IIST, Thiruvananthapuram | AVD624 - Computer Vision

Lecture Notes: Weeks - 6 to 10

03/03/2022 - 31/03/2022

Prof. Dr. Deepak Mishra | *Summarised by: Vatsalya Gupta*

This is a weekly summary of the lectures given by *Dr. Deepak Mishra*, Professor and Head of Department of Avionics, for the course on *Computer Vision* conducted during the even semester 2021-22.

1 Hough Transform

1.1 Fitting lines

1.2 Fitting circles

2 RANSAC: Robust Estimation

RANdom SAmple Consensus (RANSAC) is used wherever we like to fit a model and we expect there are outliers possible. We can repeat the following steps until the best model is found with high confidence.

1. *Sample* the number of data points required to fit the model.
2. *Compute* model parameters using the sampled data points.
3. *Score* by the fraction of inliers within a preset threshold of the model.

Least squares estimation is sensitive to outliers, so that a few outliers can greatly skew the result. The fit procedure implicitly assumes there is only one instance of the model in the data. Robust estimation fits model to inliers while ignoring outliers.

2.1 Choosing the number of samples

We need to solve the following for *N*.

$$1 - (1 - (1 - e)^s)^N = p$$

e = probability that a point is an outlier
s = number of points in a sample
N = number of samples (we want to compute this)
p = desired probability that we get a good sample

$(1 - e)$ = probability that choosing one point yields an inlier
$(1 - e)^s$ = probability of choosing s inliers in a row (sample only contains inliers)
$(1 - (1 - e)^s)$ = probability that one or more points in the sample were outliers (sample is contaminated)
$(1 - (1 - e)^s)^N$ = probability that N samples were contaminated

$(1-(1-(1-e)^s)^N)$ = probability that at least one sample was not contaminated (at least one sample of *s* points is composed of only inliers)

Early termination: Terminate when inlier ratio reaches expected ratio of inliers. Choose T so that, with probability *p*, at least one random sample set is free from outliers.

$$T = (1-e) \times (total\ number\ of\ data\ points)$$

2.2 Pros of using RANSAC and Robust estimation

- Robustly deal with outliers
- Works well for 1 to roughly 10 parameters (depending on the number of outliers)
- Easy to implement and understand

2.3 Cons of using RANSAC and Robust estimation

- Computational time grows quickly with fraction of outliers and number of parameters needed to fit the model
- Not good for getting multiple fits

3 Scale-Invariant Feature Transform (SIFT)

There are mainly four steps involved in the SIFT algorithm, which can be followed by **Keypoint Matching**.

- **Scale-space peak selection:** Potential location for finding features
- **Keypoint Localisation:** Accurately locating the feature keypoints
- **Orientation Assignment:** Assigning orientation to keypoints
- **Keypoint descriptor:** Describing the keypoints as a high dimensional vector

3.1 Scale-space peak Selection

Real world objects are meaningful only at a certain scale. The scale space of an image is a function $L(x, y, \sigma)$ that is produced from the convolution of a Gaussian kernel (blurring) at different scales with the input image. Scale-space is separated into octaves and the number of octaves and scale depends on the size of the original image. So we generate several octaves of the original image (each of image size half the previous one).

3.1.1 Blurring

$$L(x, y, \sigma) = I(x, y) * G(x, y, \sigma) = I(x, y) * \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}$$

G is the Gaussian Blur operator and *I* is an image. While x, y are the location coordinates and σ is the *scale* parameter, the amount of blur, greater the value, greater the blur.

3.1.2 Difference of Gaussian (DoG) kernel

Now we use those blurred images to generate another set of images, the Difference of Gaussians (DoG). The difference of Gaussian is obtained as the difference of Gaussian blurring of an image with two different σ, let it be σ and $k\sigma$. This process is done for different octaves of the image in the Gaussian pyramid. Difference of Gaussians are then used to calculate Laplacian of Gaussian (LoG) approximations that are scale invariant.

3.1.3 Finding keypoints

One pixel in an image is compared with its 8 neighbours as well as 9 pixels in the next scale and 9 pixels in previous scales. This way, a total of 26 checks are made. If it is a local extrema, it is a potential keypoint. This means that keypoint is best represented in that scale.

