Generating pose hypotheses for 3D tracking: a bottom-up approach

Martim Brandão, *Instituto Superior Técnico*

***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 a priori. 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 existing object information to better decide on where to automatically start or recover the tracking process is proposed. Each 3D pose of an object is observed in the image as a 2D shape and so training is made to infer pose from image information. The object is first segmented through color, then shape description is made using geometric moments and finally a learning stage maps 2D shapes to 3D poses with an associated likelihood measure. These generated pose hypotheses can then be 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 movements.**

***Index Terms*—3D pose estimation, bottom-up attention, model-based 3D tracking, particle filters.**

I. 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 [1]. These represent the distribution of an object's 3D pose as a set of weighted hypotheses (particles). Particle filters, by maintaining several hypotheses over time, have increased robustness [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 color, shape, 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) is, though, a problem 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 whether you use few or many particles respectively. 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, in a bottom-up manner.

Bottom-up approaches try to estimate object pose solely from image information such as distribution of color or intensity. Its analysis is made from the lowest level of image interpretation: pixel neighborhoods. They do not need to explicitly project the object in the image, thus gaining speed though at the cost of less precision.

Methods based on sparse sets of hypotheses, such as particle filters, allow us to elegantly control the computational cost at each stage by limiting the number of considered hypotheses. It is the right allocation of these resources to the most promising areas of the images, through a visual attention method, that will make visual searching systems perform greatly both in speed and precision.

In this paper, pose hypotheses are generated in a 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 not only to positions in the image but to areas of the whole space of 3D position and orientation. With the integration of both approaches, top-down's precision is kept, while both initialization speed and robustness to object reappearance is obtained from the bottom layer.

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

II. Related work

A. Particle filters

Model-based tracking tries to compute the state density function of a target given information on its model. Both Kalman and Particle filtering maintain a set of state hypotheses, even if the current state's representation is a single estimate (in the Kalman filter's case). This approach to tracking has greater robustness [2] and makes pose prediction in the next frame easier. Kalman filters are restricted in the distribution of hypotheses, though, which may not be acceptable in many situations.

In particle filters, on the other hand, the distribution of particles is not restricted, and a random, non-Gaussian distribution is assumed. A state density function is kept and updated at each step, which will guide the re-sampling process

of particles towards conversion. An object's pose (position and orientation, although velocity and higher order derivatives could be used) is at each time t represented as a set of particles $V_t(i)$, with observations $O_{1:t}$, $i \epsilon \{1...N\}$, N being the number of particles. In a Monte Carlo approach [3], these particles are associated to a weight $w_t(i)$.

On the re-sampling stage, samples are taken from the $V_{t-1}(i)$ distribution according to the weights $w_{t-1}(i)$ and its pose perturbed with a motion model. This way the a priori density of the next frame is computed. In that frame the weights will be updated according to a likelihood measure of the particles – after projecting them in the image – to compute the a posteriori density.

Limitations of top-down tracking and particle filters in particular include their time dependence (bad decisions in the past affect the future, dealing with unpredictable movement) and the amount of hypotheses that are limited by computer resources. Furthermore, initialization – and recovering from occlusions – is another problem. Because the search is "blind", particles in these situations are scattered randomly in pose space. 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.

B. 3D Tracking and pose estimation

Examples of top-down 3D tracking using particle filters include [4], where hypotheses are tested by explicitly computing the object's edges on the computer's graphics processing unit (GPU). A CAD model of the object is known and a "pixel shader" program is used to compute likelihood in the GPU itself. [1] uses color information near the object's borders to evaluate likelihood. It does so in an intelligent way, according to resemblance of color in sampled points inside and outside of each hypothesis' edges – thus reducing computational time.

Regarding pose estimation, [5] does so in all 6DoF of the 3D space using two cameras. The object is first segmented through color by establishing simple threshold levels; and then orientation is estimated by comparison with a database of object projections. These projections are stored as images in the database and then reduced in size by applying PCA. The database is built with the object at a fixed position, being the orientation rectified later on according to the object's real position obtained through stereo matching. It differentiates from our proposed method for being stereo instead of monocular and looking for a single best estimate instead of a set of hypotheses. Problems include the possibly excessive size of the database and dependence on the high quality of the camera. [6] estimates pose as well, this time in a monocular method, using an iterative process of 2D segmentation and 3D pose estimation. Segmentation is region-based and done using active contours and an energy function to be minimized. The object's pose parameters are then changed, and the object re-projected, according to the segmentation's energy function change. The two processes are thus combined in a segmentation-estimation loop, giving high robustness to noise and occlusions. Unfortunately it is not suited to real-time pose estimation since a loop of energy minimizing and re-projection of the object is not fast to compute.

