Problem Set 10 Solutions

Computer Vision
University of Bern
Fall 2021

1 Outlier Rejection

The RANSAC algorithm is used for outlier rejection, and it repeats the following four steps.

- Select a small set of data points randomly.
- Compute model parameters for those points.
- Find inliers to this model.
- If the number of inliers are large enough, recompute model parameters.

1. Let p denote the probability of a data-point being an outlier. Assume we choose the seed point randomly from all points in an i.i.d. fashion. Compute the probability of finding the correct model parameters if we try N samples and we need at least m points for parameter estimation.
   Solution
   The probability of selecting an inlier is $1-p$. The probability of selecting m inlier for the estimation (therefore it is a good model) is $(1-p)^m$. Not finding a good model has a probability of $1-(1-p)^m$. If we try it N times, $(1-(1-p)^m)^N$ is the probability that we do not find a good model in either sample. Therefore the probability of finding at least one good set of points for the model is $1-(1-(1-p)^m)^N$.
2. Let's register two images to create a panorama image. For this you have to use a homographic transformation model between images. Assume that

10 percent of the matching points are incorrect. How many times do we need to try sampling the points to find the correct model with 99 percent probability?

Solution

For panorama pictures we need $m = 4$ point pairs. Substituting into the equation $P = 1-(1-(1-p)^m)^N$, we get $N = \log(0.01)/\log(0.3439) = 4.31$. We need at last $N = 5$ samples.

3. Which is more difficult? Stereo matching or image stitching for panorama pictures? Assume that the quality of feature matching is the same in both cases and you use RANSAC.

   Solution

   For the panorama $m = 4$, for stereo $m = 8$. We can see that with bigger m the probability of finding a good model decreases (when N is the same). Stereo matching is more difficult.

4. Give a scenario, where RANSAC would fail. How would you change RANSAC to cope with the problem?

   Solution

   At step four of the RANSAC procedure the model parameters are estimated from the inlier points of that model. The problem is that one needs to define an error function and a threshold for deciding if a point is an inlier. If the threshold is too low, only the seed points will be valid. If it is too high, most points will be considered inliers regardless of their quality. RANSAC can fail when this threshold is not set properly.

   Another way RANSAC can fail is when the images contain repeated patterns (for example fishes in an aquarium). Correspondences will be found across all instances of that pattern. The likelihood of having m points (pairs) coming from a consistent pattern (pair) is very small. We can solve this problem by changing the sampling. Instead of choosing the points independently, we choose points that are close to each other, thus increasing the chance that they come from the same instance of repeated pattern (same fish).

2 Sliding windows

The pipeline of the sliding window approach is simple. First we compute a feature representation of the image. It is a 3 dimensional array. The first two dimensions correspond to the image coordinates and the third are the channels of features. An important parameter is the stride. When the features are computed n pixels apart, the stride is n. In the case of raw RGB "features" the stride is 1 and the number of channels are 3. In the case of HOG features, the stride is usually 8 and there are 31 channels.

1. The size of the image is 400×400 pixels. We use HOG features with stride 8 and the number of channels is $c = 31$. The template size is 5×5 pixels. How long does it take to compute the feature responses, if we have a processor with 10 GFlops?
   Solution.
   Denote the image size $L = 400$, stride $s = 8$ and template size $t = 5$. The feature representation of the image has height and width

$$h = w = \left\lfloor \frac{L - t}{s} + 1 \right\rfloor = 50.$$

   We have to evaluate the template at $h \cdot w = 50^2$ positions, and each evaluation takes $2 \cdot t^2 \cdot c$ floating point operations. Computing all feature responses takes $2 \cdot 50^2 \cdot t^2 \cdot c \approx 4 \cdot 10^6$ FLOP. With 10 GFLOP per second this takes 0.4 ms.

2. Consider different scales. The image size is 400×400 pixels for the largest magnification. We use 4 octaves and an octave resolution $r = 10$ to render the images at different scales. All the other parameters are the same as above. What is the running time?
   Solution.
   An octave difference in magnification means reducing the resolution 2 times. The octave resolution is 10 which means we sample the resolution between two octaves equally spaced (in log space) 10 times. The feature response computation depends on the area of the image, i.e. L^2 pixels. The time to compute all resolutions is

$$T_n = t_0(1 + s^2 + s^4 + \cdots s^{2n}) = t_0 \sum_{i=0}^{n} s^{2i},$$

   where t_0 is the time for the base scale and $s = 2^{-\frac{1}{r}} = 0.93$. As $n \to \infty$ the limit of this geometric series is $T = \frac{t_0}{1-s^2} = \frac{0.4}{1-0.93^2} \approx 3$ ms.

