Computer Vision

CS-512 - Assignment 5

MEGHA TATTI
CWID: A20427027
Course no: CS-512
Fall 2018

1. Robust estimation and segmentation.

(a) Explain what are outliers, and is the fundamentals problem associated with them when fitting a model.

⇒ Outliers are ~~points~~ noise points that is distant from other points. The problem associated with ~~outl~~ outliers is that when the model is fit considering the outliers, it results in a wrong solution.

b) The objective function used for robust estimation is as follows:-

$$E(\theta) = \sum_{i=1}^{n} \rho_\sigma(d(x_i, \theta))$$

In robust estimation :- $\rho_\sigma(x) = \frac{x^2}{x^2+\sigma^2}$.

In standard least square objective function ie,

$$E(\theta) = \sum_{i=1}^{n} d(x_i, \theta)^2;$$

the outliers will have higher value and influence model more. However, in robust estimate $\rho_\sigma(x) = \frac{x^2}{x^2+\sigma^2}$ will ~~lowe~~ lower the influence of the outlier.

c) Geman-McClure function for robust estimation is

$\rho_\sigma(x) = \frac{x^2}{x^2+\sigma^2}$, where $x >> \sigma$; $\rho_\sigma = 1$

$x << \sigma$; $\rho_\sigma = \frac{x^2}{\sigma^2}$

The advantage of this function is that it is not affected by ~~outl~~ outliers. Using this function, the maximum weight the outliers can get is 1, where as ~~the~~ in the standard least square function the weight given to outliers is x^2.

Bandwidth parameter σ can be adjusted in an iterative manner using below steps:-

- drawing a large subset of points uniformly at random.
- Fit model using robust estimation, given θ_n
- Compute $\sigma_n = 1.5 * \text{median}(d(x_i, \theta_n))$
- Repeat the process while $(\theta_n - \theta_{n-1}) >$ Threshold.

We start with a large σ ie, $\sigma_n = 1.5 * \text{median}(d_i(x_i, \theta_n))$ and as we fit better and better model, the median of the points decreases and in turn σ decreases as we estimate σ is $1.5 * \text{median}(d(x_i, \theta))$.

---

d) Principle of the RANSAC algorithm is to use maximum number of points to fit the model and repeat this process many times and choose the best model after many trials.

Try k times, choose distance to get best model.

(i) ~~repeat~~ repeat k times.
↳ draw n points uniformly at random replacement.
↳ fit model to point
↳ find all inliers
↳ if there are d inliers, recompute model

(ii) choose best solution.
↳ The number of points drawn at each attempt should be small because there are less chances of getting outliers and atleast in one of my many traids will lead to a better model.

---

e) Parameters of RANSAC algorithm:-

$n \rightarrow$ number of points to draw at each evaluation.

$d \rightarrow$ ~~min~~ minimum number of points needed.

$k \rightarrow$ number of ~~tra~~ trials/evaluations

$t \rightarrow$ distance to identify outliers.

Formula for estimating the number of trails, k:

$$k = \frac{\log(1-p)}{\log(1-w^n)}$$

where $p =$ ~~prop~~ probability that atleast one of the trials will succeed.

$w =$ probability that a point is an inlier.

$n =$ number of points to draw at each trial.

we update $w = \frac{\text{Number of inliers}}{\text{Number of points}}$.

$$(1-p) = (1-w^n)^k \Rightarrow \log(1-p) = k\log(1-w^n) \Rightarrow$$

$$k = \frac{\log(1-p)}{\log(1-w^n)}$$

f) Segmentation is nothing but to separate foreground from background.

In merge (agglomerative) approach, we start with each pixel in a different cluster and merge iteratively based on the distance or similarity of the feature vectors. By merging similar pixels together, we make the clusters denser.

In Split approach, we start with having all pixels in a single cluster and iteratively split the cluster by looking at the distance/similarity of (pixels) feature vectors. We decrease the size of clusters by removing the pixels which do not belong to a particular cluster.

9) K-means algorithm.

- Select k
- start with initial guess of k-means
- repeat until stopping criteria is met ie, mean donot change.
  - for each pixel, assign the pixel to the cluster nearest to it.

$$l_i = \underset{j \in (1,k)}{\operatorname{argmin}} \| f_i - m_j \|^2 ;$$

$f_i \Rightarrow$ feature vector of i^{th} pixel.

$m_j \Rightarrow$ means of j^{th} ~~the~~ cluster.

  - calculate the new mean of the cluster as

$$m_j = \frac{\sum_{i \in S_j} f_i}{\text{number of pixels in } S_j} ; \quad S_j \text{ is all the pixels labelled } l_j.$$

Mixture of gaussian algorithm for segmentation.

The process in mixture of ~~gau~~ gaussian is same as that of k-means.

The difference is in the distance measure used to assign pixels to the cluster centers.

Instead of using $d = \|f_i - m_j\|^2$ as the evaluation of distance, It uses $d = (f_i - m_j)^T \Sigma_j^{-1} (f_i - m_j)$ where Σ_j is the covariance ~~mat~~ matrix. and

$$\Sigma_j = \frac{\sum_{i \in S_j} (f_i - m_j)(f_i - m_j)^T}{\text{number of pixels in } S_j} ; \quad m_j = \frac{\sum_{i \in S_j} f_i}{\# S_j}.$$

~~h) Meanshift algorithm for segmentation.~~
~~Similar to k means, instead of $m_j = \frac{\sum_{i \in S_j} w(f_i - m_j) f_i}{\sum_{i \in S_j} w(f_i - m_j)}$~~

---

h) Meanshift algorithm for segmentation.

