2007 IEEE International Conference on
Robotics and Automation
Roma, Italy, 10-14 April 2007

Multi Sensor Fusion in Robot Assembly Using Particle Filters

U. Thomas, S. Molkenstruck, R. Iser, and F. M. Wahl
Institute for Robotics and Process Control, Technical University of Braunschweig, Germany

Abstract— **In this paper, we present a new method for sensor fusion in robot assembly. In our approach, model information can be derived automatically from CAD-data. We introduce Force Torque Maps, which are either computed automatically exploiting modern graphical processors or are measured by scanning forces and torques during contact motions. Subsequently, Force Torque Maps are applied as model information during execution of real assembly tasks. Also, computer vision is included by comparing relative poses of features in virtual images with their real relative poses given from measured images. For fusion of these two (or more) different sensors we suggest to use particle filters. Experiments with variations of peg in hole tasks in a real work cell demonstrate our new approach to be very useful for the whole process chain from planning to execution.**

I. INTRODUCTION

A long term aim in robot programming for assembly is the automation of the complete process chain, i.e. from planning to execution. One challenge is to provide solutions which are able to deal with position uncertainties in robot assembly. If such systems exist, robot programming becomes less cost intensive. CAD-data of objects are usually available and can be used for planning purposes. Due to tolerances in models, gripped objects and pose tolerances in workspace, sensor integration is indispensable. Many sophisticated implementations of the execution for complex assembly tasks use different sensors, but the control algorithms as well as the assembly strategy are still designed by experts, e.g. [1]. A main goal is to automate the programming process, so that the expensive task of programming such applications will be simplified or even automated. Our approach aims at two goals:

- Planning and executing the assembly task automatically without any user interaction
- Simultaneously integrating vision and force feedback for correcting uncertain poses in 3d.

Our approach uses particle filters – more precisely the condensation algorithm [2] – for the autonomous execution of various assembly tasks. During the planning phase the CAD-data are known, so that we can generate Force Torque Maps from the CAD-data (section II). Also, during the planning phase, it is possible to extract geometrical features from the CAD-data. These two model representations - one for force feedback and one for vision feedback - can be used, while hypothetical 3d poses, represented as particles, will be evaluated. In mobile robotics, particle filters have shown to be a powerful tool for robot localization [3]. Why should this strategy not work for robot assembly? In [4], [5], applications of particle filters for sensor guided assembly have been demonstrated. The configuration space (c-space) obstacles are generated by moving the gripped object into contact with its environment and interpreting the resulting position sensor information only. A map is obtained from collected contacts, which represents all suitable configurations. During execution the particle filter guides the robot for successful assembling. In contrast to this, the approach provided here generates so-called Force Torque Maps (FT-Maps) from CAD-data prior to execution. Hence, no time and cost consuming scanning of the environment with the robot is necessary. The FT-Maps are used as model description to evaluate each particle (relative pose in 3d). In order to obtain FT-Maps efficiently prior to execution, we have implemented a graphical processor unit based algorithm. The cost intensive on-line generation of the c-space maps as shown in [4] are no more necessary. The generation of FT-Maps is a computational expensive task, but no robot is necessary to perform it.

Other solutions of compliant motion programming are shown in [6]. There contact formation graphs are necessary to configure hybrid position/velocity and force controlled motions. In [7] task templates are used to generate nets of skill primitives, known from [8]. In [9] the particle filter is applied for sensor fusion in contour following tasks. The particle filter for contact formation estimation has been suggested in [10]. A solution to integrate vision and force feedback, with a camera attached to a robot hand is demonstrated in [11].

Our work suggests to solve the automated assembly programming problem by usage of particle filters, Force Torque Maps and vision processing. The main benefit of our method is, that model information for the particle filter can automatically be derived from CAD-data prior to execution, so that no further human interaction is necessary. The next section describes the algorithm to generate FT-Maps from CAD-data. The third section shows how vision can be used for evaluating hypothetical poses in 3d. The fourth section describes our implementation of the particle filter in detail. The last section demonstrates our experiments on variations of peg in hole assembly tasks.

II. Force Torque Maps for Particle Evaluation

FT-Maps are applied as model information for the particle filter. The main advantage is, that FT-Maps can be generated prior to execution. The algorithm is implemented using a graphical processor unit (GPU). With GPUs, distance computation can be obtained very fast. In this section, the definition of FT-Maps is given first. Then follows a description of the

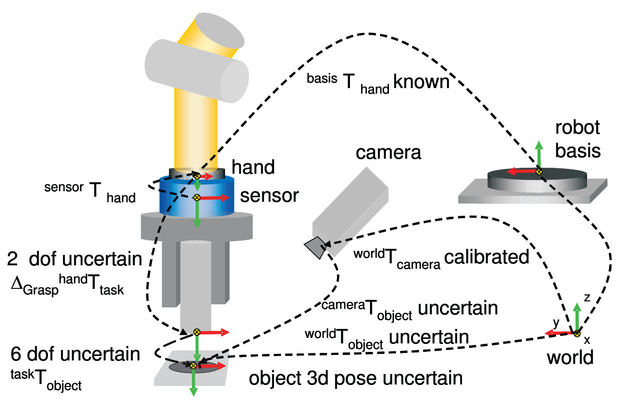

Fig. 1. Object poses and uncertainties in the assembly work cell

algorithm applied for obtaining these maps and the last part shows some examples of FT-Maps for variations of peg-in-hole assembly tasks.

