Vanish Points Estimation Based on EM Algorithm

Members: Hu Weilin (A0195438R), Liu Ruiping (A0198797W)

Overview

EM algorithm is an iterative way to solve MAP problem, by doing estimation and maximization. In our project, we are solving Manhattan World problem with three vanishing points in each picture. We need to find these three vanishing points and assign top 2000 pixels with the largest gradient in the picture to one of the vanishing points.

To apply EM algorithm in vanishing point detection problem, we firstly need to know the equation to compute of the location of vanishing points.

Given camera intrinsics K and rotation matrix R, the vanishing points can be computed as:

$$v = K * R * V$$

Once the picture is given, we can get camera intrinsic matrix *K* directly. R is the camera rotation matrix, which can be computed from the camera rotation angle (α, β, γ). V is the vanishing point direction matrix which is given. Hence, if we select a tuple of camera rotation angles, we can compute the location of three vanishing points.

Vanishing points estimation (equivalent to camera rotation angle estimation) and pixel assignment is a chicken and egg problem. If we know either one, we can directly know the other one. But the sad truth is that we know neither of them. Here comes the EM algorithm.

In the E-step, we assume we already know about the camera rotation matrix (i.e. vanishing point is known) and we will solve the assignment problem using the camera rotation matrix from the previous M-step. The result of the assignment is a weight matrix w of size 2000*4, each element of which (w_{mp}) is the probability that the m_{th} pixel belongs to the p_{th} model.

In the M-step, we re-estimate the camera rotation matrix by maximizing the expected log-posterior function Q which will be explained in detail in the M-step section.

Initialization

In the initialization part, we aim at finding a good camera rotation matrix to get an initial vanishing point by searching through the parameter space spanned by (α, β, γ). In each searching round, we compute and sum over the log posterior probability for each combination of pixel and vanishing point model. The computation equation is shown below:

$$S = \sum_p \quad \sum_m \quad logP(m|\phi_p, f(\Psi))$$

Where *P(m)* is the prior distribution, *p* is the pixel and *m* is the vanishing point model. In this problem we have three vanishing points so we have four models. The fourth model means that the pixel does not belong to any vanishing point.

Then by maximizing S over $f(\Psi)$, we can choose the corresponding camera rotation matrix as our initial R.

E-step

The feature of the pixel is its gradient. We can compute the gradient direction of a pixel using python cv2.sobel, and we can easily compute the edge direction from each pixel to vanishing points (Note: the vanishing point location has been given by either initialization step or previous M-step). We can do a subtraction between these two directions and remove the polarity then we get an angle error representing the assignment error of the pixel to a certain vanishing point.

From the *'Atlanta World'* paper, we know that the angel error follows the zero-mean Gaussian distribution whose standard deviation is 0.5.

As we stated before, the goal of E-step is to give a new weight matrix w, so we compute each element of w, i.e. w_{mp}, by following the equation:

$$w_{mp}=\frac{1}{Z_p} P(E_p|m_p)P(\phi_p|m_p,V^{old})P(m_p)$$

Note that Z_p is a normalization factor over 4 vanishing point models for this specific pixel m and the first probability item can be ignored because we only use pixels on the edge. As mentioned before, we have

$$P(\phi_p|m_p,V^{old})=\mathcal{N}(\text{err} \mid \mu,\sigma)$$

Which error=$\theta_1-\theta_2$,$\theta_1$is the direction with regards to vanishing points and θ_2 is the gradient direction for a specific pixel m. $P(m_p)$ is the prior weight probability distribution which is given by TAs.

The computed w_{mp} is the only thing E-step provide to the next M-step.

M-step

In M-step, by maximizing the expected log posterior (Q function) with respect to the camera rotation matrix, we can get the updated camera rotation matrix based on the weight matrix computed in E-step. As shown in the beginning, we can sum up the product of log posteriors of each pixel and the corresponding element in the weight matrix we have got in the E-step. The computation equation can be shown as follows:

$$Q(\Psi,\Psi^{old})=\sum_p \quad \sum_m \quad w_{mp} log P(\phi_p|m,f(\Psi))$$

$$\Psi^{new} = argmax_{\Psi}\ Q(\Psi, \Psi^{old})$$

Where w_{mp} is the weight matrix from E-step, and the log likelihood is equal to the Gaussian Distribution over the angle error which denoted asϕ_p .

$$P(\phi_p|m, f(\Psi)) = Gaussian(\phi_p) \simeq exp(-{\phi_p}^2)$$

In addition, from the given paper we know the gaussian distribution is equivalent to the exponential function of the square of angle error. By doing the logarithm, the exponential operation is eliminated. Therefore, maximizing Q function problem is equivalent to minimizing the square of angle error.

To summarize, in M-step we can re-compute the optimized camera rotation matrix using the weighted function. We use python scipy least_square function to handle this problem efficiently.

Result Analysis

After repeating E-step and M-step for 100 iterations, we found the camera rotation matrix and weight matrix has converged. After a series of operations, we can derive three vanishing points and assign each of 2000 chosen pixels to one of these three vanishing points. The result image after 100 iterations is shown as ***P1030001_results.png*** and ***P1080055_results.png*** respectively.