3 Pictorial Structures

The score of part placements can be defined by the function J below, where l_i are the part locations $(i = 1, \ldots, k)$, $m_i(l_i)$ are the unary terms and $d_{ij}(l_i, l_j)$ are the binary terms between part placements. The part placements are on a discrete grid, l_i can only be placed at N locations.

$$J(L) = \sum_i m_i(l_i) + \sum_{(v_i, v_j) \in E} d_{ij}(l_i, l_j) \tag{1}$$

We also know that the the structure of the graph (V, E) is a tree. We need to minimize J with respect to l_i to get the optimal part placements.

1. What happens when $d_{ij}(l_i, l_j) = 0$ for all $(v_i, v_j) \in E$? What is the running time of the optimization?
   Solution. When the binary terms become zero, the optimization becomes separable. The running time is $\mathcal{O}(kN)$.

2. What happens when $d_{ij}(l_i, l_j) = 0$ when $l_i = l_j$ and $d_{ij}(l_i, l_j) = +\infty$ otherwise? What is the running time of the optimization?
   Solution. All parts have to be placed at the same location. The function to be minimized becomes $J(l) = \sum_i m_i(l)$. Computing $J(l)$ takes $\mathcal{O}(kN)$, finding the optimum takes $\mathcal{O}(N)$, so the running time is $\mathcal{O}(kN)$.

3. When $d_{ij}(l_i, l_j) = a_{ij}\|l_i - l_j\|^2$, what is the running time of the most naive optimization?
   Solution. Trying all possible placement brute force takes $\mathcal{O}(N^k)$ time. This is infeasible.

4. Give a faster optimization algorithm by eliminating the variables leaf by leaf.
   Solution. Consider the graph with k nodes. We can assume that the node v_k is a leaf and it is connected to node v_1 (othervise we can simply rename the nodes).

$$
\begin{aligned}
J(L) &= \sum_{i=1}^{k} m_i(l_i) + \sum_{(v_i,v_j)\in E} d_{ij}(l_i, l_j) && (2)\\
&= \sum_{i=1}^{k-1} m_i(l_i) + \sum_{(v_i,v_j)\in E-(v_1,v_k)} d_{ij}(l_i, l_j) + && (3)\\
&\quad m_k(l_k) + d_{k1}(l_k, l_1) && (4)
\end{aligned}
$$

   We optimize J first with respect to l_k. We can see that it is equivalent to optimize $m_k(l_k) + d_{k1}(l_k, l_1)$, the rest does not depend on l_k. Brute force trying all possibilities for all l_1 values takes $\mathcal{O}(N^2)$ times and it results in the function $f(l_1) = \min_{l_k} m_k(l_k) + d_{k1}(l_k, l_1)$.

$$
\begin{aligned}
J(L) &= \sum_{i=1}^{k-1} m_i(l_i) + \sum_{(v_i,v_j)\in E-(v_1,v_k)} d_{ij}(l_i, l_j) + f(l_1) = && (5)\\
&= \sum_{i=1}^{k-1} m_i(l_i)' + \sum_{(v_i,v_j)\in E'} d_{ij}(l_i, l_j), && (6)
\end{aligned}
$$

   where $m'_i = m_i$ for $i > 1$ and $m'_1 = m_1 + f$ and $E' = E - (v_1, v_k)$. Computing m'_1 this takes $\mathcal{O}(N)$ time. This form is exactly the same as

before, but the number of vertices decreased by 1. We have to repeat the procedure k times, thus the running time is $\mathcal{O}(kN^2)$.

5. The generalized distance transform solves optimization $f(p) = \min_{q \in [1,N]} m(q) + d(p,q)$. It takes $\mathcal{O}(N)$ time to calculate. What is the running time of optimizing J using distance transform?
   Solution.
   Notice that the distance transform optimizes exactly the same function as we solved in the previous question by brute force. Now one leaf elimination takes $\mathcal{O}(N)$ steps, so the complete running time is $\mathcal{O}(kN)$.