# Image driven generation of pose hypotheses for 3D model-based tracking

Martim Brandão, Alexandre Bernardino, José Santos-Victor
ISR – Instituto Superior Técnico
1049-001 Lisboa - Portugal
mbrandao@isr.ist.utl.pt, alex@isr.ist.utl.pt, jasv@isr.ist.utl.pt

## Abstract

*Tracking an object's 3D position and orientation from a color image can been accomplished with particle filters if its color and shape properties are known. Unfortunately, initialization in particle filters is often manual or random, thus rendering the tracking recovery process slow or no longer autonomous. A method that uses image data to generate likely pose hypotheses for known objects is proposed. These generated pose hypotheses are then used to guide visual attention and computer resources in a "top-down" tracking system such as a particle filter: speeding up the tracking process and making it more robust to unpredictable movement.*

## 1. Introduction

Object pose estimation is a problem of great importance in applications such as robotics, video-surveillance or augmented reality. Often in these applications it is possible to know the object's model, for example its shape or color information, beforehand.

A class of methods used for 3D model-based object tracking is based on particle filters. These represent the distribution of an object's 3D pose as a set of weighted hypotheses (particles) [1]. Their advantage, to contrast with Kalman filtering, is the fact that the distribution of hypotheses is not restricted to Gaussian, and a random distribution is assumed [2]. Hypotheses are tested by explicitly projecting the object model in the image and comparing actual image pixel information. This is an example of a top-down approach to tracking: they depend not only of image information but on knowledge such as the object's typical kind of movement and expected location in space. In every frame a better fit for the object position is looked for, according to appropriate motion and noise distributions.

Initialization (or re-initialization), recovery after occlusions and dealing with unpredictable movement are, though, problems in particle (and basically all) top-down filters. When no a-priori information of an object's location is known, particles are scattered randomly in space – making the method difficult or slow to converge. Even if a high number of particles is available, it is difficult for the method to converge when it depends on hitting the right area of pose space by chance. In an attempt to address this problem, the method proposed in this paper focuses on quickly and intelligently choosing where to place particles and start or restart looking for the target, based on image information – to which we call a bottom-up approach to pose estimation.

A few works have been made that explore this idea for 2D tracking. [3] uses such an approach as a way to solve the problem of 2D tracking of people and their body parts: a bottom-up layer identifies candidates for body parts; while the top-down process searches for the whole body, constituted by several detected parts – assuming humans usually adopt certain poses. In [4], on the other hand, Adaboost is used as a bottom-up detector of objects (hockey players) and deals with the appearance of new players in the image. Then, a "mixture particle filter" (MPF) is used in the upper layer to track multiple players at the same time. This proposal's goal is most similar to the aforementioned ones, although we extend these paradigms to complete 6 degree-of-freedom pose estimation and tracking of 3D objects.

## 2. Bottom-up pose estimation

In this paper, pose hypotheses are generated in an image driven – bottom-up – manner and used on a 3D tracking process through particle filters. This generalizes the concept of visual attention in the sense that computational resources are allocated to areas of the whole pose space, 3D position and orientation. With the integration of both approaches, top-down's precision is kept, while both initialization speed and robustness in object reappearance is obtained from the bottom layer.

The proposed method is divided in 3 parts: segmentation, 3D localization and particle generation.

### 2.1. Segmentation

The first step in our method consists in segmenting the objects by color. We do so through color segmentation on the HSV color-space of the image, which was chosen in order to better achieve luminosity invariance. A color histogram of the object is known and so a Histogram Backprojection algorithm [5] is applied, building a map representing the likelihood of each pixel belonging to the object.

A scale-space of this backprojection map is created to better deal with the simultaneous presence of both small and large objects. Each scale is computed by filtering the map with a Gaussian function of different variance.

The segmentation algorithm used in each scale was obtained through a flood-fill method using local maxima of the map as seeds. Instead of the standard stop criteria,

Sauvola's binarization formula [6], usually used in document segmentation, was used to adapt the boundary detection threshold to the region's standard deviation.

After segmentation is completed in each scale, the results are condensed in a single binary map through an OR of all scales.

