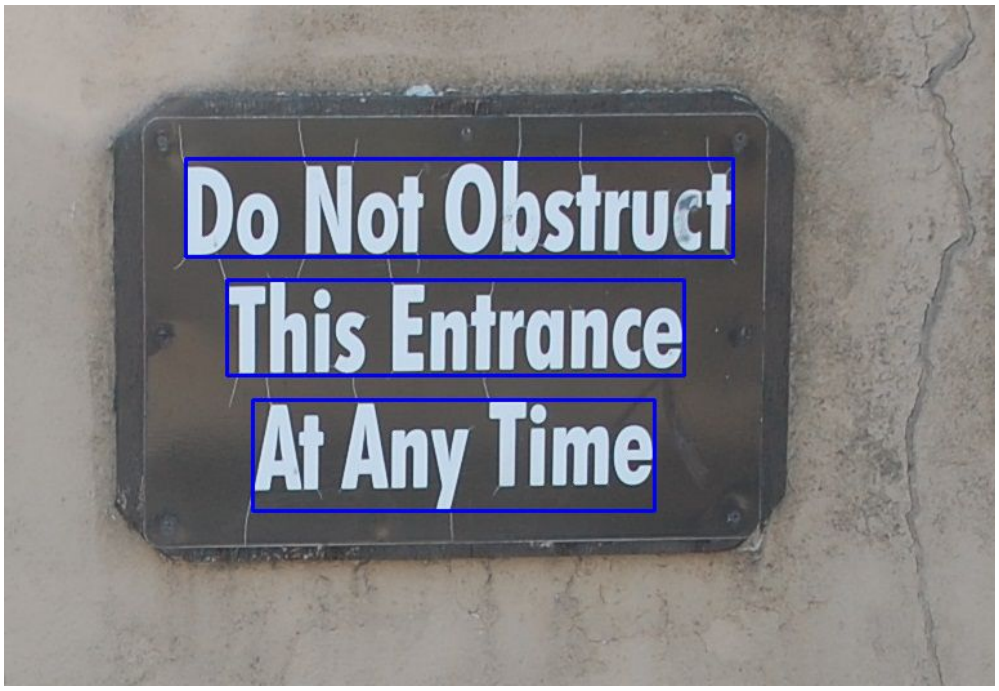
Notice 1


Ní ghlacann Comhairle Contae Dhún Laoghaire-Ráth an Dúin freagracht as damáiste do dhaoine nó d'fheithiclí a úsáideann an charrchlós seo.
Dún Laoghaire-Rathdown County Council does not accept responsibility for damage to persons or vehicles using this car park.
dlr

