15 How to approach vision problems

Computer vision has many practical applications and to demonstrate an understanding of the area it is necessary to be able to come up with potential solutions to real problems. When presented with an application problem the student should ask themselves what general type of approach would be most appropriate? The following list is not exhaustive but instead is intended to give the student a notion of the types of issues they might want to consider:

- □ Image preprocessing. In Binary vision do we need to use an opening and/or a closing, etc? In gray-scale or colour imaging do we need to use a Smoothing technique? If there is any non-linearity do we need to geometrically correct the image?

- □ Binary vision. Is the domain such that a sufficient distinction could be created between the foreground and background to allow segmentation by thresholding?

- □ Colour vision. Do we need to do some sort of analysis of the colours present in the scene? Could we solve the problem using some sort of summary representation such as a histogram or K-means.

- □ Region based vision. If the data cannot be segmented by binary vision but needs to be broken into distinct objects or regions would a region based technique such as split and merge allow reliable segmentation?

- □ Edge based vision. If the boundary of the objects would be sufficient to allow recognition (or whatever is needed in the question) then perhaps the edges should be detected and analyzed in some fashion (e.g. Hough, features, etc.). Also what sort of edge detector should we use (Roberts, first derivative, second derivative, …) and does this need to be followed by Contour Following, and the extraction of straight line segments, etc?

- □ Recognition. If approaching a recognition problem we typically need to extract some features (using one of the preceding technique) and then perform some explicit recognition step (e.g. Hough Transform, Template Matching, Chamfer Matching, Statistical Pattern Recognition, a cascade of strong classifiers, etc.).

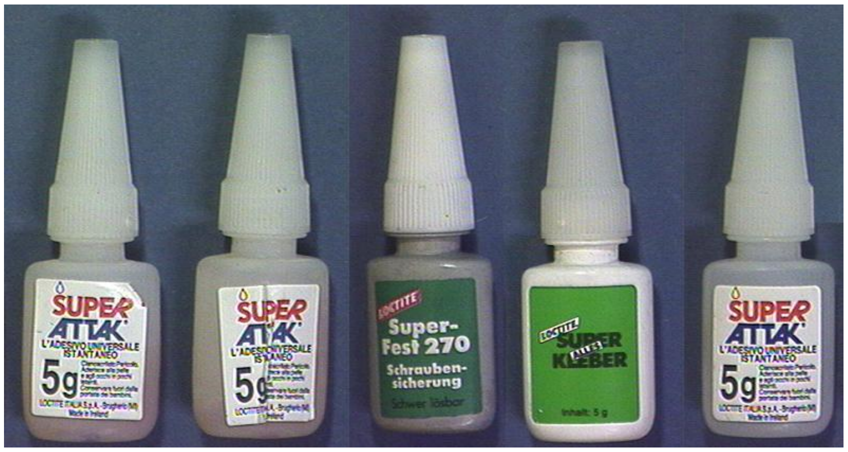
In the vision course associated with these notes application problems come up in two instances. In tutorials open vision problems are presented to the class to be tackled in groups, the idea being that each group would develop a series of steps to solve the problem. It is essential when doing so that the solution starts at the correct starting point (e.g. the specified images) and finishes at the specified target (e.g. recognized characters).

The second instance that these problems are faced is during the examination but in this case typically questions are asked in two parts. The first part asks about some specific area of vision, and the second part usually builds upon this to solve an application problem. It is essential that student provide details of the techniques that they use, and to give some idea of what is required a sample question and sample answer are provided here. Note that the level of detail required varies depending on the complexity of the answer. If there are 5 steps to solve a problem then less detail is needed to describe each these steps than if there were only 2 steps to solve the problem.

