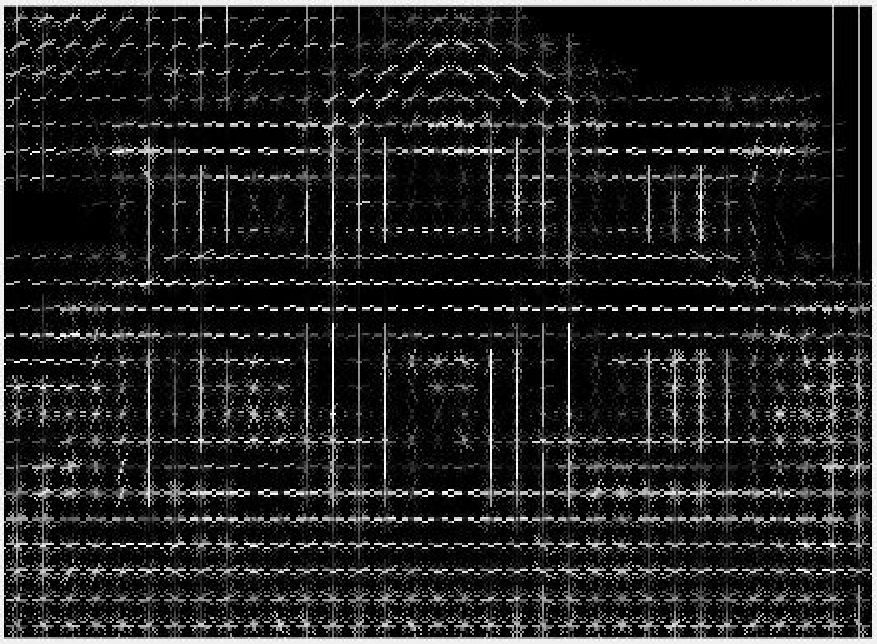
DIP ASSIGNMENT-5
REPORT

-PRANAY GUPTA
-20161088

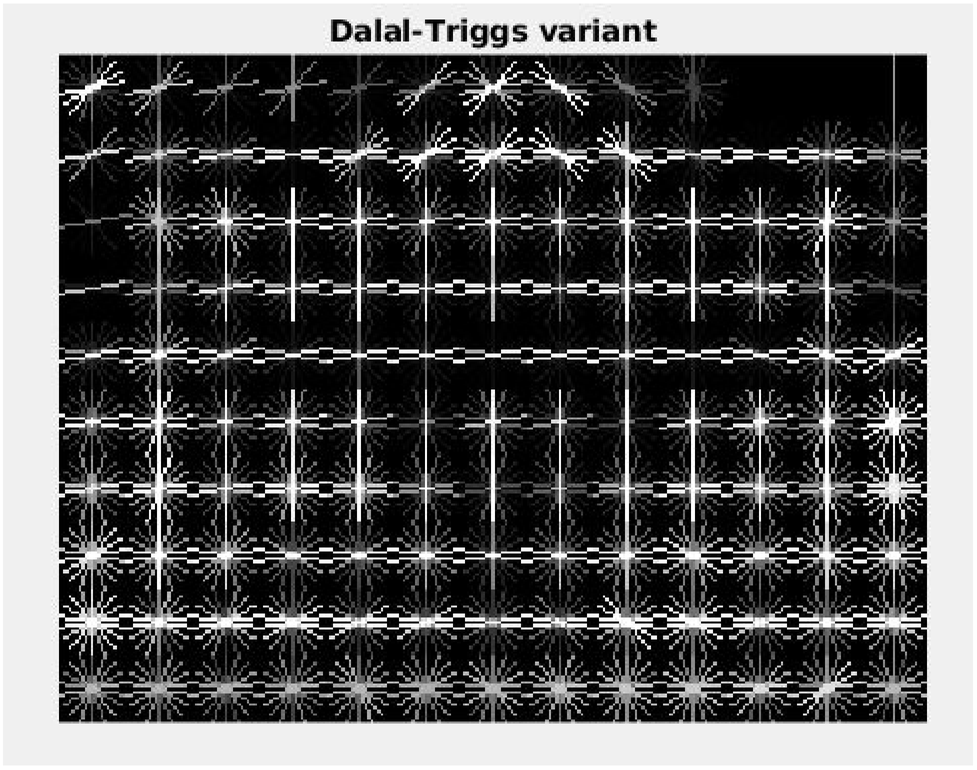
HOG Features:

The hog features operations are performed using the function vl_hog defined in vlfeat. **vl_hog(im, cell_size)** computes the HOG features for image im and the specified cell_size. im can be either grayscale or colour in SINGLE storage class. It returns an array of cells: its number of columns is approximately the number of columns of im divided by cell_size and the same for the number of rows. The third dimension spans the feature components.