The integration of both approaches, top-down and bottom-up, is proposed for example in [7]. Here, interest point matching is used together with optical flow estimation along edges on a 3D Kalman filter. [8] also joins the two worlds 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 by detecting rectangles through an edge template. The top-down process searches for the whole body, constituted by several detected rectangles – assuming humans usually adopt certain poses, and the body parts different relative poses between each other. [9] also uses a joint approach to 2D tracking. 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. Adaboost is trained to detect players and integrated with MPF to compute its distribution. This proposal is most similar to the one presented in this paper, although we herein try to address the same problem in 3D space.

III. Bottom-up pose estimation

A. 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 [10] 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. Flood-fill adds neighbor pixels to the segmented region, starting on a certain seed point and as long as a certain criteria is met. Instead of the standard stop criteria, Sauvola's binarization formula [11], usually used in document segmentation, was used to adapt the boundary detection threshold to the region's standard deviation:

$$T(x,y) = p(x,y).\left(1 + k.\left(\frac{s(x,y)}{R} - 1\right)\right) \quad (1)$$

where p(x,y) and s(x,y) represent, respectively, map intensity at (x,y) and current standard deviation of the grown region – which is updated every time a new pixel is added to the region. R is the dynamic range of s and k is a parameter here

established as 0.5.

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

B. 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 [12]:

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

where ct,st,cp,sp represent cos(t), sen(t), cos(p), sen(p), respectively; (x_0,y_0) the initial point and (x_1,y_1) 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 (x_1,y_1):

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

where (x_1,y_1) 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. A loop of this transformation until convergence is used, as the object's centroid is not guaranteed to move to the image center on the first iteration. In practice, though, a second iteration is rarely necessary as it introduces a change of less than 1º in the initial (p,t) values.

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 f_x,f_y 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, the area of the projection at depth Z is given by:

$$Area = \frac{f_x.f_y}{Z^2}.\iint I(X,Y)dXdY \quad (5)$$

Depth (Z coordinate) can as such be computed from the relation of the segmented region's area and trained area and depth.

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

The database of orientations and corresponding shape features should hence record the area of the trained projection as well.

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 $(x,y)=(f_x.X/Z,\ f_y.Y/Z)$. This introduces some errors but allows 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 u_{pq} 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 i^{th} 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.

C. 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.

IV. Integrating with a particle filter

To integrate the proposed method with an existing top-down tracker, [1] was used, which is implemented as a module on the robotcub humanoid robot iCub[13].

The tracker was adapted to accept incoming particles and store them as the last B particles of its budget N. This space will be reserved to particles generated on the bottom-up approach, and particle filter's re-sampling will be made based on the likelihood of all N particles, generating N-B new particles for the next frame according to measured likelihood and motion model.

For this integration to work in real-time, processing was distributed – each module (top-down and bottom-up) running on a separate CPU.

V. Results

A. 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 (see Fig. 1).

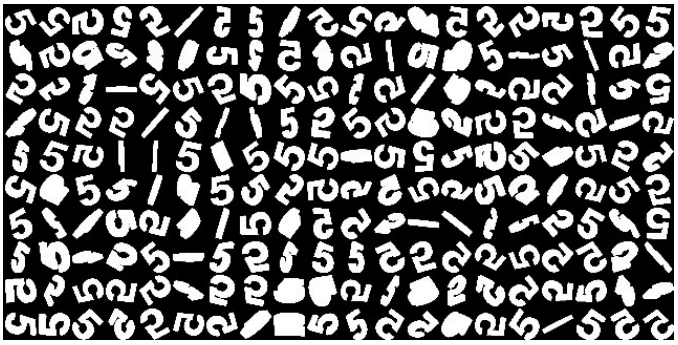

Fig. 1. Set of 200 random poses generated to test localization error.

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. Moments of order up to 7 were used as shape description features.

TABLE I
ERROR OF BEST PARTICLE, N=100

Res. (º)	Num. 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 I 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 are selected from the set of hypothesis. Generally, at N=100, an average error of 10º in orientation can be achieved.

TABLE II
ERROR OF BEST PARTICLE, N=900

Res. (º)	Num. 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 II) allows a top-down tracking layer to expect starting errors of as low as 5º in orientation. This kind of precision in visual attention, for N in [100,900], gives greater flexibility in the management of resources for the top-down layer – which can make the whole process faster and more precise. We study the influence of N in the obtained orientation error in Fig. 2.

