Joe Phaneuf
Computer Vision 16-720 Spring 2018 Homework 5
Apr. 11 2018

2

2.1 a

Point-to-point Iterative Closest Point defines an energy function to minimize in terms of Euclidean distance between a point, and a point at a subsequent timestep having undergone some transformation. This yields an estimate of the transformation.

Point-to-plane Iterative Closest Point defines an energy function to minimize the dot product of a normal vector estimate and the difference between a vertex and a transformed vertex at a subsequent timestep. This also yields an estimate of the transformation, with an emphasis on maintaining surfaces.

2.2 b

So, we would like to estimate $\tilde{T}_{g,k}$, which transforms points in the camera frame to the world frame, and consequently provides camera pose. To estimate this new transform, we compute multiple iterations of small changes to the transform and solve for a gradient at each iteration. So, we need the incremental transform in solveable form!

At each iteration z, define $\tilde{T}^z_{inc}$ such that

$$\tilde{T}^z_{g,k} = \tilde{T}^z_{inc}\tilde{T}^{z-1}_{g,k} \tag{1}$$

$\tilde{T}^z_{inc}$ contains a rotation and tanslation in 3 dimensions. Using the small angle approximation, we can approximate the rotation component as

$$\tilde{R}^z = \begin{bmatrix} 1 & \alpha & -\gamma \\ -\alpha & 1 & \beta \\ \gamma & -\beta & 1 \end{bmatrix} \tag{2}$$

and the transformation component as

$$\tilde{t}^z = \begin{bmatrix} tx \\ ty \\ tz \end{bmatrix} \tag{3}$$

We'll set up the minimization by equating the current frame transform estimate applied to a camera frame vertex to the incremental transform applied to a world frame vertex.

$$\tilde{T}^z_{g,k}\dot{V}_k = \tilde{R}^z\tilde{V}^g_k(u) + \tilde{t}^z \tag{4}$$

Say that $\tilde{V}^g_k(u) = \begin{bmatrix} v_1 & v_2 & v_3 \end{bmatrix}^T$

Then

$$\tilde{T}^z_{g,k}\dot{V}_k = \begin{bmatrix} v_1 + \alpha v_2 + -\gamma v_3 \\ -\alpha v_1 + v_2 + \beta v_3 \\ \gamma v_1 + -\beta v_2 + v_3 \end{bmatrix} + \tilde{t}^z = \begin{bmatrix} 0 + \alpha v_2 - \gamma v_3 \\ -\alpha v_1 + 0 + \beta v_3 \\ \gamma v_1 - \beta v_2 + 0 \end{bmatrix} + \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} + \tilde{t}^z = \tag{5}$$

$$\begin{bmatrix} 0 - \gamma v_3 + \alpha v_2 \\ \beta v_3 + 0 - \alpha v_1 \\ -\beta v_2 + \gamma v_1 + 0 \end{bmatrix} + \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} + \tilde{t}^z = \left[\tilde{V}_k^g(u)\right]_x \begin{bmatrix} \beta \\ \gamma \\ \alpha \end{bmatrix} + \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} + \tilde{t}^z = \tag{6}$$

$$\begin{bmatrix} \left[\tilde{V}_k^g(u)\right]_x \mid & I \end{bmatrix} \begin{bmatrix} \beta \\ \gamma \\ \alpha \\ tx \\ ty \\ tz \end{bmatrix} + \tilde{V}_k^g(u) \tag{7}$$

This is neat, because the incremental transform parameters can now be solved with linear solving tools.

3

3.1 a

Volumetric fusion methods suffer from high computational cost, and high memory requirements for Voxel grid storage.

3.2 b

Consider a point p mapped into a new frame to point p' by rotation R and translation t.

$$p = Rp + t \tag{8}$$

The point's corresponding normal vector n is mapped to the new frame by

$$n' = Rn \tag{9}$$