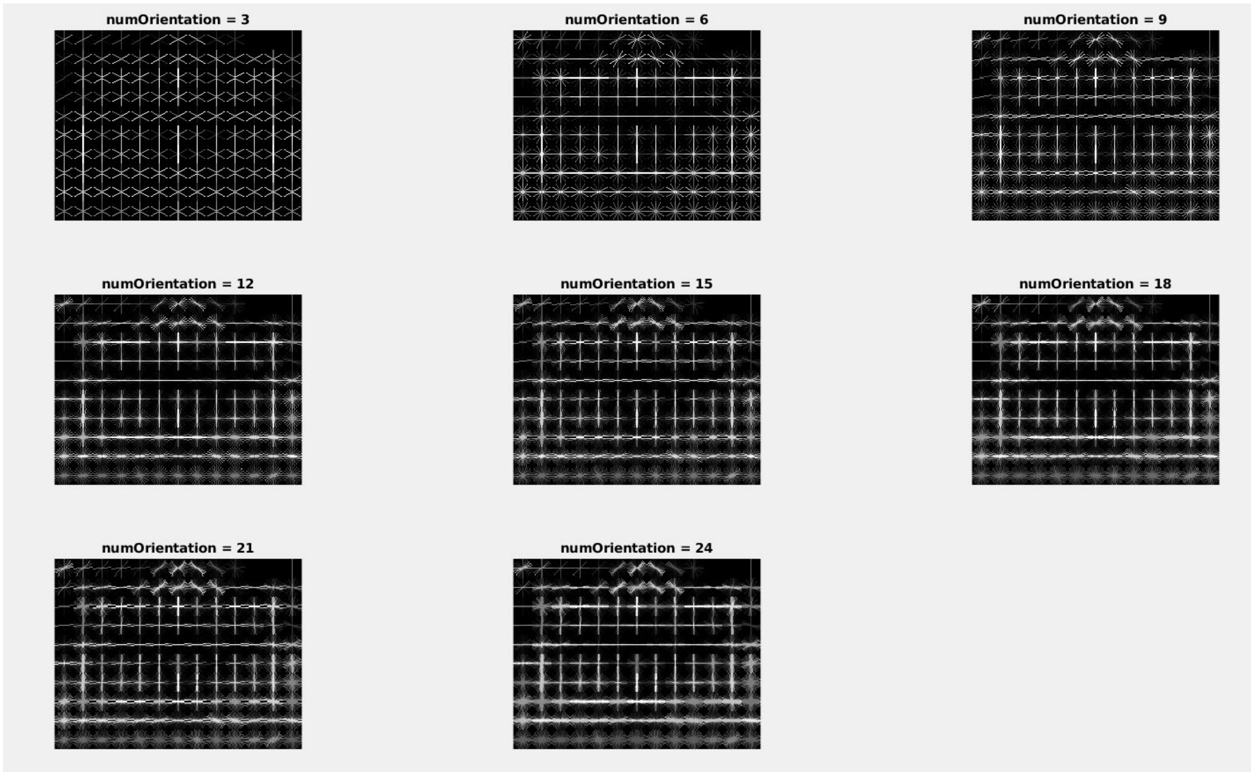

Standard HOG features with a cell size of 8 pixels.

The same function can also be used to generate a pictorial rendition of the features, although this unavoidably destroys some of the information contained in the feature itself. **IMAGE = vl_hog('render', HOG)** returns an IMAGE containing an iconic representation of the array of cells HOG. The above image is produced using the render feature in vl_hog.

HOG exists in many variants. VLFeat supports two: the UoCTTI variant (used by default) and the original Dalal-Triggs variant (with 2×2 square HOG blocks for normalization). The main difference is that the UoCTTI variant computes bot directed and undirected gradients as well as a four dimensional texture-energy feature, but projects the result down to 31 dimensions. Dalal-Triggs works instead with undirected gradients only and does not do any compression, for a total of 36 dimension. The Dalal-Triggs variant can be computed as
hog = vl_hog(im, cell_size, 'variant', 'dalaltriggs')

