Fast Feature-Less Quaternion-based Particle Swarm Optimization for Object Pose Estimation From RGB-D Images

Giorgio Toscana
giorgio.toscana@polito.it
Stefano Rosa
stefano.rosa@polito.it
Politecnico di Torino, Turin, Italy

We present a novel quaternion-based formulation of *Particle Swarm Optimization* for pose estimation which, differently from other approaches, does not rely on 2D or 3D features or machine learning. The quaternion formulation avoids the gimbal lock problem and, unlike other rotation formalisms, doesn't require conversions to and from rotation matrix form at each step. The objective function is based on raw 2D depth information only, under the assumption that the object region is segmented from the background. This makes the algorithm suitable for pose estimation of objects with large variety in appearance, from lack of texture to strong textures, for the task of robotic grasping. We find candidate object regions using a graph-based image segmentation approach that integrates color and depth information, but the PSO is agnostic to the segmentation algorithm used. Given a candidate segmented object, its pose is estimated using PSO. The algorithm is implemented on GPU, and the nature of the objective function allows high parallelization.

We focus on the problem of object detection for robotic grasping, and in particular we are most interested in the Amazon Picking Challenge (APC) scenario. A well known approach is LINEMOD [1], but tests of LINEMOD showed only 32% accuracy in an APC-like scenario. Other commonly used approaches, such as tabletop from the Point Cloud Library, rely on object textures and are not suitable for many common texture-less objects. Moreover, all these methods are limited to objects lying on a flat tabletop, and are generally not robust to occlusions.

The quaternion formulation of the pose estimation problem avoids the gimbal lock problem, and the objective function is based on raw 2D depth information only, under the assumption that the object region is segmented from the background. This makes the algorithm suitable for pose estimation of objects with large variety in appearance, from lack of texture to strong textures, for the task of robotic grasping.

In *Particle Swarm Optimization* (PSO) a set of candidate solutions (particles) $\mathbf{p}_i(t) = (\mathbf{x}_i(t), \mathbf{v}_i(t))$, where $\mathbf{x}_i$ and $\mathbf{v}_i$ are the position and velocity of particle i at time t, is maintained in the search space. The algorithm iterates over three steps: fitness evaluation, update of individual and global best position, velocity and position update for each particle.

The angular velocity update equation for the i-th particle is formulated as follows:

$$\begin{aligned} \omega_i(t+1) = w\omega_i(t) + \\ c_1 r_1 \left[2\mathrm{Log}\left(\mathbf{q}_{pbest_i}(t) \star \mathbf{q}^*_{current_i}(t)\right)\right] + \\ c_2 r_2 \left[2\mathrm{Log}\left(\mathbf{q}_{gbest}(t) \star \mathbf{q}^*_{current_i}(t)\right)\right] \end{aligned} \quad (1)$$

where $\mathbf{q}$ is the quaternion form of the particle orientation, w, c_1, c_2 (with $0 \leq w \leq 1.2$, $0 \leq c_1 \leq 2$, $0 \leq c_2 \leq 2$) are tunable parameters; r_1, r_2 are random values. The orientation of the i-th particle is then updated by means of the discrete form of the quaternion kinematics:

$$\mathbf{q}_i(t+1) = \cos(\psi(t))\,\mathbf{q}_i(t) + \frac{1}{2}\frac{\sin(\psi(t))}{\psi(t)}\mathbf{q}_i(t) \star \omega_i(t+1)T_c, \psi(t) = \|\omega_i(t+1)\|_2 \frac{T_c}{2} \quad (2)$$

T_c represents the integration time of the discrete time quaternion kinematics. In this work T_c just collapses to a tunable parameter as we are dealing with iteration steps (t) rather than with the true definition of time.