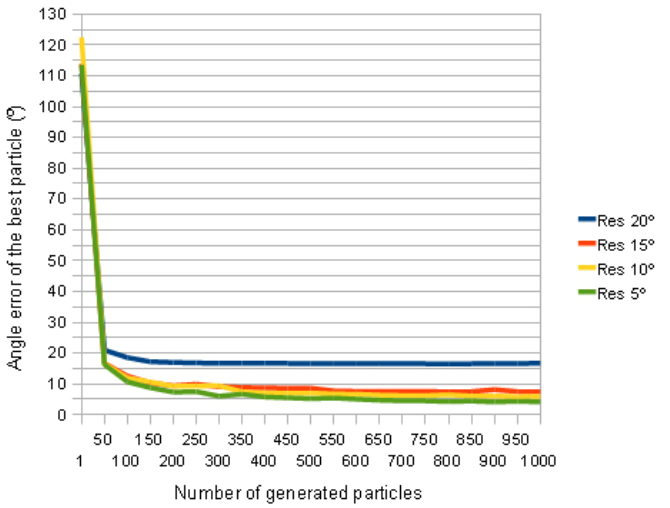

Fig. 2. Orientation error of the least error particle, for N generated particles and different training resolutions.

Computational time was also registered, to better evaluate the method's speed and its relation with the number of generated particles. Experiments were made on a computer with a 2.67GHz Intel CPU and NVIDIA Quadro FX 580 graphics card – see Table III.

TABLE III
COMPUTATIONAL TIME

Res. (º)	Num. 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 the application, the training resolution and number of generated particles need to be wisely chosen, as one can conclude from the previous tables. To achieve real-time performance, N shouldn't be higher than 500 particles. Errors of around 10º in orientation can be achieved in these cases. For less restrictive applications, expected errors can be as low as the training's resolution.

B. *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 (see Fig. 3).

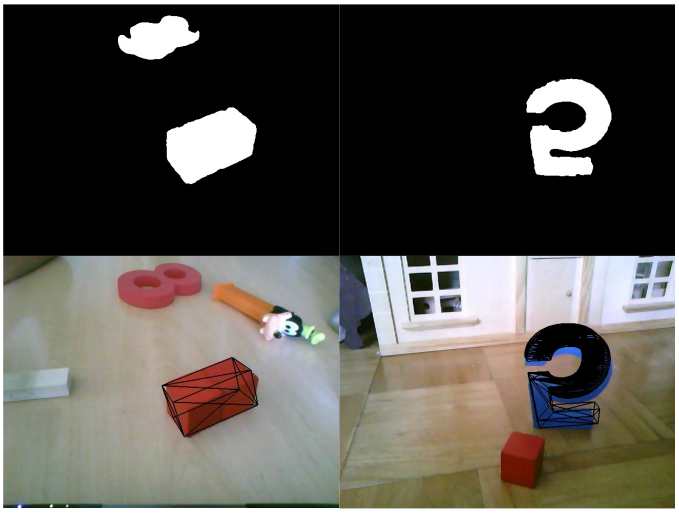

Fig. 3. Two different objects, a box and "5", their segmentation and highest likelihood pose hypothesis. Objects were learned with a resolution of 15º.

The method, and likelihood function of the hypothesis in particular, behaves well for real images, with either simple or complex objects. 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 know 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 Fig. 4).

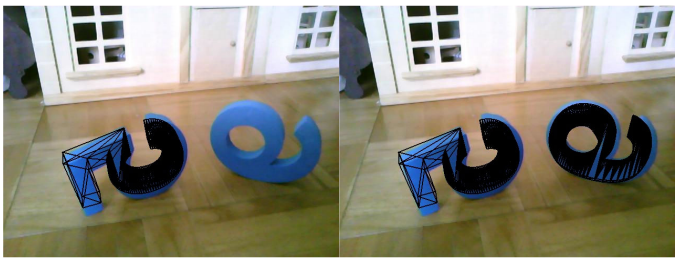

Fig. 4. Result of pose estimation of a single object (left) and multiple objects (right)

Note that in the first example, where only the object "5" exists in the database, no likely hypothesis exists on the number "6" since its shape is too different. Recognition works in this example when using two objects, since the highest likelihood object and pose of each segmented region is correctly computed.

C. *Integration with a top-down tracker*

Our pose estimation method was integrated with a top-down tracker [1] 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 (although only in 3 degrees of freedom) is important in this case.

