3D Model-based 6D Object Pose Tracking on RGB Images using Particle Filtering and Heuristic Optimization

Mateusz Majcher and Bogdan Kwolek
AGH University of Science and Technology, 30 Mickiewicza, 30-059 Krakow, Poland

Keywords: 6D Object Pose Tracking, Region-based Pose Estimation, Image Segmentation, Optimization.

Abstract: We present algorithm for tracking 6D pose of the object in a sequence of RGB images. The images are acquired by a calibrated camera. The object of interest is segmented by an U-Net neural network. The network is trained in advance to segment a set of objects from the background. The 6D pose of the object is estimated through projecting the 3D model to image and then matching the rendered object with the segmented object. The objective function is calculated using object silhouette and edge scores determined on the basis of distance transform. A particle filter is used to estimate the posterior probability distribution. A k-means++ algorithm, which applies a sequentially random selection strategy according to a squared distance from the closest center already selected is executed on particles representing multi-modal probability distribution. A particle swarm optimization is then used to find the modes in the probability distribution. Results achieved by the proposed algorithm were compared with results obtained by a particle filter and a particle swarm optimization.

1 INTRODUCTION

Estimating the 6-DoF pose (3D rotations + 3D translations) of an object with respect to the camera is very important task. It is expected that in a near future the robots will be used in larger scale in less structured environments such as shops, households and hospitals. In such applications, robots will need to be more autonomous and have abilities to estimate 6 DOF pose of objects randomly placed in environment. Due to considerable applicability potential, considerable research efforts have been devoted to tackling the 6D pose estimation problem by computer vision community (Brachmann et al., 2014), robotics community (Hinterstoisser et al., 2013) and augmented reality (Marchand et al., 2016). In virtual reality applications a precise object pose is needed to perform interaction with objects as well as to initialize tracking. In robotic applications the robot pose is needed to avoid collisions, to allow a robot to manipulate an object or to avoid moving into the object.

Estimating the object pose using only a single monocular camera is the most generic approach. In context of robotics such monocular approaches are attractive in grasping scenarios when single RGB camera is usually attached to an arm of a robot. In conventional approaches, the 6D pose of the object is recovered by optimizing the difference between the 2D observation of the object in the current image and an estimated synthetic 2D projection of a 3D model abstraction of the considered object, which is parameterized by the sought pose. Thus, the pose estimate is recovered through selection the best matching viewpoint onto the object or on the basis of 2D-3D correspondences between such local features and a Perspective-n-Point (PnP) algorithm (Fischler and Bolles, 1981). In case of tracking, the object is assumed to be seen in a sequence of consecutive images and the object motion is supposed to be relatively small between two consecutive frames. The correspondence-based approaches require rich texture features. They calculate the pose using the PnP and recovered 2D-3D correspondences, often in a RANSAC (Fischler and Bolles, 1981) framework for outlier rejection. While PnP algorithms are usually robust when the object is well textured, they can fail when it is featureless or when in the scene there are multiple objects occluding each other.

Another approaches consist in attaching artificial markers or fiducials to objects of interest. Such approaches usually permit tracking by detection, meaning that the object absolute pose can be determined in each image in real-time without exploiting temporal continuity assumptions. However, having on regard that the marker detection is based on high contrast edges in the image as well as overall high inten-

Majcher, M. and Kwolek, B.
3D Model-based 6D Object Pose Tracking on RGB Images using Particle Filtering and Heuristic Optimization.
DOI: 10.5220/0009365706900697
In *Proceedings of the 15th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2020) - Volume 5: VISAPP*, pages 690-697
ISBN: 978-989-758-402-2; ISSN: 2184-4321

sity contrast between the black and the white areas of the marker, such methods are prone to motion blur caused by fast marker or camera movement, mainly due to unsharp or blurred edges (Marchand et al., 2016). Active pose estimation approaches overcome most of limitations related to passive markers or features and are frequently used in conjunction with PnP algorithms. When no marker and no additional light sources can be used, the point correspondences can be computed passively from so-called natural features visible on the object. The most popular and commonly used natural features are so-called point and edge features. A large variety of object pose estimation approaches relying on natural point features have been proposed in the past, e.g. (Lepetit et al., 2004; Vidal et al., 2018), to enumerate only some of them.