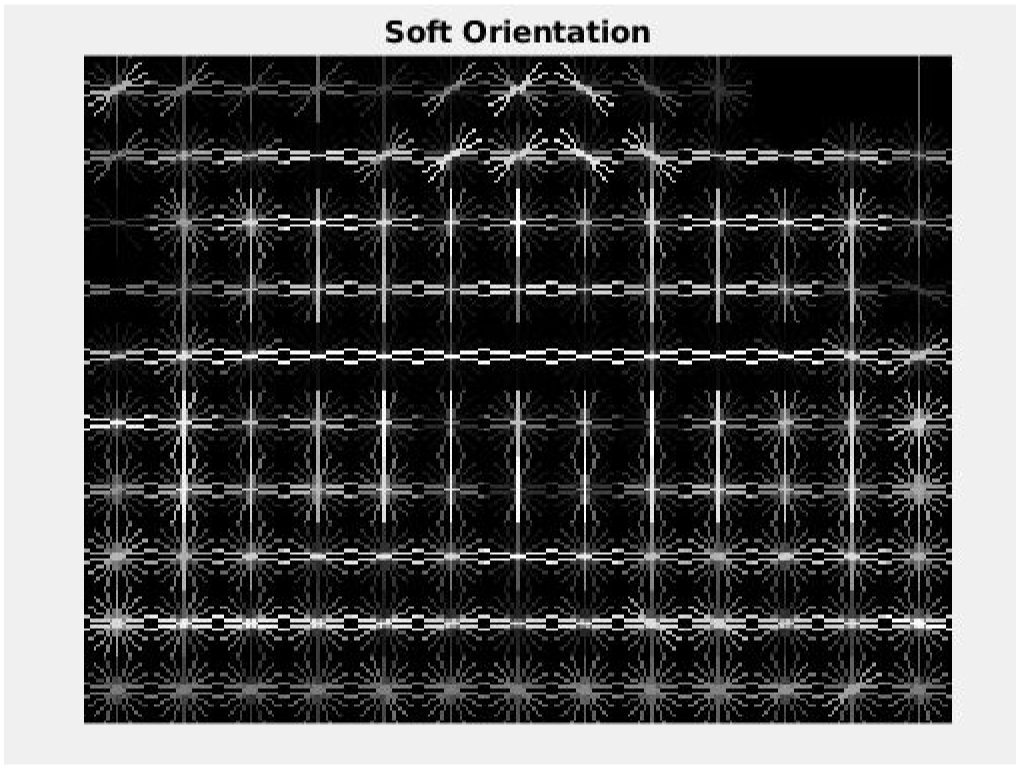


One can specify the number of orientations in the histograms by the numOrientations option.It allows us to choose a number of undirected orientations in the orientation histograms. The angle [0, pi) is divided in to numOrientations equal parts.
hog = vl_hog(im, cell_size, 'numOrientations', o)



Another useful option is BilinearOrientations. This flags activates the use of bilinear interpolation to assign orientations to bins. This produces a smoother feature, but is not some other implementations (e.g. UoCTTI).

hog = vl_hog(im,cell_size,'numOrientations', 4) ;



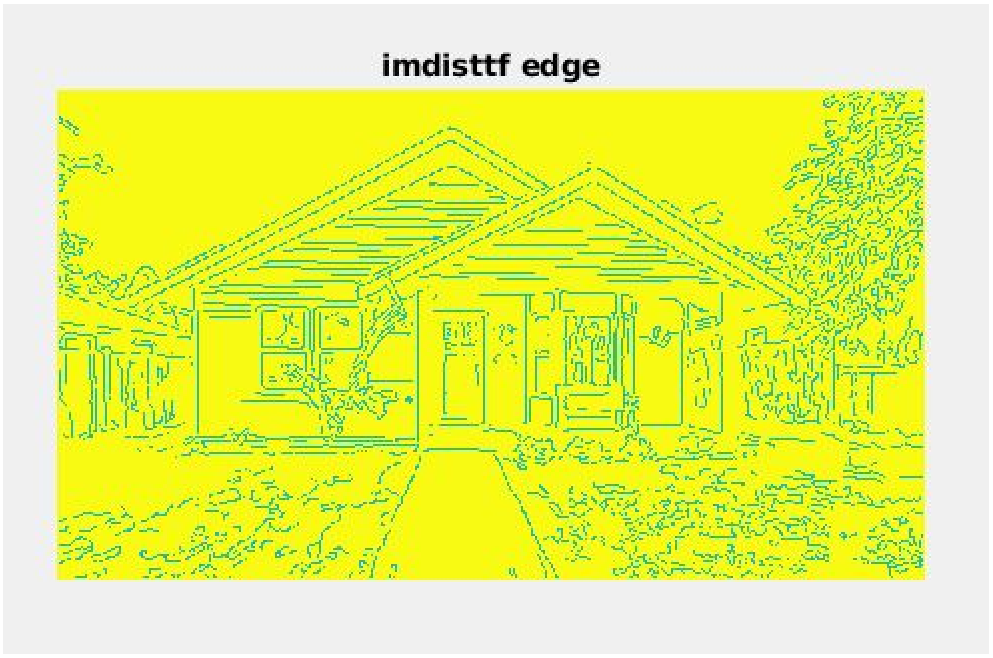
Distance Transform:

The distance transform of an image image is defined as

dt(u,v) = min image(u',v') + alpha (u'-u-u0)^2 + beta (v'-v'-v0)^2
U'v'