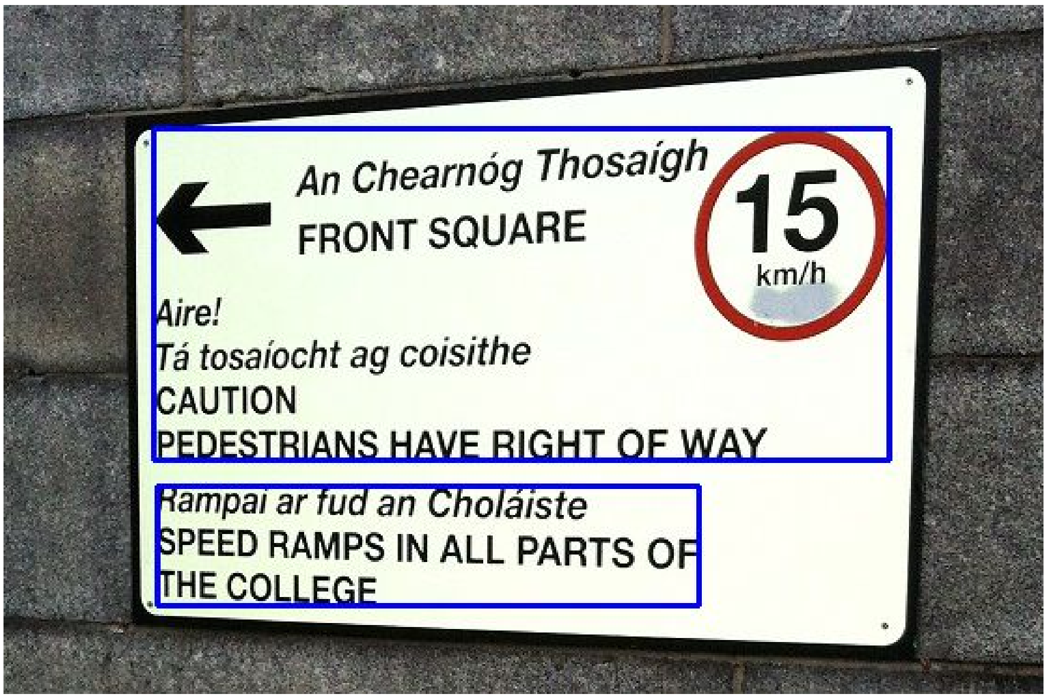
Notice 2



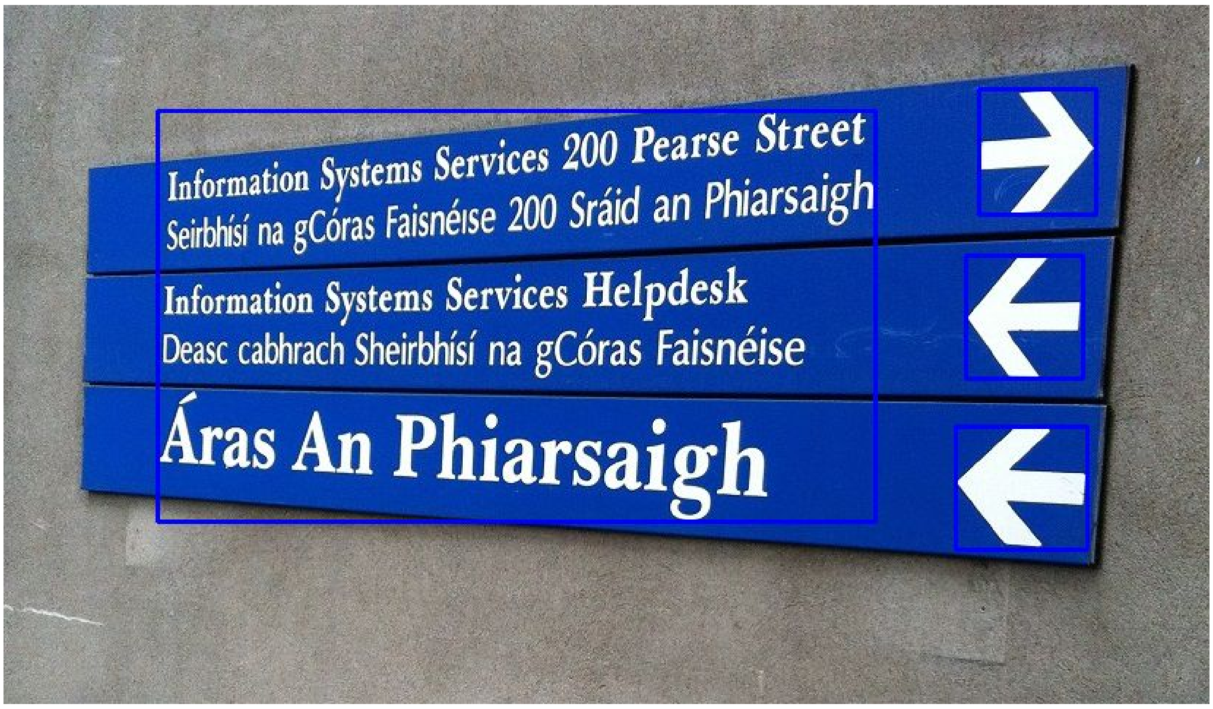
Notice 3



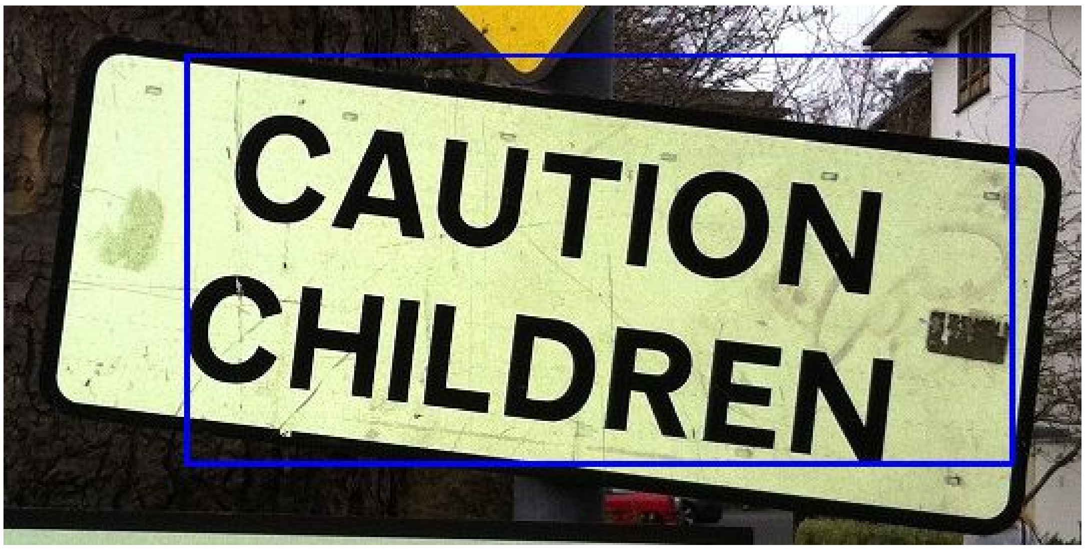
Notice 4



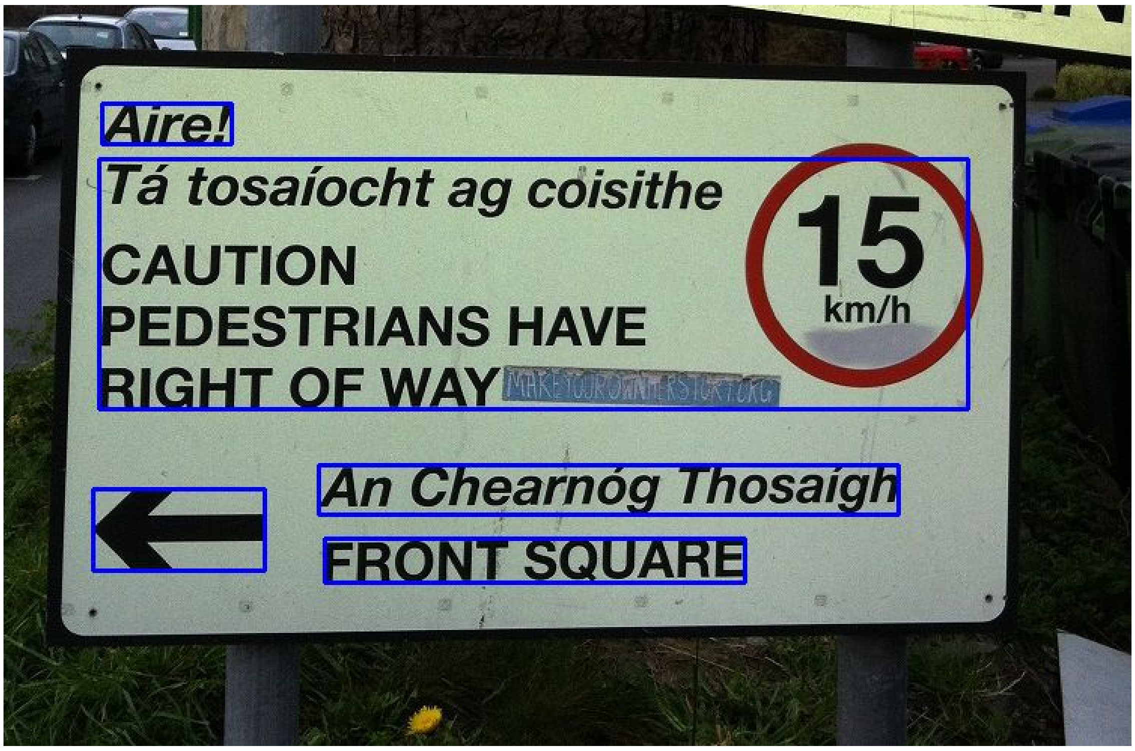
Notice 5


Information Systems Services 200 Pearse Street
Seirbhísí na gCóras Faisnéise 200 Sráid an Phiarsaigh
Information Systems Services Helpdesk
Deasc cabhrach Sheirbhísí na gCóras Faisnéise
Áras An Phiarsaigh