In general, features can either be an encoding of image properties or a result of learning. In recently proposed algorithm (Brachmann et al., 2016), which is based on learning so-called object coordinates, an auto-context random forest processes the image to predict object labels and object coordinates. As a result, pixel-wise distributions of object labels and object coordinates are calculated. The distributions over object labels are used to sample hypotheses for all objects at once. Then, pre-emptive RANSAC is used to determine preliminary pose estimates. Finally, these poses are refined on the basis of object coordinate distributions. In (Hinterstoisser et al., 2011; Hinterstoisser et al., 2013), holistic template-based methods, which can deal with texture-less objects in 6D pose recovering in cluttered scenes have been introduced.

A first attempt to use a convolutional neural network (CNN) for direct regression of 6DoF object poses was PoseCNN (Xiang et al., 2018). In general, two main CNN-based approaches to 6D pose object pose estimation have emerged: either regressing the 6D object pose from the image directly (Xiang et al., 2018) or predicting 2D key-point locations in the image (Rad and Lepetit, 2017), from which the object pose can be determined by the PnP algorithm.

There are several publicly available datasets for benchmarking the performance of algorithms for 6D object pose estimation, including OccludedLinemod (Brachmann et al., 2014), YCB-Video (Xiang et al., 2018). However, the current datasets do not focus on 6D object tracking using RGB image sequences. Most of the RGB image-based approaches to 6D pose estimation focuses on accuracies as well as processing times. As pointed out in recent work (Deng et al., 2019), the majority of current techniques to 6D object pose estimation ignore temporal information and deliver only a single hypothesis for object pose. In discussed work, a Rao-Blackwellized Particle Filter (PoseRBPF) for object pose tracking has been proposed. In this work we investigate the problem of 6-DOF object pose tracking from RGB images, where the object of interest is rigid and its 3D model is known. At the beginning, the object is segmented from the background using an U-Net convolutional neural network. The network is trained in advance using a small set of object images. A particle filter (PF) (Doucet et al., 2000) combined with a particle swarm optimization (PSO) (Kennedy and Eberhart, 1995; Sengupta et al., 2019) is then utilized to estimate the 6D object pose by projecting the 3D object model and then matching the projected image with the image acquired by the camera. The tracking of 6D object pose is formulated as a dynamic optimization problem, where a particle filter is used to represent the 6D pose probability distribution, the k-means++ is employed to find clusters in the probability distribution, and a PSO is utilized to seek for the modes in the probability distribution. In order to reduce work needed to prepare the 3D object model as well as to determine the ground-truth poses we employed an automated turntable setup.

2 OBJECT SEGMENTATION

The architecture of neural network has been based on the U-Net (Ronneberger et al., 2015) in which we can distinguish a down-sampling (encoding) path and an up-sampling (decoding) path, see Fig. 1. In the down-sampling path there are five convolutional blocks. Each block has two convolutional layers with 3×3 filters and stride equal to 1. Down-sampling is realized by max pooling with stride 2×2 that is applied on the end of every blocks except the last one. In the up-sampling path, each block begins with a deconvolutional layer with 3×3 filter and 2×2 stride, which doubles the dimension of feature maps in both directions and decreases the number of feature maps by two. In each up-sampling block, two convolutional layers decrease the number of feature maps, which arise as a result of concatenation of deconvolutional feature maps and the feature maps from corresponding block in the encoding path. Finally, a 1×1 convolutional layer is used. The neural network was trained on RGB images of size 288×512. In order to reduce training time, prevent overfitting and increase performance of the U-Net we added Batch Normalization (BN) (Ioffe and Szegedy, 2015) after each Conv2D. BN is a kind of supplemental layer that adaptively normalizes the input values of the following layer, mitigating the risk of overfitting. Since it improves gradient flow through the network, it reduces depen-

dence on initialization and higher learning rates are achieved. Data augmentation is useful for the reduction of overfitting and it has also been applied during the network training.

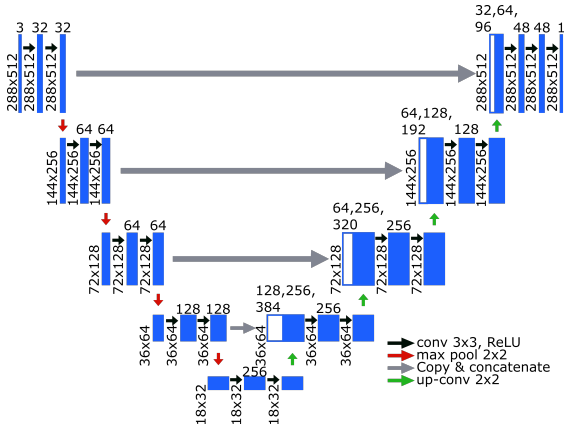

Figure 1: Architecture of U-Net for object segmentation.

