2012 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 7-12, 2012. Vilamoura, Algarve, Portugal

A Flexible 3D Object Localization System for Industrial Part Handling

Øystein Skotheim[1], Morten Lind[2], Pål Ystgaard[2] and Sigurd A. Fjerdingen[1]

Abstract— **We present a flexible system that can scan and localize workpieces in 3D for assembly and pick and place operations. The system contains a vision robot that acquires 3D point clouds by performing sweeps with a laser triangulation sensor. Because the sensor is mounted on a robot, we can choose a viewpoint and a scanner trajectory that is optimal for a given task. For example, we can sweep along a semi-circular trajectory in order to recognize and grasp parts from a pallet. With the same vision robot, we can perform a sweep of the packaging container in order to recognize and manipulate elements of cardboard.**

The system also incorporates software that recognizes and localizes objects based on an acquired 3D point cloud and a CAD model of the object to search for. The core matching algorithm is based on oriented point pairs and a Hough-like voting scheme. The method has been improved with a robust clustering algorithm as well as methods for pose verification and pose refinement that significantly increase the accuracy and robustness of the system.

As an application of the system, an industrial prototype workcell is presented. The task is to recognize, grasp and transfer parts of office chairs, such as seats, seat backs and armrests, from a pallet to a cardboard container. It demonstrates how the vision system is easily set up to recognize three different components by using CAD models. The prototype workcell further demonstrates how the pose estimation results can be fed directly to a separate handling robot in order to grasp a chair seat. A series of ten experiments was performed where the chair seat was placed in arbitrary poses in the pallet. Pose estimations were performed in just over one second per experiment, and the obtained accuracy was well within the tolerances for the grasp operation in all ten cases.

I. INTRODUCTION

Increasing competitiveness by offering a wider variety of products at a higher pace to the market is a global trend for manufacturing companies. This has a strong impact on automated production systems used to manufacture and assemble such products; flexible and agile automation are key concepts for success. Simply expanding the numbers and types of jigs, grippers and feeders to meet increased demand for flexibility results in expensive and space consuming manufacturing cells. The equipment may also become obsolete after only a short period of time, due to rapid changes in requirements.

There is a need for flexible solutions that can meet these new demands and still be compact, give high throughput and deliver a low production cost per unit. This has lead to an accelerating use of robots in production systems; due to the high flexibility and versatility of robot manipulators. Robots are typically employed for handling, feeding, packaging, and assembly of workpieces. One key challenge with using robots in flexible, automated production is the ability to identify and localize workpieces in a general, adaptable way so that the robot manipulator can manipulate them.

[1]SINTEF ICT, Trondheim, Norway. Contact: oystein.skotheim@sintef.no
[2]SINTEF Raufoss Manufacturing AS, Trondheim, Norway

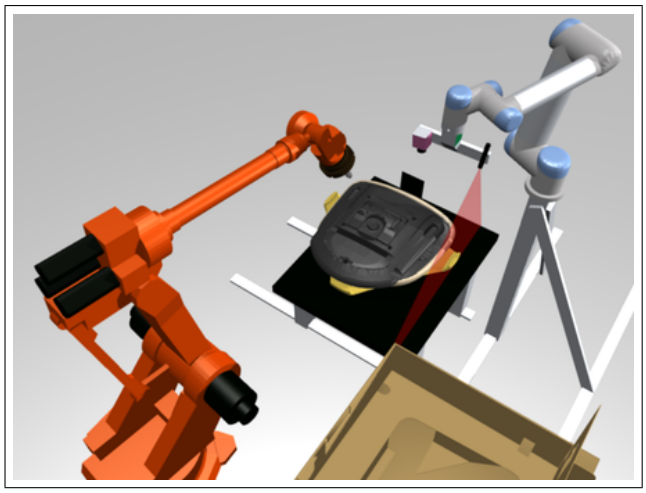

Fig. 1. The setup of robots, laser, smart camera, chair seat, and cardboard box in the transfer cell prototype. Illustration from emulation setup.