Sample Exam Question

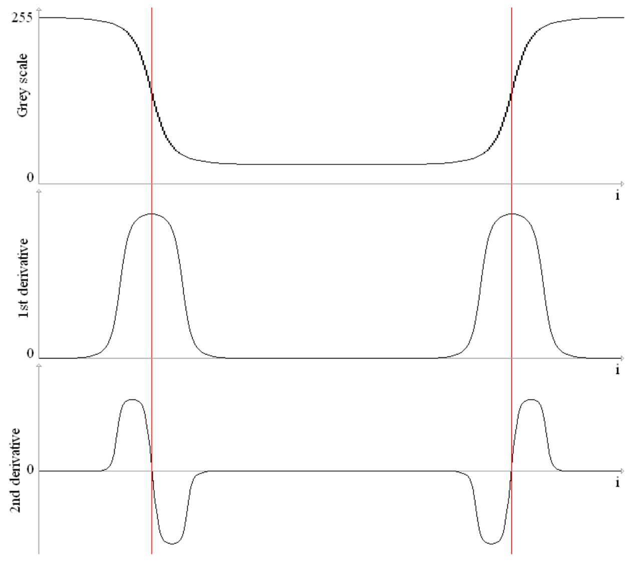
a. Compare and contrast a first derivative edge detector with a second derivative edge detector. **[8 marks]**

b. Using edge based computer vision, for the bottles below, describe how to automatically determine both if the label is straight and if it is torn (e.g. a corner missing). Sample images are shown below. **[17 marks]**



Sample Answer

a. First derivative edge detectors measure the rate of change of the image intensity colour (for grey-scale images), whereas the second derivative edge detectors look at the rate of change of the rate of change of the image intensity. Considering a single row of an image with two significant intensity changes:



First derivatives edge detectors compute the gradient of the edge as a combination (typically Root-Mean-Square) of two orthogonal partial derivatives:

$$\nabla f(i,j) = \sqrt{\left(\frac{\delta f(i,j)}{\delta i}\right)^2 + \left(\frac{\delta f(i,j)}{\delta j}\right)^2}$$

In a similar manner they combine the two orthogonal partial derivatives to also calculate the orientation of the edge:

$$\phi(i,j) = atan2\left(\frac{\delta f(i,j)}{\delta j}, \frac{\delta f(i,j)}{\delta i}\right)$$

There are a large number of first derivative edge detectors, one of the best known of which is Sobel. The partial derivative in a digital image are computed by convolution with two masks $h_1(i,j)$ and $h_3(i,j)$:

$$h_1(i,j) = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \; h_3(i,j) = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

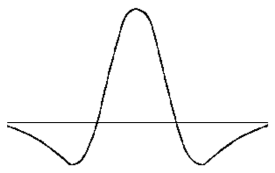
Note that in these filters there is effectively smoothing on each side of the central point.

To locate a first derivative edge we typically threshold the gradient image (although to avoid thick edges we need to apply non-maxima suppression first).

Second derivative edge detectors look at the rate of change of the rate of change and are located by searching for a “zero-crossing” in the resultant derivative image. Second derivative edge detectors are computed in the discrete domain using a single convolution filter the most common second derivative filter is the Laplacian, two discrete approximations of which are:

$$h(i,j) = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad h(i,j) = \begin{bmatrix} 2 & -1 & 2 \\ -1 & -4 & -1 \\ 2 & -1 & 2 \end{bmatrix}$$

In these approximations it is notable that the centre pixel has a significant weighting and this causes problems in the presence of noise. Hence Laplacian filtering is normally preceded by some type of smoothing (typically Gaussian). The combination of these two techniques is the "Laplacian of Gaussian" edge detector which is 1-D looks like



which for obvious reasons is referred to as the Mexican hat.