## 2.2. Localization

In order to estimate 3D pose from a segmented region, we compare its shape with trained ones. Since we use the perspective camera model, this training stage can be made independent of object position in the image. In run-time, if objects are not centered, we simulate a camera rotation to the centroid of the object. Training will therefore be made with the object centered in the image and a database is built that matches 2D shape to 3D orientation. The measured orientation can then be rectified using the equations for projecting rotated points in a pan-tilt camera [7]:

$$x_1 = \frac{ct.sp + cp.x_0 - st.sp.y_0}{ct.sp - sp.x_0 - st.cp.y_0}$$
$$y_1 = \frac{st + ct.y_0}{ct.cp - sp.x_0 - st.cp.y_0} \quad (2)$$

where ct,st,cp,sp represent cos(t), sen(t), cos(p), sen(p), respectively; (x0,y0) the initial point and (x1,y1) the point after a rotation of p and t degrees on a pan/tilt camera. We approximate the rotation of all pixels by the rotation of their average – the centroid. If we assume training is made with the object centered in the image then the initial point is equal to the origin (0,0) – and we compute the camera rotation that leads to moving that point to (x1,y1):

$$p = \arctan(x_1)$$
$$t = \arctan(y_1.\cos(p)) \quad (3)$$

where (x1,y1) are the region's normalized centroid coordinates and p,t are the pan and tilt rectification angles which, when applied after the measured rotation, give us the true orientation of the object. An orientation of the object should be defined as a single rotation sequence (as opposed to a rotation sequence followed by another of rectification). We therefore compute the angles of rotation Yaw, Pitch and Roll that define object orientation, from the final rotation matrix obtained from the composition of measured and rectification rotations.

Because perspective projection deforms the object as it moves away from the image center, a change of coordinates is made to center the region before computing its shape features. A homogeneous transformation with p and t as the rotation angles will produce such result, thus rendering orientation estimation independent of position.

After the object's final orientation coordinates are computed, we can compute depth (Z) from the area of the projection, defined as:

$$Area = \iint I(x,y)dxdy = \iint I(X,Y).\begin{vmatrix} \frac{f_x}{Z} & 0 \\ 0 & \frac{f_y}{Z} \end{vmatrix} dXdY \quad (4)$$

(X,Y) being the coordinates of the object's points in the world, I(X,Y) the object's shape represented on a binary image, and fx,fy intrinsic parameters of the perspective projection. We approximate that the object's points are all at the same depth, projecting on a plane which is parallel to the image. If so, depth (Z coordinate) can be computed from the relation of the segmented region's area and trained area and depth, which can both be stored in the database of projections.

$$Z = Z^{training}\sqrt{\frac{Area^{training}}{Area}} \quad (6)$$

Finally, X and Y are computed from the geometric center of the region, by assuming that the 3D geometric center of the object projects on the 2D center of the region given by (x,y)=(fx.X/Z, fy.Y/Z).

These approximations in position estimation introduce some errors, but allow us to generate good hypotheses that will be refined in the subsequent particle filtering stage.

To describe shape we use geometric moments, which hold point distribution information. Invariance to position and scale can easily be accomplished by using relative positions to the region's centroid and normalization to the area:

$$u_{pq} = \frac{\sum_x \sum_y (x-x_0)^p.(y-y_0)^q.I(x,y)}{M_{00}^{1+\frac{p+q}{2}}} \quad (7)$$

where upq is a normalized moment of order p+q and $M_{00}$ the area of the region.

Normalized moments should be used as shape descriptors using orders of 2 onwards. Despite this fact, the maximum order used should always be the 4th or higher. This is because for some symmetric shapes, such as squares for example, only moments of the 4th order can fully distinguish all orientations.

A normalized distance function between shapes, using moments, was then defined assuming a normal distribution:

$$d = \sum_{p,q} \frac{(\tilde{n}_{pq} - {}^i n_{pq})^2}{\mathrm{var}(n_{pq})} \quad (8)$$

$\tilde{n}_{pq}$ being the observed moment, ${}^i n_{pq}$ the moment of the ith hypothesis and $\mathrm{var}(n_{pq})$ the variance of the trained moment of order p+q. The most likely pose estimate of a segmented object will then be the one with minimum distance to the measured moments.

## 2.3. Particle generation

A likelihood function was defined from d as L=exp(-d/2). From this likelihood function, a cumulative distribution was computed, from where N particles can be

generated according to their likelihood by sampling the function in a uniform way.

## 3. Results

### 3.1. Localization error

The method was tested on perfect segmentations to evaluate its localization error alone. These were generated by projecting the object in 200 random poses, covering untrained orientations.

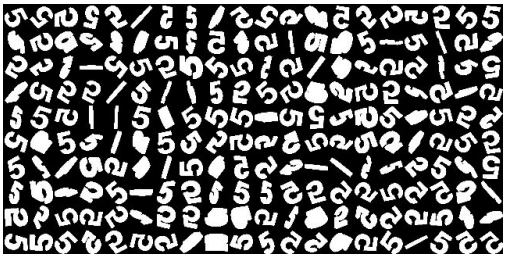

Figure 1. Set of 200 random poses generated to test localization error.