A. Definition of Force Torque Maps

In the following, we denote ${}^{task}\vec{w} := (f_x, f_y, f_z, t_x, t_y, t_z)^T$ as wrench vector of forces and torques w.r.t. a given task frame, which arise in some contact situation between a gripped object and an object placed fixed in the robot work cell. The force torque sensor is attached to the robot hand, so that its z-axis is aligned to the robot hand's z-axis, as can be seen in Fig. 1. With the vector $\vec{r} \in \Re^6$ we denote any possible *relative* pose (position and orientation) in 3d. Whereas, in general, a particle x can be described as a list of relative poses $\vec{r}$; each element represents one transformation in the work cell. Thus, a particle is defined by:

$$x_i \in X | x_i := < \vec{r}_1, \ldots, \vec{r}_n > \text{ with } \vec{r}_j \in \Re^6 \tag{1}$$

In our examples n is set to 2. The first relative pose $\vec{r}_1$ represents the inaccurately gripped object and the other relative pose $\vec{r}_2$ describes the unknown transformation from the task frame to the object frame, assuming the task frame is fixed on the peg's tip. If the part is assembled at its goal position, the transformation from the task frame to the object frame is the identity. Hence, in general cases, particles are 12 dimensional, but in most cases less dimensions are necessary to describe possible uncertainties. In the example of Fig. 1, the first transformation ${}^{Hand}T_{Task}$ contains two translational DoFs and the second transformation ${}^{Task}T_{Object\_B}$ contains four DoFs (two translational and two rotational).

Before we define FT-Maps we introduce the discrete tolerance space Ω as follows:

Definition 2.1.1*: The discrete tolerance space Ω is a grid, where each cell $\omega_i \in \Omega$ corresponds to a n-tuple of uncertain relative poses $\vec{r}_1, \ldots, \vec{r}_n$. Each relative pose $\vec{r}_j$ is bounded by two parameters $size_{trans,j}$, $size_{rot,j}$ and $\vec{r}_j$ satisfies the following equations: $r(x)_j^2 + r(y)_j^2 + r(z)_j^2 \leq size_{trans,j}^2$; $r(rx)_j^2 + r(ry)_j^2 + r(rz)_j^2 \leq size_{rot,j}^2$ respectively. Also for each DoF the interval size is set to $\delta_{trans,j}$ and δ_{rot_j}.*

Hence, with this definition of the discrete tolerance space, we can define FT-Maps:

Definition 2.1.2*: A Force Torque Map is a function $f : \omega \rightarrow \vec{w}$, which maps each value $\omega_i \in \Omega$ to a valid wrench vector w.r.t robot's task frame ${}^{task}\vec{w}$ (sensor values) with the assumption of imposed forces. These forces are imposed due to a robot motion along the approach vector ${}^{task}\vec{a}$.*

In practice, the defined imposed forces are obtained by using hybrid position/velocity and force controlled robot systems [12], [13]. For the off-line computation of such FT-Maps the following model is applied: Assuming the contact point $\vec{p}$ is known from CAD-data, and the robot moves the object A along $\vec{a}$, which is defined w.r.t. the task frame. Then, we can compute expected forces and torques by applying the following rule:

$$\vec{f}_{imposed} = -\mu \cdot \vec{f}_{normal} - \vec{f}_{friction} \tag{2}$$

with μ as the friction coefficient between two materials. $\vec{f}_{normal}$ is perpendicular to the contact surface of object B. Assuming reactio equals actio yields $\vec{f}_{sensor} = -\vec{f}_{imposed}$ neglecting robot stiffness and robot dynamics. This model is not exact, but it serves precisely enough as model information for the particle filter. One can imagine, that computation of possible sensor values for all $\omega_i \in \Omega$ results in a computational complex tasks and demands for an efficient algorithm. Thus, we exploit a graphical processor unit (GPU) for obtaining FT-Maps.