The pixel-wise cross-entropy has been used as the loss function for object segmentation:

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)] \quad (1)$$

where N stands for the number of training samples, y is true value and $\hat{y}$ denotes predicted value.

3 6D OBJECT POSE TRACKING

At the beginning of this Section we discuss 6D object pose tracking using PSO. Then, in Subsection 3.1.2 we present the fitness function. Afterwards, we present 6D object pose tracking using particle filtering. In Subsection 3.2.2 we outline observation and motion models. Finally, in Subsection 3.3 we present PF-PSO algorithm for 6D object pose tracking.

3.1 6D Object Pose Tracking using PSO

3.1.1 Particle Swarm Optimization

Particle swarm optimization (Kennedy and Eberhart, 1995; Sengupta et al., 2019) is an heuristic optimization algorithm. It is derivative–free, stochastic and population–based computational method, which demonstrated a high optimization potential in unfriendly non–convex, non–continuous spaces. The swarm consists of a set of particles, and each swarm member represents a potential solution of an optimization task. The particles are placed in the search space and move through such a space according to rules, which take into account each particle's personal knowledge and the global knowledge of the swarm. Every individual moves with its own velocity in the multidimensional search space, determines its own position and calculates its fitness using an objective function $f(x)$. On the basis of the fitness function the particles determine the best locations and the global best location.

The ordinary PSO algorithm begins by creating particles at initial locations, and assigning them initial velocities (Kennedy and Eberhart, 1995). Afterwards, it determines the value of the objective function at each particle location, as well as determines the best function value and the corresponding best location. It determines new velocities, based on the current velocity, the particles's individual best locations, and the best location of the entire swarm. In his work, a topology with the global best has been selected due to its faster convergence in comparison to neighborhood best one.

At the beginning of optimization, each particle is initialized with a random position and velocity. While seeking the best fitness, every individual i is attracted towards a position, which is affected by the best position $p^{(i)}$ found so far by itself and the global best position p_g found by the whole swarm. In every iteration k, each particle's velocity is first updated based on the particle's current velocity, the particle's local information and global information discovered by the entire population. Then, each particle's position is updated using the velocity. In the ordinary PSO, the position and velocity are calculated as follows:

$$v_{k+1}^{(i)} = w v_k^{(i)} + c_1 r_1^{(i)}(p_k^{(i)} - x_k^{(i)}) + c_2 r_2^{(i)}(p_{g,k} - x_k^{(i)}) \quad (2)$$

$$x_{k+1}^{(i)} = x_k^{(i)} + v_{k+1}^{(i)} \quad (3)$$

where w is the positive inertia weight, $v^{(i)}$ is the velocity of particle i, $r_1^{(i)}$ and $r_2^{(i)}$ are uniquely generated random numbers with the uniform distribution in the interval $[0.0, 1.0]$, c_1, c_2 are positive constants, $p^{(i)}$ is the best position that the particle i has found so far, p_g denotes the best position that was found by any member of the swarm.

Equation (2), which updates the particle velocity has three main components. The first component that is frequently referred to as inertia models the particle's tendency to keep it moving in the same direction it was moving previously. In fact it controls the exploration of the search space. The second component, called cognitive, attracts the particle towards the best position $p^{(i)}$ that was found formerly. The last component is referred to as social and it pulls the particle towards the best position p_g found by any particle. The fitness value that corresponds to $p^{(i)}$ is called lo-

cal best $p_{\text{best}}^{(i)}$, whereas the fitness value corresponding to p_g is defined as g_{best}. We implemented an asynchronous PSO, where in a given iteration, each particle updates and communicates its state to particles after its move to a new position, see pseudo-code of PSO algorithm. This means that the particles that will be updated in the same iteration can exploit the new best position immediately, instead of using the global best calculated in the previous iteration.

```
1   function PSO(X, iter)
2       for k = 0 to iter − 1 do
3           for each particle x^(i) ∈ X do
4               fx = f(x^(i))
5               if fx < p_best^(i) then
6                   p^(i) = x^(i)
7               if fx < g_best then
8                   p_g = x^(i)
9               v^(i) ← update velocity using (2)
10              x^(i) ← update position using (3)
11          endfor
12      endfor
13      return p_g, X
```

The discussed algorithm is typically employed for solving static optimization problems. The tracking of the 6D object pose can be accomplished by dynamic optimization and incorporating the temporal continuity information into normal PSO. This means that the 6D object pose can be estimated by a sequence of static PSO-based optimizations, followed by re-diversification of the particles to generate a search area containing the potential object poses that can arise in the next frame. The re-diversification of the particle i can be obtained on the basis of normal distribution concentrated around the best particle location p_g in time $t-1$, which can be expressed as: $x^{(i)} \leftarrow \mathcal{N}(p_g, \Sigma)$, where $x^{(i)}$ stands for particle's location in time t, Σ denotes the covariance matrix of the Gaussian distribution, whose diagonal elements are proportional to the expected movement.

