2B-L3 Generalized Hough transform - Then and now

Week4

1. Intro
   (a) this part has two significant differences as before
      i. Non-analytic models
         A. Parameters express variation in pose or scale of fixed but arbitrary shape (that was then)
         B. not as lines, circles, etc. analytic model
      ii. Visual code-word based features
         A. Not edges but detected templates learned from models (this is "now")
         B. edges is the old computer vision interesting point and features about these points is the new CV
2. Generalized Hough Transform
   (a) if we have specific shape of the obj, we have the equation and then it's easy for each pixel to vote.
   (b) but now what if the shape is arbitrary, how to vote? One approach - Hough table /R table 1980s
      i. Creation of R table
         A. At each boundary point, compute displacement vector: r = c − pi, where c is the reference point, indicating the object
         B. Measure the gradient angle θ at the boundary point.
         C. Store that displacement in a table indexed by θ.
      ii. Recognition
         A. At each boundary point, measure the gradient angle θ
         B. assuming that the orientation of the object doesn't change.
         C. the gradient is computed w.r.t. the coordinate rather than the reference point
         D. Look up all displacements in θ displacement table.

        E. Vote for a center at each displacement, i.e. vote for all the displacement indexed by this θ. In the end, the correct reference point gathers a whole bunch of votes.

    iii. key idea

        A. this R-table or Hough table is to tell each point in the new test image how to vote.

3. Generalized Hough Transform Example

    (a) assume that we know which way is inside to the edge

    (b) look at the bottom line

        i. creation of the R table, all the displacement for the same theta

        ii. recognition: vote all the displacement indexed by this theta

        iii. after doing this for two thetas, we can get the reference point

4. Generalized Hough Transform Algorithm

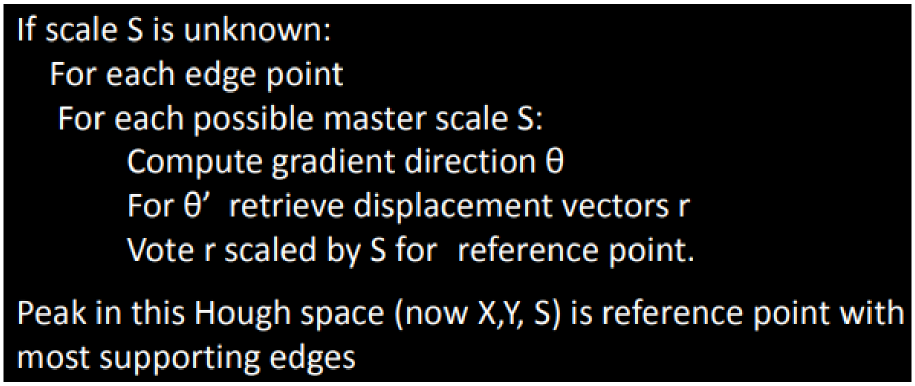
If orientation is known:

1. For each edge point
    - Compute gradient direction θ
    - Retrieve displacement vectors r to vote for reference point.
2. Peak in this Hough space (X,Y) is reference point with most supporting edges

If orientation is unknown:

For each edge point
- For each possible master θ*
    - Compute gradient direction θ
    - New θ' = θ - θ*
    - For θ' retrieve displacement vectors r to vote for reference point.

Peak in this Hough space (now X,Y, θ*) is reference point with most supporting edges



Complexity

(a) for case 1: know orientation scale, quadratic

(b) for case 2 and 3: one is unknown: cubic

(c) for case 4: both unknown: biquadrate

5. Application in Recognition

   (a) locating the obj making use of Hough table/displacement vector

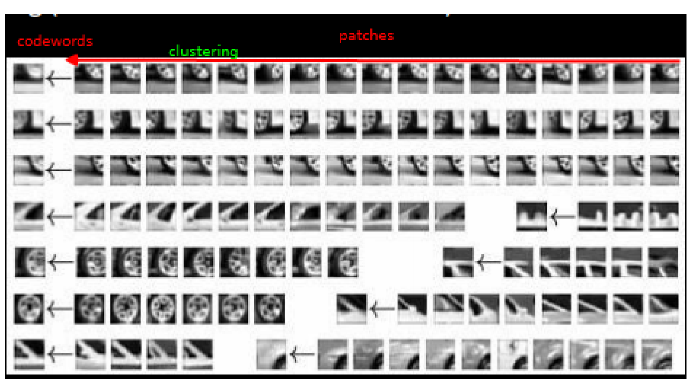
   (b) instead of the gradients of edge pixels, we use little feature patches

   (c) these features are referred to as visual codeword, which are used to index the displacements

   (d) so we can build a R table based on these codewords/features

6. Training procedure to develop what's called visual code-words

   (a) Build codebook of patches around extracted interest points using clustering



i. procedure

   A. take your interest point operator to pull out all the interest points on a bunch of training images.PS: interest point: points in the image where reasonable amounts of interesting stuff is happening.

   B. collect the little image patch right around those points

   C. and then use some algorithm to cluster them [the patches]

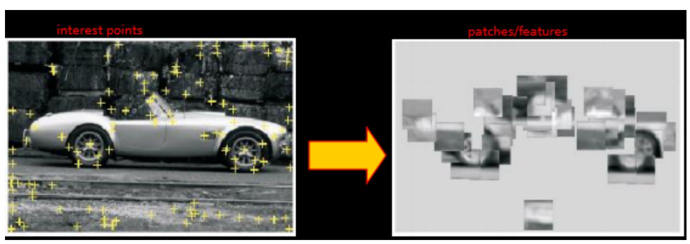
   D. the centers of those clusters are referred to, as visual code words.

ii. misc

   A. the code words are the features we're gonna looking at in the images

   B. all of this is done automatically so you may see some things that look a little strange in terms as code-words.

(b) Map the patch around each interest point to closest codebook entry



i. take these code words to find everywhere that the code words landed in the image.

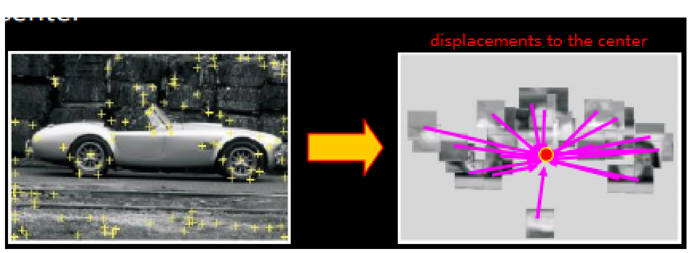
ii. So what we have here is all of these little interest points.

iii. for every interest point, we find the feature that seems to look best at that point.

   A. So that becomes the label of that point.

   B. e.g. here the label is the bottom right-hand corner of a tire

(c) Training: Displacements



i. For each codebook entry, store all displacements relative to object center

ii. find the displacement vector to the center for these little features

iii. write down that displacement vector in a table that's indexed by a patch label.

7. End

   Hough forest

   (a) combine the Hough offset ideas that features vote for offsets with an ensemble or a collection of classifiers

   (b) similar to random forests, so called hough forest