B. A GPU-Based Algorithm

Usually in robot assembly CAD-data are available, hence we use triangle meshes as input for the FT-Map computation. For all possible geometrical poses within Ω , we move the gripped object A as well as the object B to the poses according to each discrete value $\omega_i \in \Omega$. First of all, let us define the minimal distance between object A and object B along the vector $\vec{a}$ as

$$d_{\vec{a}} = \min_{\vec{p}_a \in A, \vec{p}_b \in B} \{ |(\vec{p}_b - \vec{p}_a) \cdot \vec{a}| \} \, .$$

All pairs of points $(\vec{p}'_a, \vec{p}'_b)$ with $|(\vec{p}'_b - \vec{p}'_a) \cdot \vec{a}| = d_{\vec{a}}$ are called contact pairs and need to be computed. Many algorithms for collision detection using hull hierarchies have been introduced during the last decades. Some of these algorithms could be adopted for our task, but due to the size of our discrete tolerance space Ω and the circumstance that we always need *all contacts*, high demands have to be put on efficiency of such an algorithm. Thus, we have employed a GPU for obtaining all pairs of contact.

1) Computing the minimal distance and all contact pairs: The method introduced here for obtaining all pairs of contact and its contact normal vectors uses the z-buffer and the color buffer of graphics hardware. First, we need to compute the vector aligned bounding box (*VABB*), so that it is aligned to the approach direction $\vec{a}$ for the active object A and for the passive object B, respectively. Next, we minimize both *VABB*, so that projection of the two *VABB* on any to $\vec{a}$ perpendicular plane has equal size. The number of needed pixels is obtained by the length of the *VABB* and given resolution δ from the applied tolerance space Ω. Subsequently, the number of needed rendering steps (images) is calculated by using the number of necessary pixels and the maximal possible resolution of the graphics hardware. Subsequently the camera is placed in the nearest corner of the left bottom of the $VAAB_A$. The camera z-axis is aligned with the direction specified by the approach vector $\vec{a}$. We used an orthogonal projection and computed the z-buffer of the object B. Also each triangle is given a different color, so that we obtain the surface normal, also. We repeat the same procedure for the object A, where the z-axis of the camera points opposite to the approach vector. The minimal distance for each pixel is computed by the sum of both depth buffer values. Searching all 2d points, for which the differences of their depth buffers are smaller than 0 results in all contact points in 3d, due to known camera parameters. After computing all pairs of contact, we can compute resulting forces and torques as follows:

2) Computing forces and torques from all contact pairs: The algorithm projects all 3d contact points $\vec{p}_{A,B}$ to the plane perpendicular to $\vec{a}$. Now, the convex hull is constructed of all these projected contact pairs. The point of interest K is the point with smallest distance to the intersection point P and the convex hull. The point P is obtained by projecting the sensor origin among $\vec{a}$ into the plane (see Fig. 2). If the point P is inside the convex hull, we obtain expected forces by: $\vec{f}_{sensor} = -\vec{f}_{imposed}$ and as torques we set $\vec{t}_{sensor} = \vec{0}$. If the point P is outside the convex hull, the minimal distance between the convex hull and the point P can be denoted by $\vec{PK}$ and influences the torques which are obtained by $\vec{t}_{sensor} = \vec{PK} \times \vec{F}_{imposed}$. Fig. 2 illustrates our algorithm.

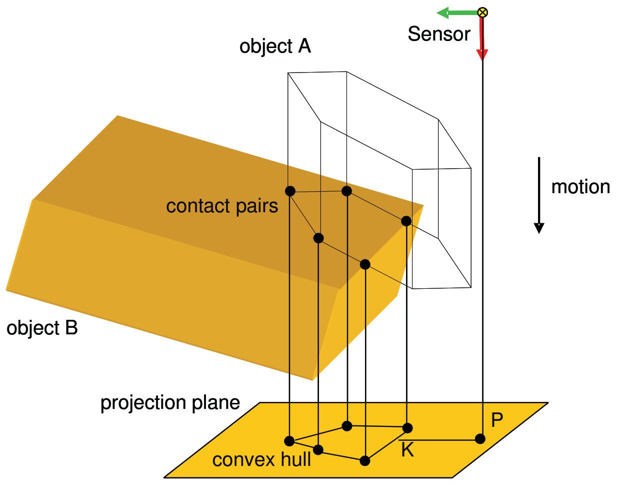

Fig. 2. Algorithm to compute the lever causing torques from a given imposing force

C. Some Examples of Force Torque Maps

We have computed FT-Maps for various assembly tasks. Fig. 3 and Fig. 4 show some FTM-Maps for variations of peg in hole tasks. The computation time took from 2 minutes up to 20 hours, where maximal 3.5 million configurations have been computed. Usually we set $size_{trans}$ to 5 denoting a translational uncertainty of 5 mm in x-direction and y-direction respectively. The rotational uncertainty are given within $[-2, 2]$ degrees around the x-axis and y-axis. Fig. 4 shows some layers of the FT-Maps for two relative poses of the active object. The map on the left side corresponds to the relative pose of peg $\vec{r}_1 = (3,3)^T$ and visualizes t_x according to $\vec{r}_2 = (p_x, p_y, 0°, 0°)^T$. The map on the right side corresponds to a configuration for the active object w.r.t. to the robot hand $\vec{r}_1 = (0,0)^T$ and also displays t_x according to $\vec{r}_2 = (p_x, p_y, 0°, 0°)^T$.