In this paper we present a flexible system for localizing a workpiece in 3D. The system consists of a robot manipulator with a laser triangulation sensor mounted on the robot wrist and algorithms for scanning, recognizing and localizing an object in 3D. Experiments with system performance and usability are conducted with an industrial prototype packaging cell from a manufacturer of office chairs.

Manually operated processes in manufacturing systems have high tolerances on positions and orientations of parts and components. This is mainly due to manual labor intelligence and versatility. It poses a major challenge to automating the manual operations by use of traditional approaches with industrial robots. The common solution is to lower the tolerances and require high precision and a structured layout of material, parts, and components; permeating the entire production and manufacturing system.

Vision systems for recognition and localization of mechanical parts have been in use for some decades now. Most systems are based on standard cameras and 2D image analysis. Naturally, these systems only work when the object poses are limited to a few degrees of freedom, such as parts arriving in a single layer on a conveyor belt. 3D vision systems are steadily advancing in availability, but are still in their infancy with regards to flexibility and robustness. Typically the logic required for recognition needs to be custom designed for each component. This also means that a significant amount of programming effort is needed with each new type of component introduced into the system.

978-1-4673-1736-8/12/S31.00 ©2012 IEEE

There is a demand for more generic, flexible and robust 3D vision systems that allow full 3D localization of workpieces in production. Especially so for small-series manufacturing that need to handle a large diversity of products.

The main contributions of this paper are twofold. First, a flexible way of scanning a workpiece in 3D is presented. This method combines the advantages of having a freely moving robot manipulator with the benefits of a robust sensor system. The sensor system in itself provides good results under varying conditions, even for shiny and difficult surfaces, due to a high dynamic response and a low sensitivity to ambient light. However, it is limited to scanning a 2D line at a time. Mounting it on a robot manipulator wrist alleviates this restriction and gives us the ability to scan each workpiece using a scanning trajectory, called a sweep, that is optimal for the given task and workpiece. For example, we can use a semi-circular trajectory to scan the workpiece in order to minimize self-occlusions and to acquire as many 3D points as possible from its exterior surfaces. The same vision robot may be used to scan the target packaging container for the workpiece, this time with a different trajectory and at a different location.

Our second contribution is an experimentally validated algorithm for 3D localization based on a 3D point cloud and a set of CAD models of the objects to search for. The core matching algorithm is based on oriented point pairs and a Hough-like voting scheme[1]. We have extended this method with a robust clustering algorithm as well as algorithms for pose verification and pose refinement that significantly increase the accuracy and robustness. This makes the system fast and robust enough for use in our industrial prototype cell.

The paper is organized as follows. Section II discusses related work. Section III gives an overview of our prototype with specifications for the robots and the 3D sensor as well as how the system has been calibrated. In section IV, the algorithms for 3D object recognition are presented. Section V describes some experiments that we have performed in order to demonstrate the abilities and the performance of the system. Finally, a summary, conclusions and comments regarding further work are given in sections VI and VII.

II. RELATED WORK

Three essential sub-tasks of a robotic assembly or pick-and-place operation are (1) object recognition and localization, (2) path planning, and (3) grasping. Each of these areas represent their own distinct research field.

There has been an increasing interest in robotic bin picking in recent years, both due to appearance of new and versatile 3D sensors and the introduction of more light-weight robot manipulators. The main advantages of using a robot manipulator instead of a specially-suited mechanical solution are cost per unit and adaptability. One recent example of such work is presented by Fuchs et. al. [2]. Here, a KUKA LWR-III has a wrist-mounted time-of-flight camera as well as a gripper. The system must be reprogrammed to grip different geometries, and the operation speed is limited by having the sensor system mounted on the handling robot.

Pauli et. al. [3] presents a vision-based robot system for object inspection and handling where the sensor system is independent from the handling robot. A stereo vision camera is used, mounted on a pan-tilt unit. Their aim is to recognize features of objects held in front of the camera by the robot manipulator so that the manipulator may handle the object correctly at a later stage. Generality of the approach is limited by having to define recognizable features for each object, and the cycle time is limited because the handling robot spends a significant amount of time in order to grasp and hold the workpiece in front of the sensor head. Similar approaches using external cameras are given by several other authors, *e.g.,* by Son [4], focusing on control planning strategies in partially unknown environments.