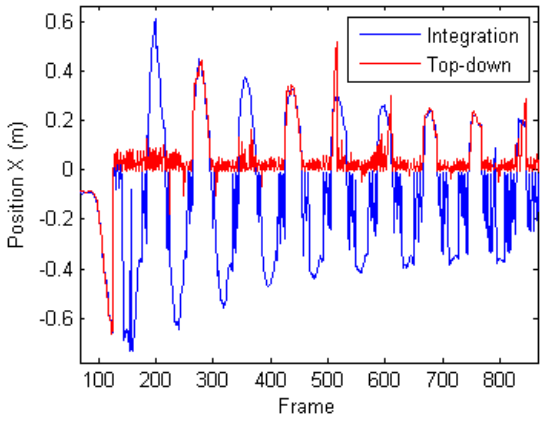

Fig. 5. Estimate of x coordinate on the tracker before and after integration with our bottom-up pose estimation method. The object is put on a pendulum movement with an obstacle in the middle of the image.

As we can see from Fig. 5, when the object's movement is no longer well represented by its motion model, is unpredictable, 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 the generated particles in the bottom-up layer had a small enough starting error for the tracker to refine.

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

TABLE IV
AVERAGE LIKELIHOOD OF THE ISOLATED METHODS AND AFTER INTEGRATION

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

We can see how tracking gains about 250% in the average likelihood of its estimate after integration; and also that bottom-up's precision is low compared to top-down's when used by itself. Even though its precision is lower, it provides an upper layer with good enough starting points for a local and refining search to be made.

Overall performance, be it in precision or robustness is obtained in tracking after integration with the proposed method.

VI. 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 speed and lower layer's robustness – 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.

Since the proposed pose estimation method relies on full segmentation of the object, methods to produce these segmentation for textured objects would be interesting to generalize this work. Also, generalizing pose estimation to part-based objects would be of most interest. Just as in [8], each part of the object would contribute to a certain area of the pose space, the intersection of which would be the final pose estimation of the object. This allows for general, multiple colored or textured objects – each part having its own characteristics.

REFERENCES

[1] M. Taiana, J. Nascimento, J. Gaspar and A. Bernardino. “Sample-Based 3D Tracking of Colored Objects: A Flexible Architecture”, BMVC2008, Leeds, UK, 2008.
[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] A. Doucet, J. de Freitas and N. Gordon, Sequential Monte Carlo Methods in Practice. Springer Verlag, New York, 2001.
[4] G. Klein and D. Murray. Full-3d edge tracking with a particle filter. In Proc. of BMVC 2006, page III:1119, Edinburgh, Scotland, 2006.
[5] P. Azad, T. Asfour, and R. Dillmann, Combining Appearance-based and Model-based Methods for Real-Time Object Recognition and 6D Localization, in International Conference on Intelligent Robots and Systems (IROS), Beijing, China, 2006.
[6] S. Dambreville, A. Yezzi, R. Sandhu, and A. Tannenbaum. Robust 3d pose estimation and efficient 2d region-based segmentation from a 3d shape prior. In Proc. 10th European Conference on Computer Vision, LNCS. Springer, 2008.
[7] V. Kyrki and D. Kragic. Integration of model-based and model-free cues for visual object tracking in 3d. In IEEE Int. Conf. on Robotics and Automation, ICRA’05, pages 1566–1572, Barcelona, Spain, April 2005.
[8] D. Ramanan, D. A. Forsyth, and A. Zisserman. Tracking people by learning their appearance. IEEE PAMI, 29(1):65–81, Jan 2007.
[9] K. Okuma, A. Taleghani, N. de Freitas, J. Little, D. Lowe. A Boosted Particle Filter: Multi Target Detection and Tracking. ECCV, 2004.
[10] M.J. Swain and D.H. Ballard, Color Indexing, International Journal of Computer Vision, vol. 7:1, 1991
[11] J. Sauvola and M. Pietaksinen. “Adaptive document image binarization”, Pattern Recognition, 33, 2000.
[12] B. Tworek, A. Bernardino, and J. Santos-Victor, “Visual self-calibration of pan-tilt kinematic structures”, Proc. ROBOTICA 2008, April, 2008.
[13] G. Metta, L. Natale, F. Nori, G. Sandini, D. Vernon, L. Fadiga, C. von Hofsten, J. Santos-Victor, A. Bernardino, L. Montesano. The iCub Humanoid Robot: An Open-Systems Platform for Research. Special Issue of Neural Networks on Social Cognition: From Babies to Robots, Accepted 2010