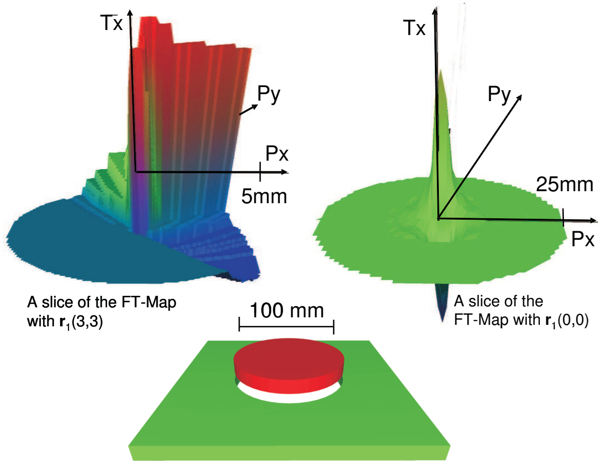

Fig. 3. Example: FT-Map for peg in hole

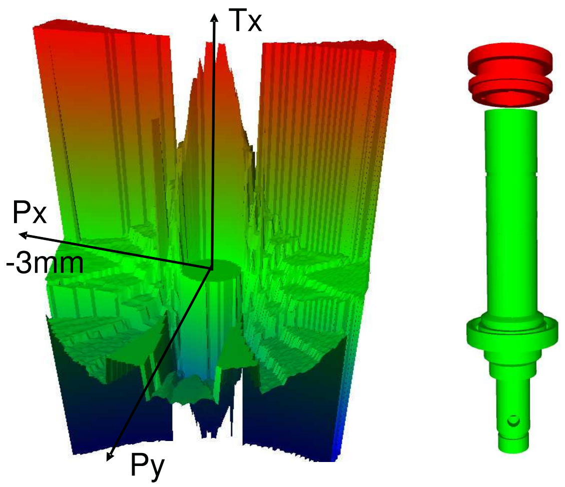

Fig. 4. Example: FT-Map for gear wheel on a shaft

If we need to evaluate a certain particle x_i, we just need to look up in the map and compare the virtual wrench vector with the real one. Looking up in the FT-Map costs $O = (1)$ time. Thus, any particle can be evaluated very fast by a least square method (see section IV).

The next section describes how particles can be evaluated using a vision sensor.

III. Vision for Pose Evaluation

Vision is commonly used for surveillance or guidance of assembly tasks. Our approach is based on detection of object

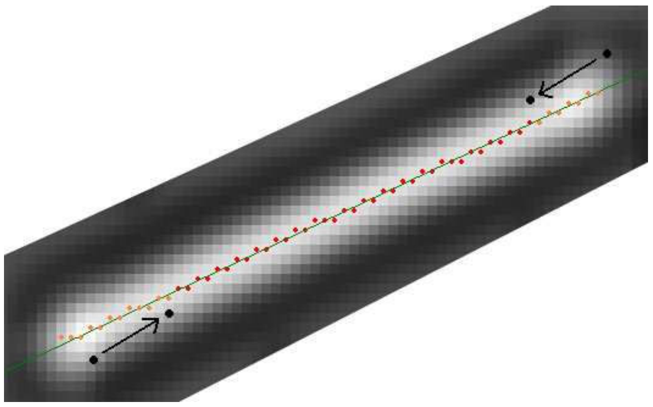

Fig. 5. Line edge detection based on intensity gradients and linear regression

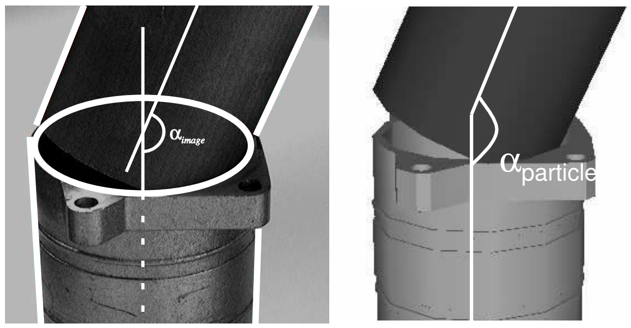

Fig. 6. Left: Typical camera image during shaft-fits-hole assembly with usable features (lines and ellipses); measured angle α_{image} between objects. Right: The hypothetical angle $\alpha_{particle}$ between the axes of two CAD objects, calculated from particle

features in camera images. (Here we select these features manually, but in the future we are preparing an automatic selection approach.)