3.2 Keypoint Localisation

Some of the keypoints generated in the previous step lie along an edge, or they don't have enough contrast. In both cases, they are not as useful as features. DoG has a higher response for edges, so edges also need to be removed. The approach is similar to Harris Corner Detector for removing edge features. We use a 2x2 Hessian matrix (H) to compute the principal curvature.

- Reject flats:
  - $|D(\hat{\mathbf{x}})|$ < 0.03
- Reject edges:

$$\mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}$$

Let α be the eigenvalue with larger magnitude and β the smaller.

$$\mathrm{Tr}(\mathbf{H}) = D_{xx} + D_{yy} = \alpha + \beta,$$

$$\mathrm{Det}(\mathbf{H}) = D_{xx}D_{yy} - (D_{xy})^2 = \alpha\beta.$$

Let r = α/β. So α = rβ

$$\frac{\mathrm{Tr}(\mathbf{H})^2}{\mathrm{Det}(\mathbf{H})} = \frac{(\alpha+\beta)^2}{\alpha\beta} = \frac{(r\beta+\beta)^2}{r\beta^2} = \frac{(r+1)^2}{r},$$

(r+1)²/r is at a min when the 2 eigenvalues are equal.

  - r < 10

3.3 Orientation Assignment

The next step is to assign an orientation to each keypoint to make it rotation invariant. A neighbourhood is taken around the keypoint location depending on the scale, and the gradient magnitude and direction is calculated in that region. An orientation histogram with 36 bins covering 360 degrees is created. The highest peak in the histogram is taken and any peak above 80% of it is also considered to calculate the orientation. It creates keypoints with same location and scale, but different directions.

3.4 Keypoint descriptor

A 16x16 window around the keypoint is taken. It is divided into 16 sub-blocks of 4x4 size. For each sub-block, 8 bin orientation histogram is created. 4x4x8 directions give 128 bin values. It is represented as a

feature vector to form keypoint descriptor.

Rotation independence: The keypoints rotation is subtracted from each orientation. Thus each gradient orientation is relative to the keypoints orientation.

Illumination independence: We can achieve illumination independence by thresholding big numbers. So, any number (of the 128) greater than 0.2 is changed to 0.2. This resultant feature vector is normalised.

3.5 Keypoint Matching

Keypoints between two images are matched by identifying their nearest neighbours. But in some cases, the second closest-match may be very near to the first. In that case, the ratio of closest-distance to second-closest distance is taken. If it is greater than 0.8, they are rejected. It eliminates around 90% of false matches while discards only 5% correct matches.

Some major *advantages* of SIFT include locality, distinctiveness, quantity, efficiency and extensibility.

4 Histogram of Oriented Gradients (HOG)

The HOG descriptor focuses on the structure or the shape of an object. HOG is able to provide the edge direction as well. This is done by extracting the gradient and orientation of the edges. The HOG feature descriptor counts the occurrences of gradient orientation in localised portions of an image.

1. **Preprocess the Data (64x128):** We need to preprocess the image and make the width to height ratio as 1:2. The image size should preferably be 64x128. This is because we will be dividing the image into 8x8 and 16x16 patches to extract the features.
2. **Calculating Gradients (direction *x* and *y*):** Calculate the gradient for every pixel in the image. This is similar to using a Sobel Kernel of size 1.
3. **Calculate the Magnitude and Orientation:** Calculate the total gradient (or magnitude) as $\sqrt{G_x^2 + G_y^2}$ and the orientation (or direction) as $\phi = \tan^{-1}(G_y/G_x)$.
4. **Calculate Histogram of Gradients in 8x8 cells (9x1):** The image is divided into 8x8 cells, and the histogram of oriented gradients is computed for each cell. We will get a 9x1 matrix for each cell.
5. **Normalise gradients in 16x16 cell (36x1):** We can reduce the lighting variation by normalising the gradients. We will combine four 8x8 cells to create a 16x16 block. So, we would have four 9x1 matrices or a single 36x1 matrix. To normalise, divide each value by the root of the sum of squares.
6. **Features for the complete image:** We will combine all the features for the 16x16 blocks to get the features for the final image. We would have 105 (7x15) blocks of 16x16. Each of the 105 blocks has a feature vector of 36x1. So, the total features for the image would be 105 x 36x1 = **3780 features**.

5 Bag of Visual Words (BoVW)

The general idea of bag of visual words is to represent an image as a set of features and their descriptors.