Algorithms for 3D object localization (also called registration) can be grouped into two categories: fine registration and coarse registration methods. A popular approach for fine registration is the iterative closest point algorithm (ICP) [5]. This algorithm works by iteratively estimating the homogeneous transformation that minimizes the distance between the points in the source point cloud and their closest points in the target point cloud. This approach works very well when the two point clouds are already close together. However, the algorithm is prone to local minima and hence cannot normally be used without a good initial guess.

Coarse registration methods are often based on correspondences between global or local features. Global features include computations such as volume, area, principal components and statistical moments [6]. The problem with methods based on global features is that they tend to work well only for complete 3D data. A measurement from a 3D sensor typically provides only partial 3D data due to a limited set of viewpoints. Shadows, occlusions, reflections and other optical effects result in additional areas of missing data.

For that reason, methods based on local features are popular for 3D object recognition on partial data. Examples of popular local feature descriptors are spin images [7], tensors [8] and harmonic shape contexts [9]. These descriptors all work by considering the local geometry in a so-called support region around a set of interest points. For small support regions these methods will be more robust to partial data, but at the same time the descriptors will be less discriminative. This means that these methods work best for objects with a relatively high level of detail in their local geometry.

Many of the industrial parts we have been working with consist mainly of relatively large, planar or slightly curved surfaces. For this reason, we have chosen to base our object recognition algorithms on simple, oriented point pair features that are fast to compute. We have earlier presented a related algorithm based on oriented point pairs and a cost function[10], [11]. In this paper, we focus on an implementation that uses a Hough-like voting scheme[1]. The method works well for most types of objects, also for objects that lack high-frequency details in their geometry. It handles relatively large amounts of clutter and occlusion, and it works well even for rather coarsely sampled point clouds as long as the global shape of the object is preserved. Because of the

relatively low computational complexity, a pose estimation result for one or more objects can normally be obtained in less than a second. A more detailed description of the method with measures for recognition rates and comparisons to other methods from the state-of-the-art was done by Drost et. al. [1]. In order to increase the robustness and the accuracy of the method, we have extended the core algorithm with several pre- and post-processing steps, such as a robust clustering algorithm and methods for pose verification and refinement. These improvements are described in section IV. In section V, we demonstrate how the output of the method can be fed directly to a handling robot for grasping.

III. OVERVIEW OF PROTOTYPE

A prototype work cell for a manufacturer of office chairs has been set up in our laboratories. The work cell operation is to perform the transfer of product parts to the product package. This operation is working directly integrated with the final assembly transport system and robustness and cycle time is of critical importance in the real production system.

The real work cell transfers the product components for the finished product, which have been through the assembly system, to a structured packaging in a folded cardboard box. The product parts arrive at the transfer station on a conveyor palette, in a structured arrangement. However, the manufacturing operations have high tolerances on the geometric positions and orientations of the parts within their dedicated regions on the pallet.

The prototype work cell is illustrated by a rendered image from a 3D model in Figure 1. It comprises a pallet carrying a seat for an office chair, a cardboard box for packaging the components, a handling robot (orange, Nachi SC15F-2) with a picker-tool mounted for transferring the components, a vision robot (aluminium/blue, Universal Robots UR5) with a mounted sensor for sweeping the pallet and the cardboard box.

A. Vision Robot and 3D Sensor

The vision robot is able to acquire 3D point clouds in a flexible way by performing sweeps with a laser triangulation sensor along a chosen trajectory. The specific robot used for performing the sweeping of the sensor in our experiment setup is a Universal Robots UR5.