Each particle renders its pose hypothesis against the depth map of the cluster. The fitness value of the j-th particle is thus computed as follows:

$$\Phi_j = \frac{\alpha}{N_{R_j}} \sum_{i=1}^{N_{R_j}} \left(z_{K_i} - z_{R_{ij}}\right)^2 + \beta \frac{\mu_j + \kappa_j}{2} \quad (3)$$

where: N_{R_j} is the number of pixels of the depth map rendered by the j-th particle, $z_{R_{ij}}$ is the depth value of the pixel i rendered by the j-th particle, while z_{K_i} is the corresponding depth value of the cluster at pixel i. α and β are two constant parameters used to weight the two terms of the fitness function. μ_j gives the percentage of cluster pixels that are not covered by the rendering of the object 3D model of the particle j. κ_j is the complement of μ_j i.e., it gives the percentage of rendered pixels of the particle j that are not covered by valid depth values in the cluster depth map.

Approach	[1]	[3]	Our Appr.
Sequence			
Bench Vise	0.85	0.96	0.72
Driller	0.69	0.9	0.9
Phone	0.56	0.73	0.98
Duck	0.58	0.91	0.97
Eggbox	0.86	0.74	0.95
Glue	0.44	0.68	0.8

Approach	[1]	[3]	Our Appr.
Sequence			
Coffee Cup	0.82	0.88	0.85
Shampoo	0.63	0.76	0.9
Juice Carton	0.49	0.87	0.4
Milk	0.18	0.39	0.67

Table 1: (a) Comparison between different approaches on the[2] dataset. (b) Comparison between different approaches on the[3] dataset.

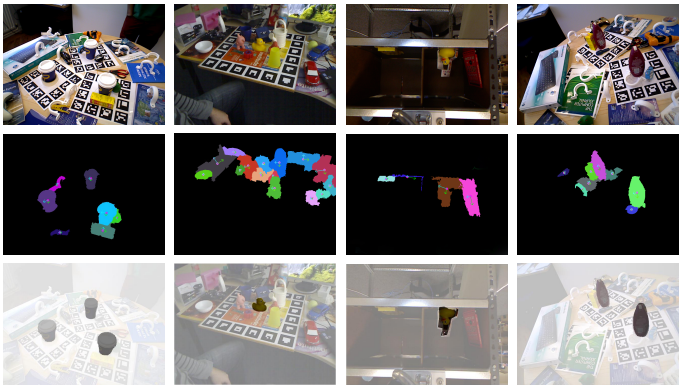

Figure 1: Examples of the approach on different datasets. First row: RGB images; second row: segmented objects; third row: detected object superimposed.

The segmentation phase provides a rough approximation of the 3D centroid of a cluster. The latter is not necessarily the 3D centroid of the real object as errors in the clustering step might lead to either over-segmentation or under-segmentation. The segmentation step cannot generate an estimate of the object orientation, so the object attitude initialization and optimization are performed on the whole surface of the northern hemisphere of $\mathbb{S}^3$.

In our implementation, the rendering of each particle is done directly in GPU. Our rendering pipeline is based on the optimized version of the **edge function**, that allows fast and parallel processing of the object mesh triangles. The particles rendering phase offers two levels of parallelism: each particle renders its object model independently and the rendering of the triangles for each model is also parallelized.

We tested our approach on two public datasets for 3D pose estimation [1] and [3]. The software has been developed using the CUDA library in C++ under Linux and runs on GPU. The source code will be available. In our experiments we used 1024 particles for the PSO and run 10 PSO iterations for each segmented cluster and the total time is 85ms for each cluster. Global topology was used for the PSO. Results are shown in Table 1 and Figure 1.

[1] S. Hinterstoisser, S. Holzer, C. Cagniart, S. Ilic, K. Konolige, N. Navab, and V. Lepetit. Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes. 2011.

[2] Stefan Hinterstoisser, Vincent Lepetit, Slobodan Ilic, Stefan Holzer, Gary Bradski, Kurt Konolige, and Nassir Navab. Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. In *Computer Vision–ACCV 2012*, pages 548–562. Springer, 2012.

[3] Alykhan Tejani, Danhang Tang, Rigas Kouskouridas, and Tae-Kyun Kim. Latent-class hough forests for 3d object detection and pose estimation. In *Computer Vision–ECCV 2014*, pages 462–477. Springer, 2014.