3.1.2 Fitness Function

In recent years the PSO has been successfully applied in several model-based applications, including object detection (Ugolotti et al., 2013) and 3D pose refinement via rendering and texture-based matching (Zabulis et al., 2014). In 3D model-based tracking of 6D object pose the most computationally demanding operation is computing the objective function. A substantial speed-up of computation of the fitness function can be attained on modern GPU devices (Rymut et al., 2013). In this work, more attention is paid to reliable segmentation of the object of interest as well as tracking accuracy of 6D object pose, and thus the focus is on CPU implementation of fitness function to simplify design and evaluation of various algorithms.

In PSO-based approach every particle represents a hypothesis about likely 6D pose of the object. The fitness score of the particle is calculated by projecting the 3D model and then matching the projected model with the current image observations. Our fitness score depends on the ratio of overlap between rasterized 3D model in the hypothesized pose and the segmented object. The overlap ratio is the sum of the overlap degree from the object shape to the rasterized model and the overlap degree from the rasterized model to the object shape. The larger the overlap ratio is, the larger is the fitness value. Our fitness function includes also the normalized distance between the model's projected edges and the closest edges in the image. The discussed factor is calculated on the basis of the edge distance map, see Fig. 2.

The fitness score is calculated as follows:

$$f_s = \left(0.5 * \frac{P_{outside}}{P_{model}} + 0.5 * \frac{P_{empty}}{P_{seg}}\right)^{w_1} * \left(\frac{K}{P_K}\right)^{w_2} \quad (4)$$

where $P_{outside}$ stands for number of pixels projected from model that are outside of segmented object on image, see also Fig. 2 (left),

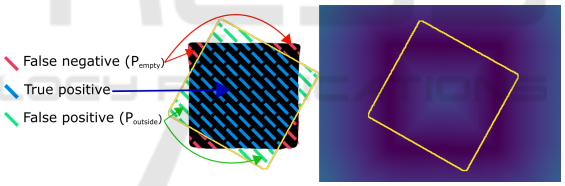

Figure 2: Fitness function.

P_{model} denotes number of pixels in the projected model, P_{empty} is number of pixels of segmented object on the image that are not covered by projected model, P_{seg} stands for number of pixels from segmented object, K is sum of L2 distance transform values from projected model's edges to object edges on the image, see Fig. 2 (right), P_K denotes number of edge pixels in the projected object outline, and $w_1 = 0.4$, $w_2 = 0.6$ are exponents that were determined experimentally.

3.2 6D Object Pose Tracking using PF

3.1.1 Particle Filtering

Particle filters permit estimating the state of partially observable controllable Markov chains, i.e. dynamical systems. For state estimation a probabilistic model of the system and a probabilistic observation model are needed. In this paper, the measurement at time t

is denoted by z_t, and x_t denotes the state of the system. Denoting by the superscript ^{t} all events leading up to time t, the measurement can be expressed as: $z^t = z_1, z_2, \ldots, z_t$. where the subscript $_t$ stands for an event at time t. Particle filters, like other Bayes filters, such as HMMs and Kalman filters, estimate the posterior distribution $p(x_t|z^t)$ of the dynamical system state conditioned on the data. This is done via the following recursive formula:

$$p(x_t|z^t) = \eta_t p(z_t|x_t) \int p(x_t|x_{t-1}) p(x_{t-1}|z^{t-1}) dx_{t-1} \quad (5)$$

where η_t is a normalization constant. To determine the posterior three probability distributions, which are referred as the probabilistic model of the dynamical system are needed: (i) a measurement model $p(z_t|x_t)$ describing the probability of measuring z_t when the system is in state x_t, (ii) a motion model $p(x_t|x_{t-1})$ characterizing the probability of transiting from the state x_{t-1} to the state x_t, (iii) an initial state distribution describing the initial system state.