It is similar to k-means. The difference is in calculating the mean of cluster:

$$m_j = \frac{\sum_{i \in S_j} w(f_i - m_j) f_i}{\sum_{i \in S_j} w(f_i - m_j)} \; ; \; w(f_i - m_j) = \text{exponent}(-\|f_i - m_j\|).$$

when recomputing the mean of clusters, we give weight for each pixels belonging to the cluster based on its distance to the previous mean of the cluster.

- the pixels closer to the mean are weighted higher than the ones farther. The closer the pixel/point is to the mean, the more it should affect the mean than the ones farther.
- It finds cluster centers as peaks of histograms.

2a) Given projection equation = $p = MP$.

forward projection :- Given the coordinates of the object in the world (3D) and projection matrix M forward projection is to find the image coordinates of the 3D object.

Camera Calibration :- This finds the camera parameters (internal and external) used in the projection given the image & world coordinates of the object.

Reconstruction :- Given the image coordinates of object, p, and the projection matrix M, find the world coordinates (3 of the object.

forward projection is the easiest as there is no ambigious decision to make. ie, each point in 3D corresponds to a single point in 2D.
Reconstruction is the difficult one as we need to add the information we already lost from going to 2D from 3D and each point in 2D can represent a line in 3D, which makes it ambigious.

2b) Necessary input for camera calibration . . .

$$\{P_i\}_{i=1}^{n} \longleftrightarrow \{p_i\}_{i=1}^{n}$$

$\{x_i, y_i\}_{i=1}^{n} \longleftrightarrow \{X_i, Y_i, Z_i\}_{i=1}^{n}$ for camera calibration, we need the corresponding points in both 2D and 3D.

2c) Steps in the non-coplanar calibration algorithm.

(i) given image points (p) and world points (P), estimate the (3×4) projection matrix M, using $p = MP$.

(ii) Find the camera parameters, internal (K^*) and external (R^* and T^*) using the estimated projection matrix in step 1, as we know that $M = K^*[R^* | T^*]$

2d) $M = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 1 & 0 & 3 & 4 \\ 1 & 1 & 1 & 1 \end{bmatrix}$ $P_i = \underset{3D}{\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}} \Rightarrow P_i = \underset{3DH.}{\begin{bmatrix} 1 \\ 2 \\ 3 \\ 1 \end{bmatrix}}$

$$p_i = MP_i = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 1 & 0 & 3 & 4 \\ 1 & 1 & 1 & 1 \end{bmatrix}\begin{bmatrix} 1 \\ 2 \\ 3 \\ 1 \end{bmatrix} = \underset{2DH}{\begin{bmatrix} 18 \\ 14 \\ 7 \end{bmatrix}} = \underset{\underline{2D}}{\begin{bmatrix} 18/7 \\ 2 \end{bmatrix}} \rightarrow \text{2D image coordinate.}$$

2e) $$\begin{array}{c} \text{for} \\ \text{point}_i \end{array} \left\{ \begin{bmatrix} P_i^T & 0^T & -x_i P_i^T \\ 0^T & P_i^T & -y_i P_i^T \\ & & \end{bmatrix} \begin{bmatrix} [m_1] \\ [m_2] \\ [m_3] \end{bmatrix} = \begin{bmatrix} 1\,2\,3\,1 & 0\,0\,0\,0 & -100 & -200 & -300 & -100 \\ 0\,0\,0\,0 & 1\,2\,3\,1 & -200 & -400 & -600 & -200 \\ \vdots & \vdots & \vdots & & \vdots & \\ \vdots & \vdots & \vdots & & \vdots & \end{bmatrix} \begin{bmatrix} m_1 \\ \vdots \\ \vdots \\ m_n \end{bmatrix} \right.$$

$$= \begin{bmatrix} 0 \\ \vdots \\ \vdots \\ 0 \end{bmatrix}.$$

2f) Minimal number of points that is necessary to be able to find a unique solution for M is ~~8~~6.
Solution to the problem is obtained by performing singular value decomposition on the $2n \times 12$ matrix and taking the last column of the matrix V.
where, $A = UDV^T$; A is our $2n \times 12$ matrix
i.e, 12×12 matrix formed by the equations.

2g) The principal used to extract the unknown camera parameters from the projection matrix M is :-

$$M = R^* \left[K^* \mid T^* \right]$$

We take dot product of the rows in M as the rotation matrix has the orthogonal vectors along the rows, thereby cancelling out some unknown.

2h) We use, $\{p_i\}_{i=1}^{n} \Leftrightarrow \{P\}_{i=1}^{n}$ correspondence of the image is world points given as input.

We use $p_i = MP_i$ and the estimated projection matrix M to find the image points corresponding to the world points and compare them with the known point.

$$E(K^*, R^*, T^*) = \sum_{i=1}^{n} \left(x_i - \frac{m_1^T P_i}{m_3^T P_i}\right) + \left(y_i - \left(\frac{m_2^T P_i}{m_3 P_i}\right)\right)^2$$

2 i) Planar calibration:

→ Estimate 2D homogenity (projective map) between calibration plane and image (for several images)

→ Estimate intrinsic parameters

→ Compute extrinsic parameters for view of interest.

In non-planar calibration one View of the calibration target is enough to calibrate the camera parameters, whereas for planar calibration, we need atleast 3 different view of calibration target.