The traditional procedure is to estimate object pose (3d) from camera images (2d). This always includes the problem of ambiguity in the 2d-3d conversion. In this paper, we examine a different approach: We use vision sensors to *evaluate* possible object poses (given as particles from the particle filter, see chapter IV). Thus, we avoid the problem of ambiguity and only need to use the 3d-2d conversion. Another advantage is that we can incorporate our vision sensor in the same way as we include Force Torque Maps (chapter II).

A. Feature Detection

Figure 6 shows a typical camera image during a shaft-fits-hole assembly. Useful features (edges of shaft and hole) have been marked. In robot assembly setups, the cameras are calibrated with respect to the world, and the absolute pose of objects is approximately known. This allows us to calculate the rough "expected" pose of object features in camera images. Thus, detection of those features can be limited to the area around the expected pose. One fast line edge detection algorithm has been implemented in analogy to [14], using a Bounded Hough Transform. Here, Hough Space resolution and range of values are automatically adopted, i.e. restricted to the "expected" area.

In the following paragraph, we present another very fast algorithm for line edge detection from an expected pose.

The "expected" image pose of the line tells us which area of the image to search. Let us assume (without loss of generality) that the line we are searching for is rather horizontal than vertical in the camera image. Fig. 5 shows the main steps of the procedure in an exemplary intensity gradient image. The brightness represents the gradient amplitude; we are searching for the line along the gradient maxima. The leftmost and rightmost black dots represent the "expected" line endpoints. In the first step, we move those points toward each other in order to deal with the possibility that the pose discrepancy is along the line direction. In the second step, we search for the maximal gradient in each image column between the end points separately, collecting the set of red/dark gray points P. Using linear regression on P, we obtain a provisional line equation l (green/gray). Along this line, we extend P toward both directions (orange/bright gray points) until the gradients fall below a certain threshold, obtaining the left and right x-coordinate bounds x_l and x_r. Linear regression is now used on the extended point set P to achieve higher precision; it gives us the detected line direction, and from x_l and x_r we can easily compute its end points.

Very similar algorithms can be used to detect other feature shapes, e.g. ellipses like in [15].

B. Relative Object Pose Evaluation

1) Relative Feature Pose Comparison Approach: 2d poses of object features cannot (easily) be used to calculate the relative pose of their corresponding objects because of the ambiguity of the 2d-3d conversion. Thus, our first approach for particle evaluation is to compare the relative pose of features in the camera image to their *hypothetical* relative pose in the particle. The hypothetical object feature poses are calculated by 3d-2d conversion of individual object features using camera calibration parameters. The real object feature poses are obtained from feature detection in the camera image (III-A).

Subsequently, *relative* poses of features can be calculated and used for comparison; e.g., angles between line edges, distances between (approximately) parallel edges (see experiment D), distances between corners, angles between cylinder axes, etc.

For example, consider the angle between the axis of a shaft and the axis of its target hole (Fig. 6). We detect the real angle α_{image} in the image. From the given particle, we calculate a hypothetical angle of $\alpha_{particle}$; now the difference between α_{image} and $\alpha_{particle}$ can be used as an evaluation criterion for the particle.

2) Camera "Recalibration" Approach: If our task was to verify a hypothetical *absolute* image pose of *one* object, we could compare its hypothetical image pose with the detected image pose. But here, we want to verify a hypothetical *relative* pose between *two* objects. This means that even if both objects are not detected where they are supposed to be, their *relative* pose might be correct. In order to deal with this problem, we suggest to set a deliberate minor displacement of the camera parameters. This displacement is based on the image poses of one object's features, so that its detected features fit their hypothetical poses. By choosing the *passive* (immobile) object, the displacement needs to be calculated only once (at the beginning of the assembly step). Now we use the displaced camera parameters to calculate the expected poses of the second object's features and compare those with the detected poses in the camera image. This comparison yields a quality criterion for the particle.

In general, this approach is not precise: We add a deliberate falsification of the camera parameters and this falsification cannot be guaranteed to "adopt" the camera parameters correctly. Nevertheless, this imprecision is expected to be much smaller than the benefit that the vision sensor can contribute to the particle evaluation.

IV. Sensor Fusion with Particle Filter

For sensor fusion during assembly, we suggest to use a particle filter. Each particle $x_i \in X$ is evaluated by all available sensors. The force model for the forces and torques given by FT-Maps is able to return a scalar value evaluating each particle, whereas the vision algorithm (described in the previous section) is able to falsify single particles and/or to evaluate the quality of each particle x_i. As implementation of the particle filter we have used the condensation algorithm [2]. The algorithm obtains current measurements as input, e.g. the wrench vector $\vec{w}$ and an image. For integration of vision, we need to compute in a preprocessing step all features, e.g. edges, lines, circles and ellipses, which might be detected in the image. Also a topological representation is necessary to evaluate each particle (given as relative poses between objects). Algorithm 1 describes our implementation of the particle filter.

