OPENCV GOOGLE SUMMER OF CODE MIDTERM REPORT

Recognition and Pose Estimation of 3D Objects Through 3D Features

Tolga Birdal
Ph.D. Candidate, Technical Univesity of Munich
May 29, 2014

This document aims to summarize the developments and the theoretical achievements covered throughout the first term of Google Summer of Code. The developments so far are heavily based on [1].

1 COMPUTATION OF POINT PAIR FEATURES (PPF)

2 INITIAL COMPUTATION OF OBJECT POSE GIVEN PPF

Let me summarize the following notation:

- p_m^i: i^{th} point of the model (p_m^j accordingly)
- n_m^i: Normal of the i^{th} point of the model (n_m^j accordingly)
- p_s^i: i^{th} point of the scene (p_s^j accordingly)
- n_s^i: Normal of the i^{th} point of the scene (n_s^j accordingly)
- $T_{m\to g}$: The transformation required to translate p_m^i to the origin and rotate its normal n_m^i onto the x-axis.
- $R_{m\to g}$: Rotational component of $T_{m\to g}$.
- $t_{m\to g}$: Translational component of $T_{m\to g}$.

- $(p_m^i)'$: i^{th} point of the model transformed by $T_{m\to g}$. ($(p_m^j)'$ accordingly).
- $\mathbf{R_{m\to g}}$: Axis angle representation of rotation $R_{m\to g}$.
- $\theta_{m\to g}$: The angular component of the axis angle representation $\mathbf{R_{m\to g}}$.

2.1 Transforming a Point Pair Onto the Ground Plane

The transformation in a point pair feature is computed by first finding the transformation $T_{m\to g}$ from the first point, and applying the same transformation to the second one. Transforming each point, together with the normal, to the ground plane leaves us with an angle to find out, during a comparison with a new point pair.

We could now simply start writing

$$(p_m^i)' = T_{m\to g} p_m^i$$

where

$$T_{m\to g} = -t_{m\to g} R_{m\to g}$$

Note that this is nothing but a stacked transformation. The translational component $t_{m\to g}$ reads

$$t_{m\to g} = -R_{m\to g} p_m^i$$

and the rotational being

$$\begin{aligned} \theta_{m\to g} &= \cos^{-1}(n_m^i \cdot \mathbf{x}) \\ \mathbf{R_{m\to g}} &= n_m^i \wedge \mathbf{x} \end{aligned}$$

in axis angle format. Note that bold refers to the vector form.

When the scene point p_s^i is also transformed on the same plane as $(p_m^i)'$, the points will be misaligned by a rotational component α. For the sake of efficiency, the paper splits it into two components α_s and α_m. Respectively, these denote the rotations from the transformed scene to the x-axis and from the transformed model to the x-axis. Luckily, both α_s and α_m are subject to the same procedure of computation, which reads as follows:

$$\begin{aligned} \alpha_m &= \tan^{-1}\left(\frac{-(p_m^j)'_z}{(p_m^j)'_y}\right) \text{ for model} \\ \alpha_s &= \tan^{-1}\left(\frac{-(p_s^j)'_z}{(p_s^j)'_y}\right) \quad \text{for scene} \end{aligned}$$

using the fact that on x-plane x=0.

in the implementation, alphas are adjusted to be rotating towards 0.

2.2 Hough-like Voting Scheme

After both transformations the difference of the point pair features remain to be $\alpha = \alpha_m - alpha_s$. This component carries the cue about the object pose. A Hough-like voting scheme is followed over the local model coordinate vector and α, which eventually recovers the object pose.

REFERENCES

[1] B. Drost, M. Ulrich, N. Navab, and S. Ilic, “Model globally, match locally: Efficient and robust 3d object recognition.” in *CVPR*. IEEE, 2010, pp. 998–1005. [Online]. Available: http://dblp.uni-trier.de/db/conf/cvpr/cvpr2010.html#DrostUNI10