Similarities:

- Both first and second derivative edge detectors incorporate some level of smoothing
- Both can be used to locate an edge and to determine the gradient (in the second derivative it is the slope of the zero-crossing)

Differences

- The orientation of an edge point can only be determined in the first derivative.
- The second derivative is better for located edges.
- A single convolution is required to compute the Laplacian of Gaussian while two are required for the Sobel operator.
- The Laplacian of Gaussian take a much larger area into account (depending on the value of the sigma of the Gaussian) and this sigma can be varied in order to analyze the image at different scales.

b. To address this problem we assume that we have an image of just one bottle. Then we apply
   - Sobel edge detection (as in part a).
   - Non maxima suppression:
     If the gradient image is thresholded (to identify significant edges) the resulting edges are generally quite wide. Hence at this stage non-maxima suppression is used:

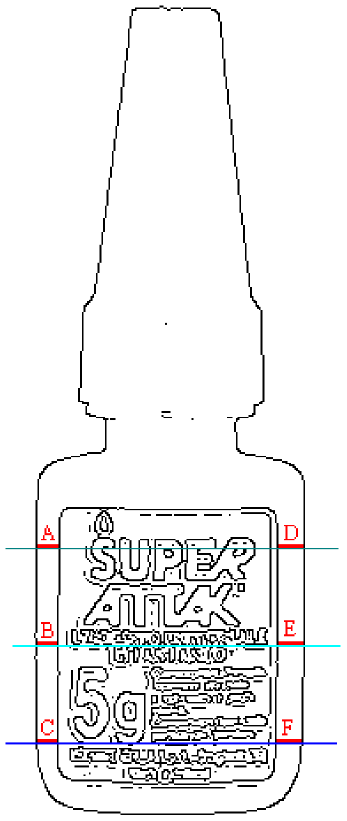
```
Quantize edge orientations
for all points (i,j)
        Look at the 2 points orthogonal to edge
        if  gradient(i,j) < gradient(either of these 2 points)
                output(i,j) = 0
                else output(i,j) = gradient(i,j)
```

   - Thresholding to identify edge points. The output is thresholded, so that maximums less than some threshold are set to 0 and all above the threshold are set to 1 (or typically 255).

```
for every point
  if the edge gradient in any channel >= a threshold value
    set the output to 1 for that point
  else set the output to 0 for that point
```

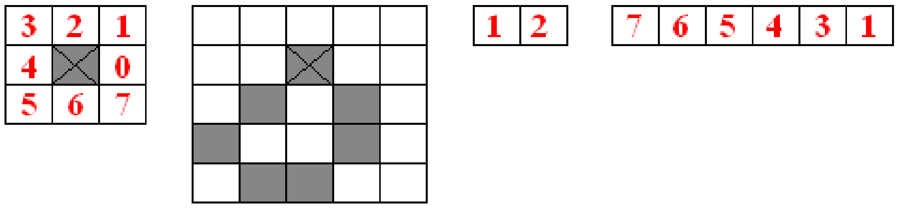
     This gives us a binary scene (each point is either 1 or 0)

   - To identify if the label is straight we consider three different scan lines on the bottle and find the first and second significant edge points from the left and right. The first points represent the bottle whereas the second points represent the label and if we calculate the distances shown on the bottle below then for a straight label:
     $A \approx B \approx C$ and $D \approx E \approx F$



   - To determine if the label is torn we must trace around the label and ensure that the shape is correct and consistent. Based on the column values found for the left and right of the label we can determine the middle column of the label just by averaging these. If we can determine a chain of edge points we can then find the two on the middle line and compare edge points on each side relative to those points (sort of like folding in the middle to see if the two sides match).

- We represent a chain of edge points like these using a boundary chain code (BCC). A boundary chain code consists of a start point and a list of orientations to other connected edge points (i.e. a chain of points). The start point is specified by the (row,column) pair, and each orientation is typically specified simply as a value from 0 to 7 (i.e. the 8 possible directions to a neighbouring pixel). e.g.



- To extract the chain of edge points we can start from any one of the label edge points already located and then use a technique similar to "heuristic search for images borders" to build up the rest of the chain of edge points. Starting at the chosen label point search forwards repeatedly for other edge points (there should be just one) until reaching the start point again. This would not work for all contours but would work for the label in this case.