The laser triangulation sensor consists of a laser module and a CMOS smart camera attached to a small aluminium fixture. The laser is a ZLaser 100mW module with a wavelength of 660nm and a cylindrical lens that provides a laser line with a fan angle of 45 degrees. The camera is a PhotonFocus MV-D1024E-3D01 CMOS camera with a CameraLink interface. It has a built-in FPGA for calculating peak positions of the laser line. This camera also has a programmable LinLog response that allows us to perform high dynamic range (HDR) imaging of difficult objects, such as metal parts with shiny areas. The camera provides laser line peak positions at a rate of 150fps when used in full resolution (1024x1024), but the rate can be increased up to a maximum of 3900fps when a smaller region-of-interest is used. An optional bandpass filter can be used in front of the camera which effectively filters out most of the ambient lighting. The PhotonFocus camera is connected to a PC running Windows via its CameraLink interface, and a software component was written that continuously reads laser line profiles from the camera, transforms these into a series of calibrated 3D points and outputs these points via UDP over Ethernet.

We haven chosen to use a laser triangulation sensor instead of a full-field 3D camera (such as structured light camera or a Microsoft Kinect), because it is a fast, compact, robust and highly accurate sensor that works well under strong ambient illumination and it provides good quality data even for dark or shiny objects due to its HDR abilities. Another reason for choosing this sensor, is that it is applicable for other typical measurement scenarios, such as for 3D scanning of parts that arrive on a conveyor belt. In this case, the sensor can be mounted directly above the conveyor, and the linear motion of the conveyor can be utilized to perform the 3D scanning.

B. 3D Sensor Calibration

In order to convert from a set of pixel positions to a set of 3D coordinates, the camera and the laser need to be calibrated.

1) Camera and laser calibration: The camera is calibrated with a chessboard pattern printed on a planar glass plate, and the popular Camera Calibration Toolbox for MATLAB [12] was used to estimate the intrinsic and extrinsic parameters for the camera. A diffuse, white planar surface is measured at two or more known heights in order to estimate the parameters for a plane in space, *i.e.,* a position $\mathbf{p_0}$ and a normal vector $\mathbf{n}$, modelling the propagation of the light from the laser. From the camera model, we get expressions for the rays in space that correspond to each pixel in the camera image. The peak positions of the laser line can then be transformed into a set of actual 3D coordinates in the sensor reference frame by finding the intersection of each pixel ray with the laser plane.

2) Transformation of coordinates: In order to use the acquired 3D information to control the handling robot, the coordinates need to be transformed from the sensor reference frame to the base frame of the handling robot. One of the approaches that we have used, is to acquire 3D point clouds of a set of calibration objects, such as half spheres, and fit geometric primitives to them to obtain their position in the sensor frame. With the help of a pointing tool and the kinematics of the handling robot, we can obtain the corresponding position of these calibration objects in the base frame of the handling robot. We can then use standard least squares methods, such as [13], to obtain a best fit estimate for the transformation from the sensor frame to the base frame of the handling robot. Note that such a transformation will be valid only for a given sensor trajectory.

Eye-in-hand calibration: An alternative to computing a simple best-fit transformation as described in the previous section, is to perform a so-called eye-in-hand calibration.