In Eq. 3 the probability of each sampled particle is computed using standard normal deviation. (The minimal square error method in combination with standard normal deviation assumes linearly independent sensor values f_i – force feedback and p_i – vision feedback, which is not necessarily true here. Although we disregard this, we obtain good results with this method.) The condensation algorithm provides many advantages. On one hand the number of particles is constant throughout the localization process. This allows high execution rates even in higher dimensional spaces. On the other hand we benefit from the subpixel accuracy, which is another very attractive property. Hence, the condensation algorithm is able to deal with models of low resolution. This is very comfortable, because computation of FT-Maps with high density is very time consuming and our implementation of the particle filter deals with low resolution of FT-Maps successfully.

V. Experiments

We have evaluated our approach with variations of peg in hole assembly tasks (see Fig. 7). For all tasks, we have computed FT-Maps, and we have scanned FT-Maps by real contact motions in the workspace. As a first step, we have chosen only the x and y position (in task frame) as unknown degrees of freedom in the relative pose between the objects.

Algorithm 1 Particle Filter for Sensor Fusion in Robot Assembly

```
Require: Model information: FT-Map and representation of
      image features
Ensure: Estimated relative pose between objects
 1: Construct particles $x_i \in X$ according to uniform distribu-
    tion in tolerance space $\Omega$
 2: Move robot into contact with the hybrid position/velocity
    and force controller. Reaching predefined contact forces
    about $-50N$ in z-direction of the task frame.
 3: Observe the environment by measuring the current state,
    described by $z_i \leftarrow (\vec{w}, image)$
 4: Process image and detect features and topology of fea-
    tures in the real image
 5: $sum \leftarrow 0$
 6: for all $x_i \in X$ do
 7:     $weight_i \leftarrow P(z_i|x_i)$ with:
        $P(z_i|x_i) = (\frac{1}{\sqrt{2\pi}\sigma})^{2dof} e^{-\frac{1}{2\sigma^2}(\sum_{j=1}^{dof}(f_j-\overline{f}_j)^2+\sum_{j=1}^{dof}(p_j-\overline{p}_j)^2)}$   (3)
 8:     $sum \leftarrow sum + weight_i$
 9: end for
10: for all $x_i \in X$ do
11:     $weight_i \leftarrow \frac{weight_i}{sum}$ normalize weights
12: end for
13: Get best estimated position according to current weights.
    Move the robot w.r.t. to the goal position and move
    particle set $X$ respectively
14: if $P_z > threshold$ then
15:     return
16: end if
17: Resample particles according to accumulative weights :
18: for all $x_i \in X$ do
19:     $aw_i = \sum_{j=1}^{i} w_j$
20: end for
21: for all $x_i \in X$ do
22:     $r \leftarrow rand[0..1]$ random number uniformly distributed
23:     search the j-th particle for which yield:
            $r > aw_{j-1} \wedge r \leq aw_j$
24:     $x_i \leftarrow$ Gaussian random distribution$(E(x_j), \sigma)$ with
        $\sigma = 0.2$, which depends on the FT-Map's interval
25: end for
```

We have conducted our experiments with the measured FT-Maps as well as with computed FT-Maps. The results presented here have been obtained with the measured maps. Computed maps have also been successfully used in some

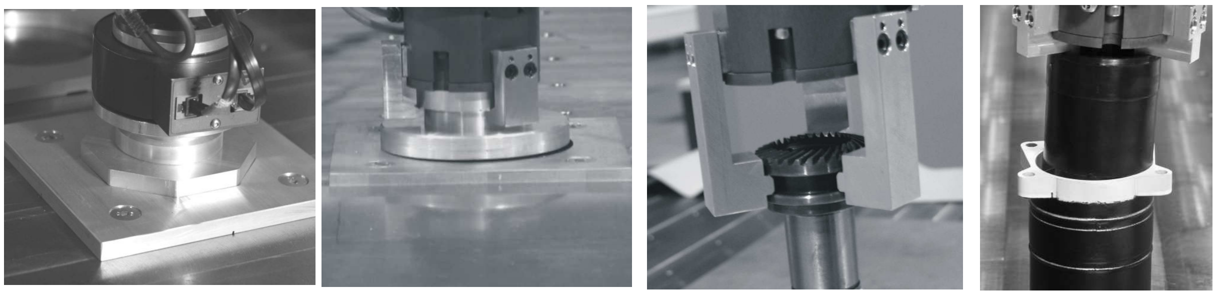

Fig. 7. Four assembly tasks for evaluation our approach; (A) an octogon, (B) peg in hole, (C) gear parts, (D) bearing housing

experiments. Experiments (A), (B), and (C) have been carried out with a force-torque sensor only, whereas experiment (D) demonstrates the successful fusion of two sensors with the particle filter, i.e. vision and force/torque.