A particle filter represents probability density function (PDF) of nonlinear/non-Gaussian system by a set of random samples. It is a common method to cope with non-Gaussian noises. The posterior PDF is approximated by a set of samples with associated weights: $X_t = \{x_t^{[i]}\}_{i=1,\ldots,m}$. Such particle set approximates the posterior $p(x_t|z^t)$. Initially, at time $t = 0$, the particles $x_0^{[i]}$ are drawn from the initial state distribution $p(x_0)$. Given X_{t-1}, the particle set X_t is then calculated recursively in the following manner:

```
1   function PF(X_{t-1})
2       set X_t = X_t^s = ∅
3       for j = 1 to m do
4           pick the j-th sample x_{t-1}^[j] ∈ X_{t-1}
5           draw x_t^[j] ~ p(x_t|x_{t-1}^[j])
6           set w_t^[j] = p(z_t|x_t^[j])
7           add ⟨x_t^[j], w_t^[j]⟩ to X_t^s
8       endfor
9       for i = 1 to m do
10          draw x_t^[i] from X_t^s with prob. ∝ to w_t^[i]
11          add x_t^[i] to X_t
12      endfor
13      return X_t
```

In lines 2 through 10 a new set of particles is generated on the basis of the estimate X_{t-1} of the previous time step through incorporating the probabilistic motion model, the probabilistic observation model and a resampling. Thus, the particle filter estimates recursively the particle set X_t on the basis of X_{t-1}. For large m the resulting weighted particle set is asymptotically distributed according to the desired posterior.

3.2.2 Motion and Observation Models

The probabilistic observation model is as follows: $p(z_t|x_t) = \exp(-f_s/\sigma_o^2)$, where σ_o is variance chosen experimentally. Particles are propagated according to a Gaussian distribution parameterized by σ_m^2, which was determined experimentally.

3.3 Proposed PF-PSO

The motivation is to improve the PSO by enabling the algorithm dealing with multi-modal distributions. At the beginning of each frame a PF is executed to determine the posterior distribution. Then, samples are clustered using k-means++ (Arthur and Vassilvitskii, 2007) algorithm, which applies a sequentially random selection strategy according to a squared distance from the closest center already selected. Then, a PSO with two sub-swarms consisting of samples assigned to the clusters is executed to find the modes in the probability distribution. The number of the iterations in the PSO is set to three. Afterwards, ten best particles are selected to form a sub-swarm, see lines #5-6 in below pseudo-code. Twenty iterations are executed by such a particle sub-swarm to find better particle positions. The best global position returned by the discussed sub-swarm is used in visualization of the best pose. Finally, an estimate of the probability distribution is calculated by replacing the particle positions determined by the PF with corresponding particle positions, which were selected to represent the modes in the probability distribution, see lines #5-6, and particles refined by the sub-swarm, see line #8. The initial probability distribution is updated by ten particles with better positions found by the PSO algorithms and ten particles with better positions found by the sub-swarm, see lines #9-11.

```
1   function select(n_best, X)
2       X_sorted = quicksort(X) using f()
3   return X_sorted[1...n_best]
```

```
1   X_t = PF(X_{t-1})
2   X_t^c1, X_t^c2 = k-means++(X_t)
3   ~, X_t^c1 = PSO(X_t^c1, 3)
4   ~, X_t^c2 = PSO(X_t^c2, 3)
5   X_t^c1_best = select(5, X_t^c1)
6   X_t^c2_best = select(5, X_t^c2)
7   X_t^best = X_t^c1_best ⋃ X_t^c2_best
8   g_best, X_t^best = PSO(X_t^best, 20)
9   substitute 5 x ∈ X_t with corresp. x ∈ X_t^c1_best
10  substitute 5 x ∈ X_t with corresp. x ∈ X_t^c2_best
11  substitute 10 x ∈ X_t with corresp. x ∈ X_t^best
12  return g_best, X_t
```

4 EXPERIMENTAL RESULTS

The experimental evaluation has been performed on five objects: box, bottle, duck, clamp and drill. The selected objects are almost entirely without any texture. 3D models of objects were created using the Kinect 2.5D RGB-D camera (Izadi et al., 2011) and SfM techniques. The OpenCV library (Bradski and Kaehler, 2013) has been used to calibrate the RGB camera. The ground truths of the object poses have been obtained by a turnable device, on which each object has been placed and then observed from three different camera views, see Fig. 3. For each camera view the objects were rotated in range $0^\circ \ldots 180^\circ$, and nineteen images were recorded. During object rotation, an image has been acquired every ten degrees with information about corresponding rotation angle. Each experiment has been repeated three times, the error metrics were calculated and then averaged.