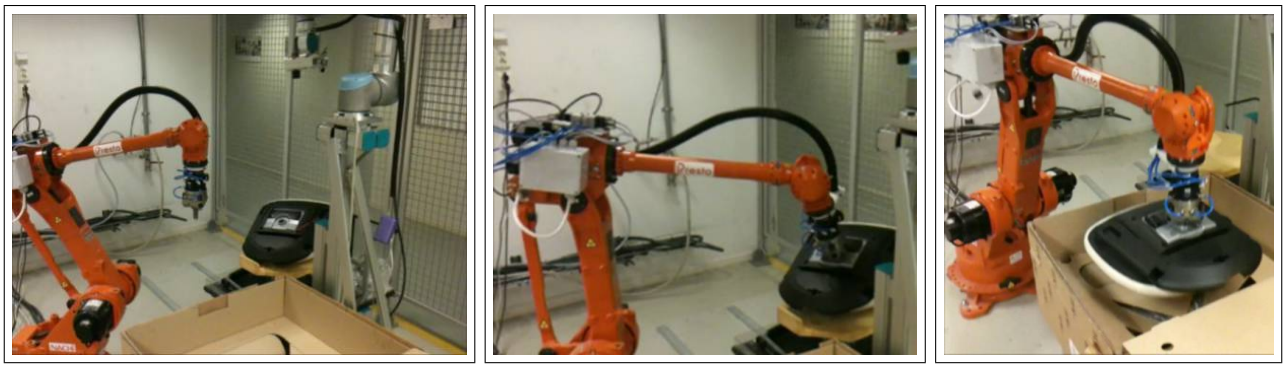

Fig. 2. The physical work cell with a vision robot (aluminium/blue), a handling robot (orange), and a chair seat in its pallet during the scanning operation (left), during the grasping (middle) and during packaging (right).

Ideally, we would like to obtain 3D measurements in the base frame of the vision robot. This would give us the flexibility to acquire 3D point clouds by sweeping the sensor along arbitrary trajectories. Eye-in-hand calibration refers to the process of identifying the sensor frame with respect to the tool flange frame of the vision robot. Numerous techniques exist; confer *e.g.,* [14], [15], [16]. Generally the principles of calibration are based on a sensor-observable calibration object with fixed, but unknown, relation to the robot base. A series of observations of the object by the sensor from various robot arm poses may be used to set up an over-determined set of equations to be solved by a least-squares fit.

C. Handling Robot

The handling robot in this setup is a Nachi SC15F equipped with a 3-finger concentric gripper to lift the chair seat. The gripper fingers are designed to make an internal grip in the hole where the chair lift is to be mounted. The tip of the gripper is 7 mm in diameter and the hole has a diameter of 27 mm, so there is a geometric tolerance of ± 10 mm in the gripping operation.

IV. 3D OBJECT RECOGNITION

Our system includes a module for recognizing and localizing objects in three dimensions. The input to this component is a 3D measurement (the scene) and a 3D model of the object to be detected and localized (the template). The output is a list of pose hypotheses (positions and orientations) for the object, each with a corresponding score value.

The template point cloud may be obtained either by 3D scanning of the object or it may be obtained from its CAD model. The open source GoTools library [17] has been used to automatically convert CAD models in the IGES format to point clouds (including accurate surface normals) with the desired density.

The core matching algorithm is implemented as described by Drost et. al. [1] However, the method has been extended with various pre- and post-processing steps, such as a fast clustering algorithm, methods to evaluate the accuracy of the pose suggestions, and methods for discarding false matches. All time-critical steps have been implemented in C++. Python was used for some supporting code as well as to tie all the components together and provide a user-friendly interface.

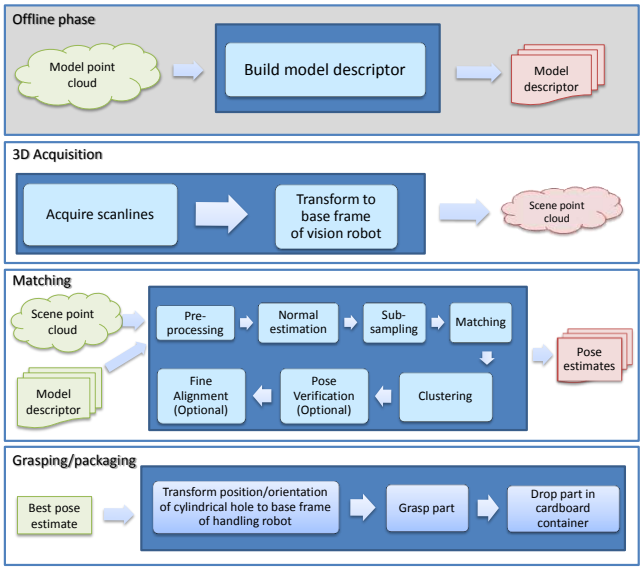

Fig. 3. Illustration of the various steps in the system. The model descriptor is calculated off-line and only once for each model that we want to recognize. The output from the 3D acqusition module is a point cloud in the base frame of the vision robot. This point cloud is fed along with the model descriptor to the matching module, which outputs a set of pose estimates with score values. The best pose estimate is sent to the handling robot for grasping and packaging

The matching consists of 7 different steps: preprocessing and outlier removal, normal estimation, subsampling, matching, clustering, pose verification and fine alignment (see figure 3 for a flow chart). These 7 steps will be further described in the following sections.

