CS786 Assignment-2

Amrit Singhal (150092)
Department of Computer Science
IIT Kanpur
Kanpur, India
amrits@iitk.ac.in

Aarsh Prakash Agarwal (150004)
Department of Electrical Engineering
IIT Kanpur
Kanpur, India
aarshp@iitk.ac.in

Mrinaal Dogra (150425)
Department of Computer Science
IIT Kanpur
Kanpur, India
mrinaald@iitk.ac.in

1 Problem Statement

The aim of this assignment was to simulate the observations in the domain of visual perception pertaining to the task of identifying odd-one out of given visual cues which can either be feature based or conjunction based search. The observations of this visual search paradigm should confirm with that of Feature Integration Theory of Triesman (1980).

2 Solution

The simulation to above paradigm involved implementing a series to steps to achieve the final goal. The steps are as follows:

2.1 Implementing Complex cells in Visual cortex

The simple cells in visual cortex are known for capturing contrasts and edges. Gabor filters are famous for capturing edges along a certain direction/angle, and thus, compared to simple cells in their functionality. Thus, the aim is to build on the output of the Gabor filter to simulate the functionality of complex cells to detect shapes such as triangles and squares. For simplicity, the triangles are kept equilateral with base parallel to horizontal and squares consistent with their bases also being parallel to horizontal. The filter is run for four different orientations namely, π, $\pi/6$, $5\pi/6$ and $\pi/2$. Note these angles are from the vertical. The responses for triangle would be higher for π, $\pi/6$ and $5\pi/6$ while low for $\pi/2$, as triangular edges are oriented along those directions. Similarly for square, responses along $\pi/2$ and π were higher than the responses along $\pi/6$ and $5\pi/6$. Thus, two complex cells for detecting triangles and squares are respectively calculated as follows:

$$c_1 = \frac{1}{3}\Big(\phi(\pi) + \phi(\pi/6) + \phi(5\pi/6)\Big) - \phi(\pi/2)$$

$$c_2 = \frac{1}{2}\Big(\phi(\pi) + \phi(\pi/2)\Big) - \frac{1}{2}\Big(\phi(\pi/6) + \phi(5\pi/6)\Big)$$

where, $\phi(\theta)$ is the response of Gabor filter along the direction/angle θ. The triangles would have fairly positive response for c_1 and slightly negative response for c_2 and for squares vice versa would be true. Both the complex cells would have little bit of static noise for white background, if none is present, therefore a little positive threshold was needed instead of zero. Therefore, threshold of $\alpha = 1e-6$ was chosen

if $c_1 > \alpha$ .... Triangle is present

if $c_2 > \alpha$ .... Square is present

With above conditions and threshold detection was robust. Figures 1 and 2 show an image for a single shape, and output of the Gabor filters for it.

Figure 1: Image having a single shape

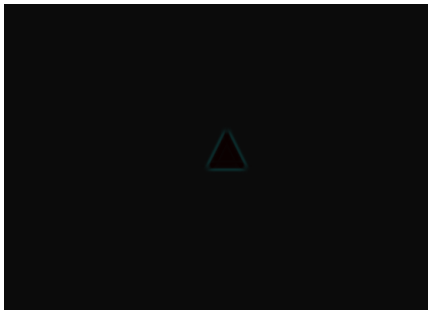

Figure 2: Gabor filters output

2.2 Generate Image Frames for detection

The task was to generate the examples for visual search paradigm given the experiment type and number of objects. Along with shapes, i.e., triangles and squares, we can also vary the color. For our purpose we have chosen the colour as red and blue. Experiment can be two types: conjunction search or feature search. In feature only one of the features is varied, i.e., either color or shape, while in conjunction search both the features are varied, after randomly choosing the odd one out. Also, for image frame, objects are appropriately sized and spaced in a cell, so that none of the objects overlap with each other. Figures 3 and 4 show an image randomly generated for feature search and conjunction search respectively. As clearly visible, the filters successfully detect the edges in the image.