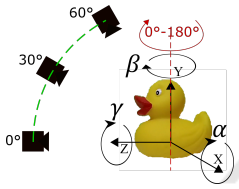

Figure 3: Experiments setup.

In order to learn the segmentation model the considered objects were observed by the camera from different views. For each object more than one hundred manual delineation of objects were done and then used to train neural networks for object segmentation. The models trained in such a way were then used to segment the objects from the background. Afterwards, a dataset consisting of RGB images with the corresponding ground truth data has been stored for the evaluation of algorithms for 6D object pose estimation and tracking.

4.1 Object Segmentation

A single U-Net neural network discussed in subsection 2 has been trained to segment the considered object using the set of manually segmented images. Table 1 presents the Dice scores achieved on the test subset of the dataset. The test subset contains thirty images for each object. As we can observe in Tab. 1, promising segmentation results were achieved for U-Net trained separately for each of the considered objects. Having on regard a better usefulness of a single U-Net for segmentation of all five objects, in the following subsections we present the experimental results achieved on the basis of the common U-Net for all five objects, see sample results on Fig 4.

Figure 4: Segmented objects.

Table 1: Dice scores on the test sub-dataset (ind. stands for individual U-Net for each object, whereas common means a common U-Net for all objects.

U-Net	box	bottle	duck	clamp	drill
ind.	0.985	0.964	0.974	0.971	0.928
common	0.946	0.972	0.978	0.968	0.965

4.2 Experimental Evaluation

We conducted experiments consisting in tracking the 6D object pose in sequences of RGB images. The objects were observed from different views. The experiments were performed on sequences of RGB images of size 288×512 acquired in advance and stored in mp4 files. For every camera view, the 6D pose of each object has been tracked on nineteen images. The objects were rotated about the vertical axis with poses changed about ten degrees. We evaluated the quality of 6-DoF object pose estimation using ADD score (Hinterstoisser et al., 2013). The 6D object pose estimate is considered valid if the ADD is smaller than ten percent of object's diameter. Table 2 presents scores [%] achieved by trackers on box [ADD $<$ 3.5 cm], bottle [ADD $<$ 2.6 cm], duck [ADD $<$ 1.5 cm], clamp [ADD $<$ 2.2 cm] and drill [ADD $<$ 3.0 cm]. The PF-PSO simple is a simplified version of the PF-PSO, where no k-means++ clustering has been executed, i.e. single PSO executing three iterations has been employed. As we can notice in Tab. 2, the PF-PSO achieves superior results on all objects except the duck. The poses of symmetrical objects were estimated with satisfactory accuracies with rotations about vertical symmetry axes in range $0^\circ \ldots 180^\circ$. The discussed results were achieved on the basis of one thousand calls of the objective function in single frame. The number of the calls of the fitness function in PF-PSO has been identical to number of the calls performed by the ordinary PF or the ordinary PSO. A multinomial resampling has been used in the PF. The state vector consists of 3D position and three Euler angles describing rotation about axes, see Fig. 3.

Table 3 presents ADD scores [%] achieved by PF-PSO in 6D object tracking in 0°, 30° and 60° camera views. As we can notice, the reason that the PF-PSO achieves worse results for the duck are bigger errors for 60° camera view.

Figure 5 presents plots of angle tracking errors over time, which were achieved by the algorithms for the bottle and the drill. As we can observe, the parti-

Table 2: Scores [%] achieved in 6D object tracking for box [ADD $<$ 3.5 cm], bottle [ADD $<$ 2.6 cm], duck [ADD $<$ 1.5 cm], clamp [ADD $<$ 2.2 cm] and drill [ADD $<$ 3.0 cm].

tracking score [%]	box	bottle	duck	clamp	drill
PF, angle 0...180°	0.345	0.222	0.246	0.281	0.450
PSO, angle 0...180°	0.807	0.877	0.719	0.579	0.813
PF-PSO simple, ang. 0...180°	0.889	0.912	0.351	0.520	0.614
PF-PSO, ang. 0...180°	0.901	0.947	0.673	0.643	0.889

Table 3: Scores [%] achieved by PF-PSO in 6D object tracking.