5.1 Creating the Dictionary or Codebook

We use techniques like SIFT, BRISK, etc. These feature detectors return an array containing the descriptors. We do this for every image in our training dataset. We have *N* (number of images in training dataset) arrays. Now we will use clustering algorithm like K-Means to form *K* clusters. K-Means will return the centre of each group. Each cluster centre (centroid) acts as a visual word. All these *K* centroids form our codebook.

5.2 Creating the Histogram

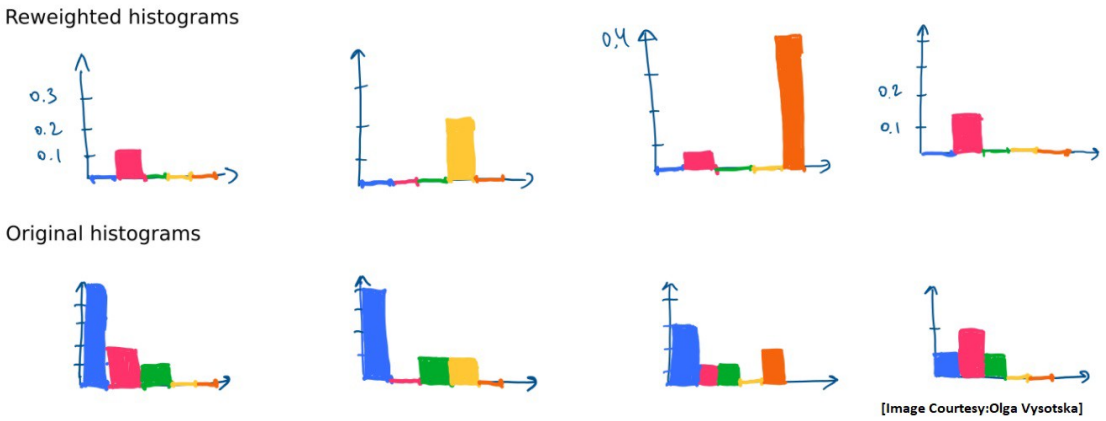
We iterate through our images and look for words in the image present in our dictionary. Once we detect a word present in both dictionary and image, increase the count of the particular word, i.e. array[*i*][*w*]+=1 where *i* is the current image and *w* is the word. This way we create the histograms for the images.

5.3 Classification and image Similarity

We can train a model (Random Forest, LinearSVC) for classifying the images. The images in the test dataset are pre-processed in the same manner and then fed to the model to predict the image category. There is, however, a problem with this approach. It occurs when the visual words occur in a lot or every image of the image database. These words make the classifying task harder. To solve this problem, we can apply *TF-IDF* (Term Frequency - Inverse Document Frequency) approach. It reweights every bin of a histogram and will downweight the 'uninformative' words (i.e. features that occur in a lot of images/everywhere) and enhance the importance of rare words.

$$t_{id} = tf \times idf = \left(\frac{n_{id}}{n_d}\right)\left(\log\frac{N}{n_i}\right)$$

t_{id} = histogram bin of word *i* for image *d*,
n_{id} = occurrences of word *i* in image *d*,
n_i = number of images that contain word *i*,
n_d = total number of words in image *d*,
N = total number of images in the database.



In the above image, we can see how the blue word gets almost zero weight in the reweighted histograms. BoVW approach uses cosine distance for comparison of 2 histograms. Images have 0 distance to themselves.

$$d_{cos}(x,y) = 1 - cossim(x,y) = 1 - \frac{x^T y}{||x||\,||y||} \quad , \quad d_{cos}(x,y) \in [0,1]$$

6 Image Segmentation

6.1 Image binarisation

6.1.1 Otsu's thresholding

6.2 Region based segmentation

6.2.1 Region growing

6.2.2 Region splitting and merging

6.3 Clustering based segmentation

6.4 Mean shift segmentation

6.4.1 Density estimation

6.5 Simple Linear Iterative Clustering (SLIC)

Super pixel based image segmentation

7 Optical Flow

The basic problem definition of optical flow is that given two consecutive image frames, estimate the motion of each pixel. We consider two assumptions for this - **brightness constancy** and **small motion**. A video can be thought of as a spacetime cube and optical flow helps in estimating the properties of this cube.

7.1 Lucas-Kanade method

7.2 Horn-Schunck method

7.3 Large motion: Multi-resolution approach