For the peg in hole task (B) we repeated our experiment 20 times. Fig. 10 shows the number of contact iterations necessary to execute the robot task successfully. For the assembly tasks (A) we need more than five iterations on average. This was due to the high production accuracy of our objects: The peg with the hole (B) has tolerances of only about $\frac{1}{10}mm$ and the octogon (A) has tolerances of only about $\frac{5}{100}mm$.

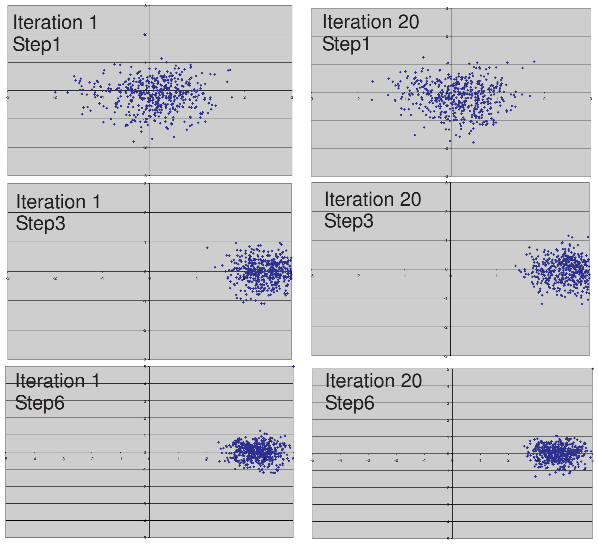

Fig. 8. Particle distribution of robot task (A) after first, third and sixth contact motion.

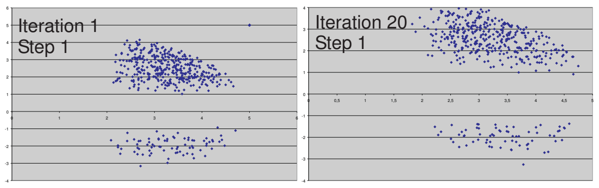

Fig. 9. Particle distribution of robot task (C); only one contact motion was necessary

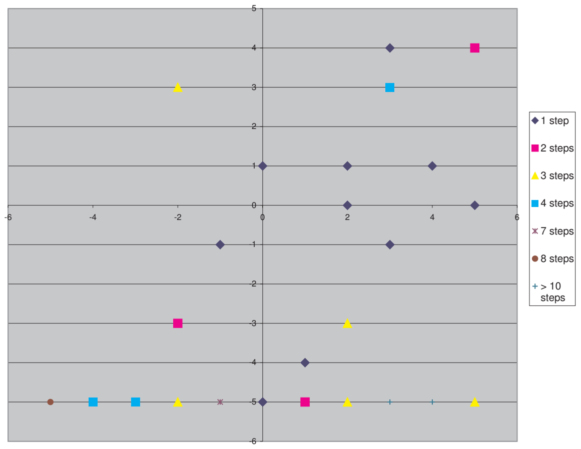

Fig. 10. Number of steps needed for successful execution of peg in hole robot task, according to x and y positions.

The distribution of particles is depicted in Fig. 8, while task (A) is assembled, and Fig. 9 illustrates the particles while the gear wheel is assembled with the gear shaft (C). In the first example six contact measurements were needed. Between each measurement 20 iterations of the particle filter have been carried out. The pose estimation was successful in the last step (right side). In the second example (C), two clusters were detected (due to ambiguities in FT-Map), but the correct pose has been estimated after one contact measurement. For this experiment the correct pose could be estimated in less than three steps, mostly.

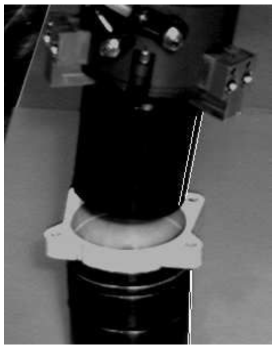

Fig. 11. Vision evaluation: comparison of hypothetical and real line distances.

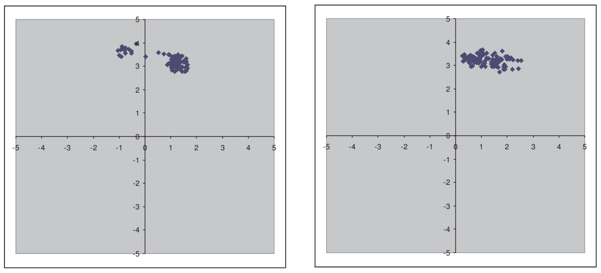

Fig. 12. Particle filter with vision and FT-Map: the first two iterations

In experiment (D) vision has been used as an additional sensor. It significantly decreases the number of necessary contact measurements. Our algorithm detects the right edges of both parts and compares their real pixel distance to the hypothetical pixel distance of every particle (Fig. 11), thus evaluating the particle. The same is done with the left edges. The camera has been set up looking in x-direction, so that it can judge the y-coordinate very well, but not the x-coordinate. The effect can be seen in Fig. 12: After the first particle filter iteration, the particles center around the y coordinate 3.2 (correct for the given pose would be 3.0) but are widely spread along the x-axis. This happens because the vision sensor cannot distinguish between them, and their entries in the FT-Map are rather similar. After only few more iterations, the assembly succeeds.