A. Preprocessing and Outlier Removal

Since the speed of the matching strongly depends on the number of points in the scene, it is beneficial to exclude as many irrelevant points as possible. The user can hence specify a 3D region of interest, given as minimum and maximum X, Y and Z coordinates. This can be used to exclude

certain regions of the scene. (One example is to discard all the points below and including the surface of a table). In the preprocessing step, we have also implemented a filter that segments the range image into connected components and removes small and isolated clusters of points from the point cloud.

B. Normal Estimation

The next step is to estimate surface normals for the scene point cloud. This has been done by looping over all the points in the point cloud and fitting local planes to small support regions around each point. The FLANN open source library[18] was used to perform fast nearest neighbor queries. Since surface normals have a 180 degree ambiguity, the normal estimator needs to be aware of the position of the camera in order to consistently orient all the normals against a common viewpoint.

C. Subsampling

One of the advantages of the matching algorithm that we have implemented is that it tends to work well even for coarsely sampled point clouds, as long as the global shape of the template and the scene is preserved. (On the contrary, some of the local feature based algorithms, such as spin image matching[7], typically require dense sampling in order to preserve local geometric details.) We use a voxel grid filter to uniformly subsample the point cloud. This works by constructing a voxel grid with a specified unit size over the point cloud and by replacing all the points that fall inside each voxel with the centroid of these points. Typically the number of points is decreased from around 1 million points to around 1000 points in this step. This allows us to obtain matching speeds around 1 second. We have also implemented more sophisticated subsampling strategies, *e.g.,* strategies that increase the point density in high-curvature regions. However, we have found that uniform subsampling works well for most objects.

D. Matching

The main recognition algorithm is based on a Hough-like voting method, which was developed by Drost et. al. and is described in detail in [1]. The algorithm has an off-line step and an on-line step. In the off-line step, a model descriptor is calculated for the template. The model descriptor is a table that allows us to quickly look up model points with a given feature descriptor. The feature descriptor for a given point pair $(\mathbf{m_1}, \mathbf{m_2})$ with surface normals $(\mathbf{n_1}, \mathbf{n_2})$ is calculated as follows:

$$\mathbf{F}(\mathbf{m_1}, \mathbf{m_2}) = (\|\mathbf{d}\|, \angle(\mathbf{n_1}, \mathbf{d}), \angle(\mathbf{n_2}, \mathbf{d}), \angle(\mathbf{n_1}, \mathbf{n_2})) \quad (1)$$

In this equation $\mathbf{d} = \mathbf{m_2} - \mathbf{m_1}$ is the displacement vector between the model points and $\|\mathbf{d}\|$ is the Euclidean distance between the model points. Basically we calculate a very simple feature descriptor that consists only of the distance between the points and three different angular relationships between the surface normals and the point-to-point vector.

In the on-line step, we loop through all the points in the scene and select them one by one as the reference point $\mathbf{s_r}$. Each reference point is then paired against every other point in the scene, $\mathbf{s_i}$, and the feature vector $\mathbf{F}(\mathbf{s_r}, \mathbf{s_i})$ is calculated. The model descriptor is used to check whether there exist model point pairs $(\mathbf{m_r}, \mathbf{m_i})$ with a similar feature vector.

Let us refer to a point with an associated surface normal as an oriented point. If we have a correspondence between an oriented scene point pair $(\mathbf{s_r}, \mathbf{s_i})$ and an oriented model point pair $(\mathbf{m_r}, \mathbf{m_i})$, we first align the points $\mathbf{s_r}$ and $\mathbf{m_r}$ by a simple translation. Afterwards we align their surface normals $\mathbf{n_r^s}$ and $\mathbf{n_r^m}$ such that both of them are parallel to the x-axis. The final step is to rotate the oriented model point $\mathbf{m_i}$ around the x-axis with an angle α until it aligns with the oriented scene point $\mathbf{s_i}$.