tracking score [%]	box	bottle	duck	clamp	drill
0° camera view, angle 0...90°	0.933	1.000	0.900	0.733	0.900
0° camera view, angle 0...180°	0.895	1.000	0.890	0.421	0.825
30° camera view, angle 0...90°	0.933	0.833	1.000	0.833	0.900
30° camera view, angle 0...180°	0.895	0.895	0.632	0.737	0.947
60° camera view, angle 0...90°	0.833	0.933	0.400	0.867	0.900
60° camera view, angle 0...180°	0.912	0.947	0.526	0.772	0.895
Average, angle 0...90°	0.900	0.922	0.767	0.811	0.900
Average, angle 0...180°	0.901	0.947	0.673	0.643	0.889

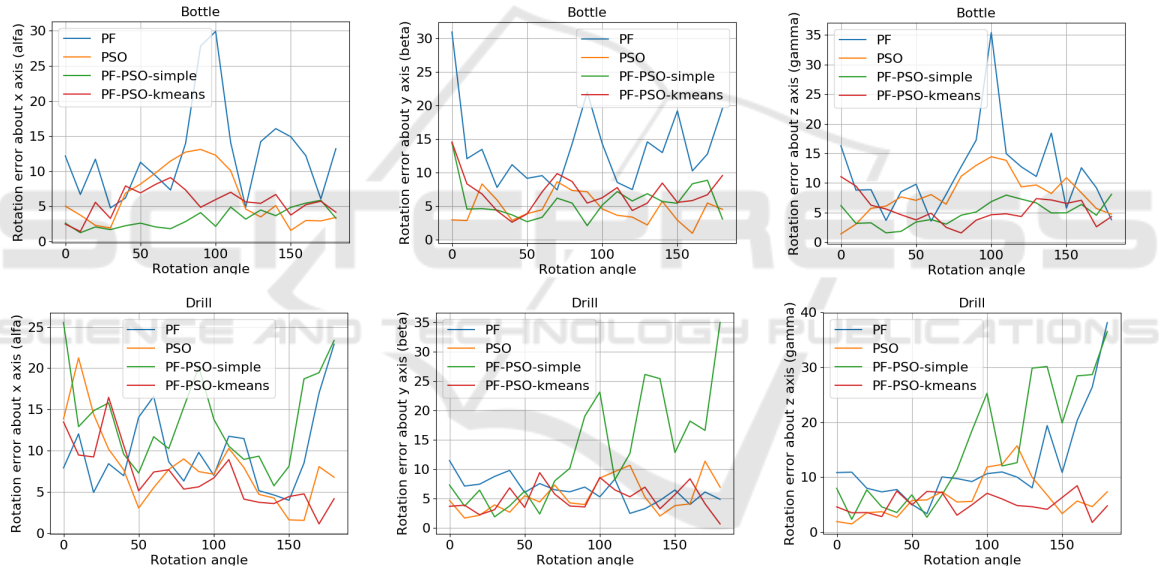

Figure 5: Pose errors over time for bottle and drill.

cle filter can continue tracking the object pose despite currently larger errors in several consecutive frames, see also errors about one hundred degrees for the bottle. The errors achieved by the PF-PSO for specific angle are smaller than 10°, except the error in a few frames for the drill, where in x axis, see Fig. 3, it is slightly bigger than ten degrees.

In experiments with pose tracking algorithms we noticed that the ordinary PSO can achieve promising results. However, in tracking the 6D pose of more complicated objects, the ordinary PSO can have difficulties with the tracking due to selecting in a given frame a mode, which in next frames will not allow to seek the optimal pose. On the other hand, the proposed PF-PSO contains particles representing worse mode in the given frame, which can be useful and can give better results in forthcoming frames.

The complete system for 6D pose estimation and tracking has been implemented in C/C++ and Python. The system runs on an ordinary PC with a GPU card. The images for training and evaluating the segmentation algorithm as well as extracted objects with corresponding ground-truth for evaluating the 6D object pose estimation and tracking are freely available for download at: http://home.agh.edu.pl/~majcher/src/visapp.

5 CONCLUSIONS

We have presented a 3D model based algorithm for 6D object pose tracking in sequence of RGB images. The object has been segmented using U-Net neural network. The 6D object pose estimation has been performed by particle filter combined with particle swarm optimization algorithm. Owing to clustering particles representing the probability distribution in the PF, the extracted modes were processed by PSO to represent them by a few representative particles in the refined probability distribution. In future work we are going to apply this algorithm for object manipulation by Franka Emika. The initialization of the tracking will be done on the basis of pose regression neural networks.

ACKNOWLEDGEMENTS

This work was supported by Polish National Science Center (NCN) under a research grant 2017/27/B/ST6/01743.