Given the top-down integration context, the error that tracking will be subject to is related to the least error particle. A quaternion representation was used to compute a single error value between the real and estimated orientations.

The average of the absolute angle error was then measured for different training resolutions and numbers of particles generated (N). Moments of order up to 7 were used as shape description features.

Table 1. Error of best particle, N=100

| Res | #Poses | X (cm) | Y (cm) | Z (cm) | Angle (º) |
|---|---|---|---|---|---|
| 20 | 3240 | 0,31 | 0,25 | 1,27 | 18,57 |
| 15 | 7488 | 0,29 | 0,21 | 0,94 | 12,6 |
| 10 | 24624 | 0,26 | 0,21 | 0,84 | 11,95 |
| 5 | 191808 | 0,26 | 0,22 | 0,69 | 10,7 |

From Table 1 we can confirm how lower orientation errors lead to lower error in the depth coordinate Z. Also, errors of 1cm in depth and lower than 0.5cm in X and Y can be achieved.

In the proposed method, particles are generated in an uniform way along the cumulative distribution function, thus leading to unfair generations for low values of N – lower than 100 – because not enough samples were selected from the set of hypothesis.

Table 2. Error of best particle, N=900

| Res | #Poses | X (cm) | Y (cm) | Z (cm) | Angle (º) |
|---|---|---|---|---|---|
| 20 | 3240 | 0,32 | 0,25 | 1,25 | 16,62 |
| 15 | 7488 | 0,26 | 0,2 | 0,7 | 8,19 |
| 10 | 24624 | 0,25 | 0,21 | 0,59 | 6,13 |
| 5 | 191808 | 0,24 | 0,2 | 0,43 | 4,2 |

On the other hand, a high number of generated particles such as 900 (see Table 2) allows the top-down tracking layer to expect starting errors of as low as 5º in orientation. From both Tables 1 and 2 we can see how expected errors can be as low as the training's resolution.

This kind of precision in visual attention gives greater flexibility in the management of resources for the top-down layer – which can make the whole process faster and more precise.

Computational time was also registered (see Table 3) on experiments made with a 2.67GHz Intel CPU and NVIDIA Quadro FX 580 graphics card.

Table 3. Computational time

| Res | #Poses | Time (ms) N=100 | Time (ms) N=500 | Time (ms) N=900 |
|---|---|---|---|---|
| 20 | 3240 | 21 | 31 | 39 |
| 15 | 7488 | 22 | 41 | 55 |
| 10 | 24624 | 33 | 42 | 51 |
| 5 | 191808 | 112 | 131 | 153 |

According to our experiments, an almost 30Hz real-time performance can be achieved. Also, being the proposed process easily parallelized, a real-time application would possible even for thin resolution, high particle requirements.

### 3.2. Pose estimation in real images

The whole method was tested on real images as well, for simple and complex objects, demonstrating the credibility of pose estimates with highest likelihood.

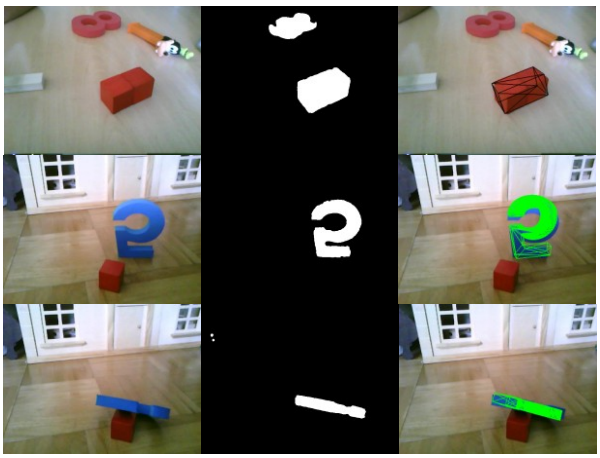

Figure 3. Two different objects, a box and “5”, their segmentation and highest likelihood pose. Objects were learned with a resolution of 15º.

The method, and likelihood function of the hypothesis in particular, behaves well for real images, with either simple or complex objects (see Figure 3). Note that in the first example no likely hypothesis exists on the number “8” since its shape is too different from the object being looked for (a box).