VI. Conclusion and Future Works

In our paper, we present a new approach for autonomous execution of robot tasks. The FT-Map computation has been introduced and it is to our knowledge the first implementation to generate this model information from CAD-data. It has become possible by employing modern graphical processor units. Also we have applied the particle filter for sensor fusion in robot assembly tasks, i.e. combining FT-Maps and vision. With this idea, we provide a further step toward automated robot programming, with the aim of using CAD-data for generating executable, robust programs. Of course, the particle filter for the fusion of various sensors is well-known in mobile robotics, but our way of using it in assembly tasks is new. The particle filter has been applied successfully to the robot tasks shown here. It is very useful for fusion of sensory information. Off-line generation of visible features from CAD-data will be a new approach and can be used to automate vision application in the future. More experiments with the integration of automatically computed FT-Maps as well as automatically selected visible object features, and more unknown degrees of freedom will be carried out by us in the near future.

VII. Acknowledgments

We would like to thank QNX Software Systems for providing free software licenses of their real time operating system QNX 6.3.

References

[1] S. Joerg and J. Langwald and J. Stelter and G. Hirzinger and C. Natale, ”Flexible robot-assembly using a multi-sensory approach”, *IEEE International Conference on Robotics and Automation*, USA, 2000, pp.3687-3694.

[2] M. Israd and A. Blake, ”CONDENSATION - conditional density propagation for visual tracking”, *International Journal Computer Vision*, 1998.

[3] S. Thrun, ”Particel filters in robotics”, *Proceedings in 17th Annual Conf. Uncertainty in Artificial Intelligence*, 2002.

[4] S. R. Chhatpar and M. S. Branicky, ”Localization for Robotic Assemblies with Position Uncertainty”, *IEEE/RSJ International Conference on Intelligent Robotic Systems*, USA, 2003.

[5] S. R. Chhatpar and M. S. Branicky, ”Localization for Robotic Assemblies Using Probing and Particle Filtering”, *IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, USA, 2005, pp. 1379-1384.

[6] T. Lefebvre and H. Bruyninchx and J. De Schutter, *Nonlinear Kalman Filtering for Force-Controlled Robot Tasks*, Springer Berlin, Heidelberg, New York, 2005.

[7] U. Thomas and F. M. Wahl and J. Maass and J. Hesselbach, ”Toward a New Concept of Robot Programming in High Speed Assembly Applications”', *IEEE International Conference on Intelligent Robots and Systems* Edmonton, Canada, Aug. 2005, pp. 3932-3938.

[8] T. Hasegawa and T. Suehiro and K. Takase, ”A Model-Based Manipulation System with Skill-Based Execution”, *IEEE Transactions on Robotics and Automation*, vol. 8, no. 5, 1992, pp. 535-544.

[9] R. Smits and H. Bruyninchx and W. Meeussen and J. Baeten and P.Slaets and J. De Schutter, ”Model Based Position-Force-Vision Sensor Fusion for Robot Compliant Motion”, *IEEE International Conference on Robotics and Automation*, USA, 2006.

[10] W. Meeussen and J. Rutgeerts, K. Gadeyne, H. Bruyninckx and J. De Schutter, ”Contact State Segmentation using Particle Filters for Programming by Human Demonstration in Compliant Motion Tasks”, *IEEE International Conference on Robotics and Automation*, USA, 2006.

[11] J. Baeten and J. De Schutter, *Integrated Visual Servoing and Force Control*, Springer Verlag, Berlin, 2003.

[12] T. Kroeger and B. Finkemeyer and S. Winkelbach and L.Eble and S. Molkenstruck and F. M. Wahl, ”Demonstration of Multi-Sensor Integration in Industrial Manipulation”, *IEEE International Conference on Robotics and Automation*, Orlando, USA, 2006, Video.

[13] J. Maass and N. Kohn and J. Hesselbach, ”Open modular robot control architecture for assembly using the task frame formalism”, *International Journal of Advanced Robotic Systems*, March, 2006, pp.001-010.

[14] M. Greenspan and L. Shang and Piotr Jasiobedzki, ”Efficient Tracking with the Bounded Hough Transform”, *Computer Vision and Pattern Recognition*, 2004, pp. I-520- I-527 Vol.1.

[15] M. Vincze and M. Ayromlou and M. Zillich, ”Fast Tracking of Ellipses using Edge-Projected Integration of Cues”, *International Conference on Pattern Recognition*, 2000, pp.72-75 Vol.4.