REFERENCES

Arthur, D. and Vassilvitskii, S. (2007). K-means++: The Advantages of Careful Seeding. In *Proc. ACM-SIAM Symp. on Discrete Algorithms*, pages 1027–1035.

Brachmann, E., Krull, A., Michel, F., Gumhold, S., Shotton, J., and Rother, C. (2014). Learning 6D object pose estimation using 3D object coordinates. In *ECCV*, pages 536–551. Springer.

Brachmann, E., Michel, F., Krull, A., Yang, M., Gumhold, S., and Rother, C. (2016). Uncertainty-Driven 6D Pose Estimation of Objects and Scenes from a Single RGB Image. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 3364–3372.

Bradski, G. and Kaehler, A. (2013). *Learning OpenCV: Computer Vision in C++ with the OpenCV Library*. O'Reilly Media, Inc., 2nd edition.

Deng, X., Mousavian, A., Xiang, Y., Xia, F., Bretl, T., and Fox, D. (2019). PoseRBPF: A Rao-Blackwellized Particle Filter for 6D Object Pose Tracking. In *Robotics: Science and Systems (RSS)*.

Doucet, A., Godsill, S., and Andrieu, C. (2000). On Sequential Monte Carlo Sampling Methods for Bayesian Filtering. *Statistics and Computing*, 10(3):197–208.

Fischler, M. A. and Bolles, R. C. (1981). Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Commun. ACM*, 24(6):381–395.

Hinterstoisser, S., Holzer, S., Cagniart, C., Ilic, S., Konolige, K., Navab, N., and Lepetit, V. (2011). Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes. In *Int. Conf. on Computer Vision*, pages 858–865.

Hinterstoisser, S., Lepetit, V., Ilic, S., Holzer, S., Bradski, G., Konolige, K., and Navab, N. (2013). 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.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML - vol. 37*, pages 448–456.

Izadi, S., Kim, D., Hilliges, O., Molyneaux, D., Newcombe, R., Kohli, P., Shotton, J., Hodges, S., Freeman, D., Davison, A., and Fitzgibbon, A. (2011). KinectFusion: Real-time 3D Reconstruction and Interaction Using a Moving Depth Camera. In *Proc. ACM Symp. on User Interface Soft. and Techn.*, pages 559–568.

Kennedy, J. and Eberhart, R. (1995). Particle Swarm Optimization. In *Proc. of IEEE Int. Conf. on Neural Networks*, pages 1942–1948. IEEE Press, Piscataway, NJ.

Lepetit, V., Pilet, J., and Fua, P. (2004). Point matching as a classification problem for fast and robust object pose estimation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, volume 2, pages II–II.

Marchand, E., Uchiyama, H., and Spindler, F. (2016). Pose estimation for augmented reality: A hands-on survey. *IEEE Trans. on Vis. and Comp. Graphics*, 22(12):2633–2651.

Rad, M. and Lepetit, V. (2017). BB8: A scalable, accurate, robust to partial occlusion method for predicting the 3D poses of challenging objects without using depth. In *IEEE Int. Conf. on Comp. Vision*, pages 3848–3856.

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-Net: Convolutional networks for biomedical image segmentation. In *MICCAI*, pages 234–241. Springer.

Rymut, B., Kwolek, B., and Krzeszowski, T. (2013). GPU-accelerated human motion tracking using particle filter combined with PSO. In *Adv. Concepts Intell. Vision Syst., LNCS, vol. 8192*, pages 426–437. Springer.

Sengupta, S., Basak, S., and Peters, R. A. (2019). Particle Swarm Optimization: A survey of historical and recent developments with hybridization perspectives. *Machine Learning and Knowl. Extr.*, 1(1):157–191.

Ugolotti, R., Nashed, Y. S. G., Mesejo, P., Ivekovič, v., Mussi, L., and Cagnoni, S. (2013). Particle Swarm Optimization and Differential Evolution for Model-based Object Detection. *Appl. Soft Comput.*, 13(6):3092–3105.

Vidal, J., Lin, C., and Martí, R. (2018). 6d pose estimation using an improved method based on point pair features. In *Int. Conf. on Control, Automation and Robotics (ICCAR)*, pages 405–409.

Xiang, Y., Schmidt, T., Narayanan, V., and Fox, D. (2018). PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes. In *Robotics: Science and Systems XIV (RSS)*.

Zabulis, X., Lourakis, M. I., and Stefanou, S. S. (2014). 3D pose refinement using rendering and texture-based matching. In *Computer Vision and Graphics*, pages 672–679. Springer.