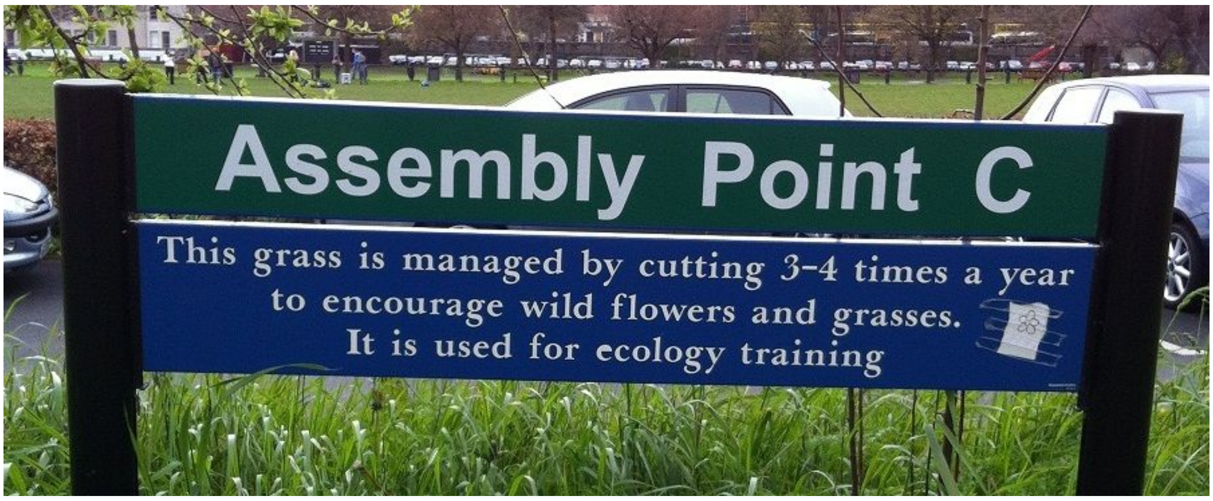
Notice 6



Notice 7



Notice 8


Assembly Point C
This grass is managed by cutting 3-4 times a year
to encourage wild flowers and grasses.
It is used for ecology training

The following are the DICE coefficients computed for the above results against the ground truths given. Calculated using the following formula :

DICE = (2* AO)/(AGT + ARR)

Where AO = Area of Overlap between groundtruth rectangles and result rectangles

AGT = Total area of groundtruth rectangles

ARR = Total area of result rectangles

Notice #	DICE coefficient (decimal)
1	0.852095
2	0.473572
3	0.493225
4	0.318022
5	0.302347
6	0.732145
7	0.615038
8	0
Average	0.4733055

Comments

One clear difference between my results and the ground truth is that my results although identifying the correct text regions do so as sentences and disjoint words rather than blocks of text. This is also the main reason for the poor DICE result even though the text is all identified correctly. This is due to the rectangle surrounding the block of text covering far more area vs my rectangles which decreases the overlap.

Another clear problem with my results is that they consider icons and graffiti on the sign to be text. This would have negatively affected my DICE scores too as the ground truths ignore these. I would argue that if the goal was to give context to a sign then these icons such as arrows should be recognised. Both notice 6 & 7 have what seem to be stickers on them which were of regular text shape and therefore considered to be text. This is incorrect and for this reason the rectangles extend further than the ground truth.

As mentioned in a previous section notice 8 fails to identify any text. This is due to an error in identifying the sign and potential solutions were previously discussed.

The average of 0.4733 is quite low and would be still low at 0.54 ignoring notice 8. I'd argue that the solution performed much better at identifying text then this average suggests. In some cases even identifying the region of text more precisely than the ground truth. This in my opinion suggests that it may be a poor metric for determining the accuracy of such a program.

The ground truths in general bound large bodies of text which results in them covering a lot of area which is not text. This penalises results which cover less area but still identify the text. Although it could be argued that getting blocks of text is more beneficial than sentences or words as we can then determine the context and split them up as we see fit.

The ground truths provided were done manually and in my opinion some them contrast with one another. Meaning it would be particularly hard to find a solution which gets a high DICE coefficients average for the given set and also performs well for other inputs. As they're done manually they ignore obvious defects to the human eye such as graffiti and stickers. But these are much harder to detect using vision operations. As a sticker may have writing on it and is it therefore considered valid or not? I would say it's not but it would be very difficult to find such a defect as it could be similar to a letter which is in a shadow etc. Then you run the risk of taking out letters that you assume to be defects. Therefore in my opinion it's better to include these defects and later filter them out.

Resources

1. Kenneth Dawson-Howe, A Practical Introduction to Computer Vision with OpenCV
2. Kenneth Dawson-Howe,https://www.scss.tcd.ie/publications/book-supplements/A-Practical-Introduction-to-Computer-Vision-with-OpenCV/Code
3. https://stackoverflow.com/questions/tagged/opencv
4. https://docs.opencv.org/3.1.0/
5. Dr.S.Vijayarani, Mrs.M.Vinupriya, http://www.rroij.com/open-access/performance-analysis-of-canny-and-sobel-edgedetection-algorithms-in-image-mining.php?aid=43752