It is also possible to use this method for multiple object pose estimation (and, so, recognition). To accomplish that, the distance and likelihood measures must be made for all the known objects' databases, each pose now being basically assigned to an object. This way, generated particles will consist not only of pose but object identification. Two similar objects, a “5” and a “6”, were trained at 15º and tested on a real image – see Figure 4.

Figure 4. Result of pose estimation of a single object "5" (left) and multiple objects (right).

### 3.3. Simple integration with top-down tracker

Our pose estimation method was integrated with a top-down tracker [8] to evaluate the advantages of this joint approach to tracking.

A ball was put on a pendulum movement with an obstacle which hides it in the middle of the image. This makes the top-down layer lose track of the object every time it disappears – and proves how 3D visual attention (though only 3D position is used in this case) is important.

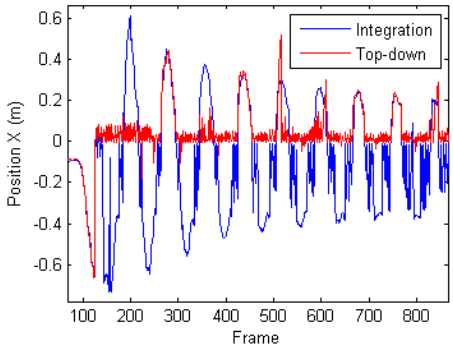


Figure 5. Estimate of x coordinate on the tracker before and after integration with our bottom-up particles. Pendulum movement with obstacle.

As we can see from Figure 5, when the object's movement is unpredictable by the motion model, or suffers occlusions, a top-down approach to tracking is not reliable on its own. After integration, recovery of the object's position is easily achieved after the object enters the scene – since bottom-up generated particles with a small enough starting error for the tracker to refine.

To better evaluate the advantage of combining both attention approaches we compute the average likelihood (Table 4), in the same sequence of images, for both top-down's and bottom-up's estimate when used by themselves; and also after integration.

Table 4. Average likelihood of methods

| Top-down | Bottom-up | Integration |
|---|---|---|
| 0,030 | 0,010 | 0,078 |

We can see how tracking improves with integration; and also that bottom-up's precision is low compared to top-down's when used by itself. Overall improvement in performance is obtained, be it in precision or robustness, after integration of the tracker with the proposed method.

## 4. Conclusions and future work

A method was introduced and tested, for generating pose hypotheses to be used on a 3D tracking paradigm that integrates both bottom-up and top-down approaches. Such joint approach was proven to take advantage of the upper layer's precision and lower layer's initialization speed and robustness to movement – obtaining a better performance than each of the layers independently. The proposed method's speed comes from the choice of shape descriptors and decoupling of orientation and position estimation problems.

We can from experiments conclude that the proposed method generates credible pose hypotheses with a tolerable error (less than 1cm on position and equal to resolution on orientation). It is also shown that real-time estimation is possible for a reasonable number of generated particles, obtaining an average error of 10º in orientation. Using graphics libraries such as OpenGL allows to generalize the localization method to any segmented rigid object – and it is also shown how credible pose hypotheses are generated for both complex and simple objects.

Generalizing pose estimation to part-based objects, just as in [3], would allow for general, multiple colored or textured objects – each part having its own characteristics.

## Acknowledgements

This work was partially funded by the EU Commission within the Seventh Framework Programme FP7, under grant agreement 248258 (First-MM) and grant agreement 231640 (HANDLE).

## References

[1] A. Doucet, J. de Freitas and N. Gordon, Sequential Monte Carlo Methods in Practice. Springer Verlag, New York, 2001.

[2] V. Lepetit and P. Fua. "Monocular model-based 3d tracking of rigid objects: A survey", Foundations and Trends in Computer Graphics and Vision, 1(1), 2005, pp1-89.

[3] D. Ramanan, D. A. Forsyth, and A. Zisserman. Tracking people by learning their appearance. IEEE PAMI, 29(1):65–81, Jan 2007.

[4] K. Okuma, A. Taleghani, N. de Freitas, J. Little, D. Lowe. A Boosted Particle Filter: Multi Target Detection and Tracking. ECCV, 2004.

[5] M.J. Swain and D.H. Ballard, Color Indexing, International Journal of Computer Vision, vol. 7:1, 1991

[6] J. Sauvola and M. Pietaksinen. "Adaptive document image binarization", Pattern Recognition, 33, 2000.

[7] B. Tworek et al., "Visual self-calibration of pan-tilt kinematic structures", Proc. ROBOTICA 2008, April, 2008.

[8] M. Taiana et al., "Sample-Based 3D Tracking of Colored Objects: A Flexible Architecture", BMVC2008, Leeds, UK, 2008.