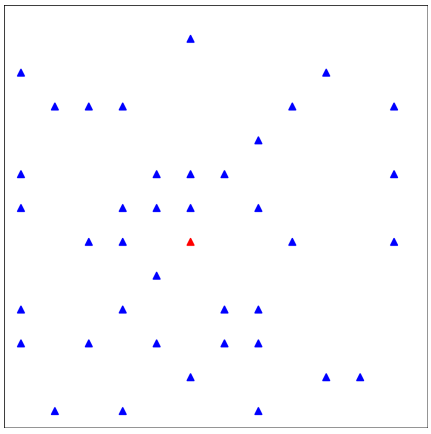

Figure 3: Feature Search image with 40 objects

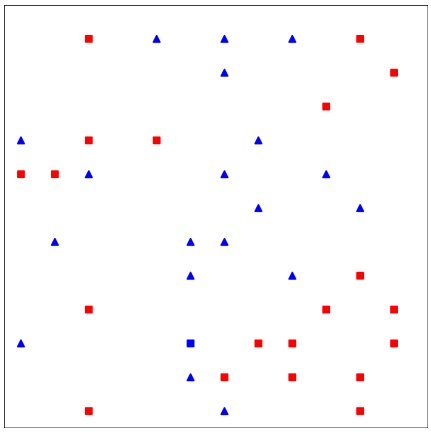

Figure 4: Conjunction Search image with 40 objects

2.3 Find feature map for each feature and implement the search

After generating image frames, the next task would be to create a feature map for all the features. We have in total four features, `red, blue, triangle and square`. All of these could be present in an image frame among different objects in combinations of color and shape. For construction of all these feature maps, only a cell is chosen at once and tested for all these features. The whole image can thought of as the matrix and center of each cell would represent that cell. Since, during generation we have taken care, the given a frame size of cell, only one of the objects, i.e., *red-square, blue-square, red-triangle, blue-triangle* would be present in the cell.

For the feature maps of particular shape, run the Gabor filter from the complex cells to detect it's presence and if the shape is present, store 1 for the center coordinate as a representative of that cell for that feature map. And, thus the feature map for triangles and squares are constructed.

For constructing the features maps of color, since the object is appropriately sized, check for the pixel value of the all the three colors at the center of the cell. E.g., for blue coloured objects value of r and g would be low and b would be high. Again store 1 for the center of the cell to capture the respective feature map matrix.

After constructing feature maps, we basically would have four matrices, each for a feature, with 1's and 0's. 1's represents the feature was present in that cell, while 0's represent it's absence.

For feature search, count the number of 1's in each matrix and the matrix with count 1 is odd one out and the position of the location of that 1 from the matrix is obtained and feature search is complete. To simulate the pop out-effect, we convert the feature map matrices to sparse matrices which enable us to count in constant time rather than $O(m^2)$ time, where m is the dimension of the square matrix. See figure 5 where red triangle is detected out of blue ones using feature search.

For conjunction search, for we will have to iterate over two matrices in nested fashion to obtain the count of combination of features. This basically would give the count for `red-square, blue-square, red-triangle, blue-triangle`. Again, the combination with the count 1 would be the odd one out. This operation again, because of the sparse matrices, and nested operations would become costlier and increase with the number of objects. See figure 5 where blue-square is detected out of red-squares and blue-triangles using conjunction search.

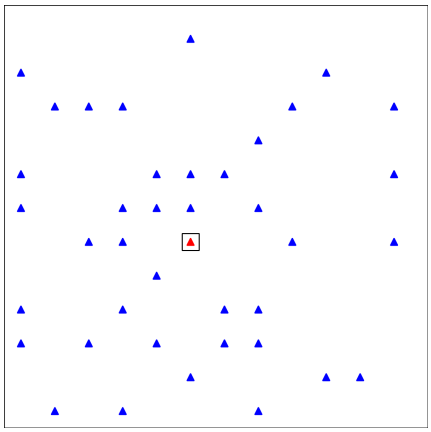

Figure 5: Feature Search result

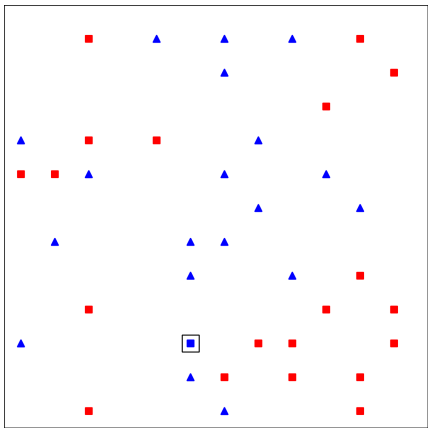

Figure 6: Conjunction Search result

Simulating Response times for the searches

To enhance the comparison we added a simulated delay in case of conjunction search, depending on the number of objects, and added a constant delay to feature search to match the base case in two search paradigms, i.e., $N = 1$. Conjunction search and feature search paradigms were run for multiple values of N and the runtime for each search, along with the simulated delay, was plotted against the number of objects. The plot obtained can be seen in figure 7 and is clearly in agreement with the observation of Feature Integration Theory (FIT) experiments.

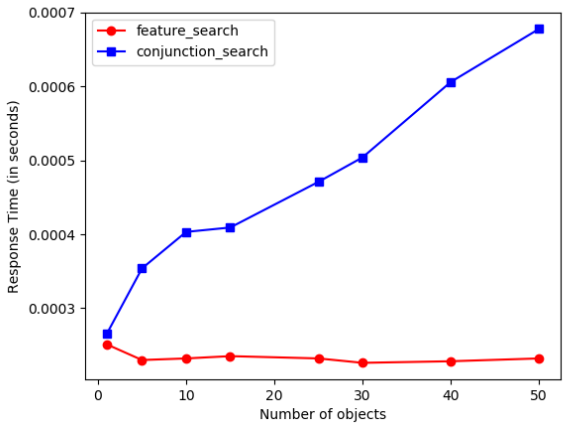

Figure 7: Response time vs Number of objects