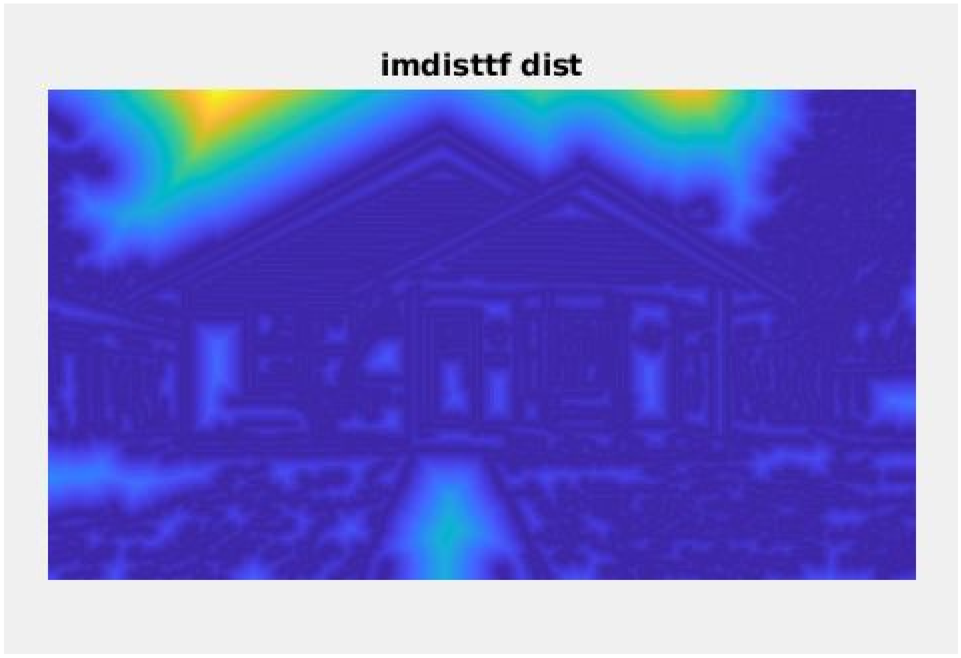
We have used the image distance transform is used to compute the distance of each image pixel to the nearest element in an edge map, obtained from the Canny's edge detector

edges(edge(rgb2gray(im), 'canny')) = 0 ;



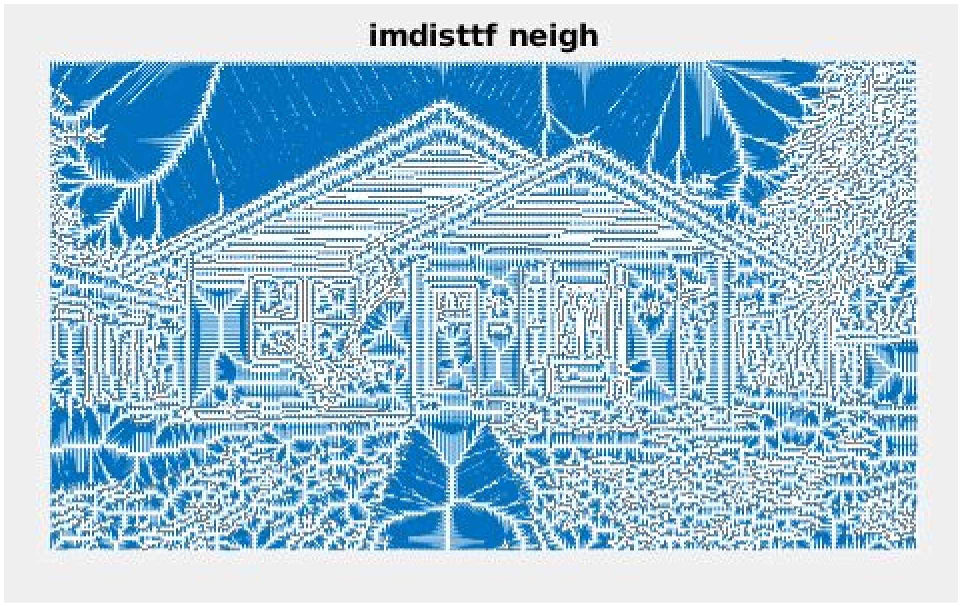
[distanceTransform, neighbors] = vl_imdisttf(single(edges)) ;

[DT, INDEXES] = vl_imdisttf(I) computes the distance transform of image I. It returns a matrix INDEXES that contains for each pixel (x,y) the index of the pixel (u,v) which is the minimizer of the distance transform objective.



The distance sqrt(distanceTransform) to the closest edge element

The matrix neighbors contains for each pixel (u,v) the index of the pixel (u',v') where the maximum is attained in the definition of the distance transform. This allows to associate to know for each pixel which is the nearest edge element, not just its distance, as exemplified by the following figure:


imdisttf neigh