Computer Vision, 3D Geometry and Machine Learning Notes

Samarth Brahmbhatt

November 20, 2018

Contents

Chapter 1

3D Geometry

1.1 Homogeneous Coordinates

The homogeneous representation of an image point $\mathbf{x} = [u, v]$ is denoted as $\tilde{\mathbf{x}} = [u, v, 1]$. A line in the image passing through $\tilde{\mathbf{x_1}}$ and $\tilde{\mathbf{x_2}}$ is $\tilde{\mathbf{l}} = \tilde{\mathbf{x_1}} \times \tilde{\mathbf{x_2}}$.

1.2 Plücker Coordinates

1.3 Types of Transforms

Table 1.1 shows the types of transforms on 2D homogeneous coordinates. Each transform preserves all the quantities in its row and below it.

Transform	DoF	Preserves
Translation	2	absolute orientation
Rotation	1	distances
Similarity	4	angles
Affine	6	parallel lines
Projective	8	ratios of distances

Table 1.1: Types of transforms on 2D homogeneous coordinates

1.4 Single View Geometry

If the transformation from world to camera coordinates is ${}^cT_w = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix}$, the pinhole camera projection equation is

$$\tilde{\mathbf{x}} = K[R|t]^{\mathbf{w}}\tilde{\mathbf{X}} \tag{1.1}$$

where $\tilde{\mathbf{X}}$ is the homogeneous representation of the 3D point in world coordinates, and

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{1.2}$$

is the ideal pinhole camera intrinsics matrix. $P = K[R|t]$ is called the camera projection matrix.

1.4.1 Dissecting the Camera Projection Matrix

Let's denote the columns of P as $P = [P_1|P_2|P_3|P_4]$. Now, $P_1 = P \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$. Hence P_1 is the image of vanishing point in the world coordinate X direction. Similarly for P_2 and P_3. P_4 is the image of the world origin.

Let's denote the rows of P as $P = \begin{bmatrix} P_1^T \\ P_2^T \\ P_3^T \end{bmatrix}$. Since $P_1^T = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} P$, P_1^T is the plane passing through the camera center and the image line $[1, 0, 0]$. To see this, consider a projected point $\tilde{\mathbf{x}} = P\tilde{\mathbf{X}}$. $\tilde{\mathbf{x}}$ lies on the image line $\tilde{\mathbf{l}}$ iff

$$\tilde{\mathbf{l}}^T\tilde{\mathbf{x}} = 0 \tag{1.3}$$

$$\iff \tilde{\mathbf{l}}^T P\tilde{\mathbf{X}} = 0 \tag{1.4}$$

$$\iff (P^T\tilde{\mathbf{l}})^T\tilde{\mathbf{X}} = 0 \tag{1.5}$$

Thus the plane corresponding to image line $\tilde{\mathbf{l}}$ is $P^T\tilde{\mathbf{l}}$.

Representing lines with Plücker coordinates

1.5 2-view Geometry

1.6 Multi-view Geometry

1.6.1 Panoramas

The simplest panorama stitching can be done by modeling the camera motion between images as a 2D planar homography. This model holds true for two cases:

- **Planar scene**. w.l.o.g. we can choose the world coordinates such that the scene is on the XY plane. The camera projection equation becomes:

$$\tilde{\mathbf{x}} = P\tilde{\mathbf{X}} \tag{1.6}$$

$$= \begin{bmatrix} P1 & P2 & P3 & P4 \end{bmatrix} \begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix} \tag{1.7}$$

$$= \begin{bmatrix} P1 & P2 & P4 \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \tag{1.8}$$

  which is a homography. Since the transformations from all world points to image points are homographies, the camera motions between image frames are also homographies.

- **In-place rotation**. Suppose the camera moves by a rotation R between two images. The image locations of a 3D point $\tilde{\mathbf{X}}$ are w.l.o.g.

$$\tilde{\mathbf{x}}_1 = K[I|0]\tilde{\mathbf{X}} \tag{1.9}$$

$$= KX \tag{1.10}$$

  and

$$\tilde{\mathbf{x}}_2 = K[R|0]\tilde{\mathbf{X}} \tag{1.11}$$

$$= KRX \tag{1.12}$$

$$= KRK^{-1}KX \tag{1.13}$$

$$= KRK^{-1}\tilde{\mathbf{x}}_1 \tag{1.14}$$

Thus, the relationship between $\tilde{\mathbf{x}}_1$ and $\tilde{\mathbf{x}}_2$ is a homography $H = KRK^{-1}$.

This is why smartphone panorama apps advise you to rotate in place while capturing a panorama.

Chapter 2

Machine Learning

2.1 Iterative optimization

Many problems involve minimizing the squared L2 norm of some error, which is a non-linear function of some parameters θ. This error ϵ is the difference between observed values $\mathbf{Y}$ and values predicted from an input $\mathbf{X}$ by a non-linear function $f(\mathbf{X}, \theta)$: $\epsilon(\theta) = f(\mathbf{X}, \theta) - \mathbf{Y}$. The objective function is

$$g(\theta) = \frac{1}{2}\epsilon(\theta)^T \epsilon(\theta) \tag{2.1}$$

Its derivatives are:

$$g_\theta = \epsilon_\theta^T \epsilon \tag{2.2}$$

$$g_{\theta\theta} = \epsilon_\theta^T \epsilon_\theta + \epsilon_{\theta\theta}^T \epsilon \tag{2.3}$$

The three famous iterative optimization methods differ by different choices for approximating the Hessian $g_{\theta\theta}$.

1. **Newton's method**. Hessian is computed fully. The drawback is that computing the Hessian might be complicated for some methods. This method assumes that the cost function is approximately quadratic around the minimum, and shows rapid convergence in that area. To see this, consider the Taylor series expansion of the cost function:

   $$g(\theta + \Delta) = g + g_\theta \Delta + \frac{1}{2}\Delta^T g_{\theta\theta} \Delta + \dots \tag{2.4}$$

   To find the minimizer Δ, we take the derivative w.r.t. Δ and set it to 0, ignoring terms above the second order. This gives us the update equation

   $$g_{\theta\theta}\Delta = -g_\theta \tag{2.5}$$

2. **Gauss-Newton**. Hessian is approximated as $\epsilon_\theta^T \epsilon_\theta$, avoiding computing the second derivative. Hence the update equation is

$$\epsilon_\theta^T \epsilon_\theta \Delta = -g_\theta \tag{2.6}$$

3. **Gradient Descent**. Takes steps along the direction of steepest descent of g. The Hessian is approximated as a scaled identity, leading to the update equation

$$\lambda \Delta = -g_\theta \tag{2.7}$$

   The advantage is that it is always guaranteed to lower the objective function, and the update is simple to compute. However, convergence near the minimum is not as fast as Newton's method, and choosing a right learning rate λ is difficult.

4. **Levenberg-Marquadt**. The Hessian is approximated as a linear addition of the Hessian approximations used in Gauss-Newton and Gradient Descent:

$$(\epsilon_\theta^T \epsilon_\theta + \lambda I)\Delta = -g_\theta \tag{2.8}$$

   Depending on the value of λ, this algorithm can seamlessly switch between Gauss-Newton (rapid convergence near a minimum), and Gradient Descent (guaranteed decrease in cost function).