This means that, for each reference point $\mathbf{s_r}$, we get a set of pose hypotheses that can be parametrized as (m_j, α), where j is an index into the list of model points (with accompanying surface normals) and α corresponds to the final rotation around the x axis. We construct a two-dimensional accumulator table with the model point number, $j = 1, 2, ..., N$, along one axis and a discretized set of angles, $\alpha = \alpha_1, \alpha_2, ..., \alpha_M$, on the other axis. Every time we find a correspondence between an oriented point pair in the scene and the model, we vote for a value for j and α in the table. After all points are processed, we keep only the hypotheses that have received a high number of votes.

E. Clustering

After the matching step has been run, we have a large list of hypotheses, each with an associated number of votes. The final step is to identify clusters of hypotheses that are close together in rotation and translation. Even though Drost et. al. mention pose clustering[1], they do not explain well how this should be done. We have experimented with several strategies for clustering. One strategy that we have found to work well is to represent each of the hypotheses as a vector $\mathbf{t}$ and a quaternion $\mathbf{q}$. $\mathbf{q}$ corresponds to the rotation and $\mathbf{t}$ is the translation that should be applied to the template in order to align it with the scene point cloud. We proceed by selecting the hypothesis in our list that has received the highest number of votes. We then perform a fixed distance neighbor search in translation space to identify hypotheses that are close together in translation. We have implemented this in a fast manner by storing all the translation vectors in a KD tree. We go through each of the neighbors and check the angular distance, $\theta = 2\arccos(|\mathbf{q_1^* q_2}|)$, between their associated quaternions $\mathbf{q_1}$ and $\mathbf{q_2}$, where $|\mathbf{q_1^* q_2}|$ refers to the scalar part of the product between the conjugate of $\mathbf{q_1}$ and $\mathbf{q_2}$. If the angle θ is smaller than a given threshold, we add the hypothesis to the current cluster and remove it from the table of remaining hypotheses. Each cluster gets a score that corresponds to the total number of votes received by hypotheses within the cluster. When we have processed all the neighbors in the list, we create a new cluster by selecting the remaining hypothesis with the highest score. The clustering process is terminated either when there are

no remaining hypotheses to cluster or when we have reached a given, maximum number of clusters. Finally, we sort the clusters by score in descending order, starting with the cluster with the highest score. For each cluster, we search for other clusters that are close together in translation but with different orientations. For each of these groups of clusters, we keep only the cluster with the orientation that provides the highest score. Note that, for the clustering to work optimally, the template point clouds should be translated such that their center of gravity coincides with the origin prior to the matching.

F. Pose Verification

In order to make the matching more robust to false positives, we usually employ one or more functions to verify the validity of the pose hypotheses that comes out of the clustering process above. If we have some constraints on the poses, *e.g.,* if we expect the objects to be supported by a planar surface or if we know that the objects are stable only in a subset of poses, we can use this information to exclude obvious erroneous pose suggestions. Another method that we have found to work quite well is to use what we call a projection grid. The idea is to project the scene points onto a plane that is perpendicular to the line of sight of the sensor. This plane is divided into $M \times N$ cells. The model is then inserted in the hypothesized pose and projected into the same plane. If multiple model points are projected into the same cell, we keep only the model point that is closest to the sensor. Finally, we check whether a sufficient percentage of the model points in this projection is supported by points in the scene, *i.e.,* that there exist points in the scene that are sufficiently close to the model points.

G. Fine Alignment

Because of the way the described matching algorithm works, where the pose space is partitioned into a limited set of angles and distances, the accuracy will not be higher than the bin sizes used for discretization. However, the accuracy can be increased by applying a separate algorithm for fine alignment, such as the ICP algorithm[5]. ICP works by iteratively estimating the transformation that minimizes the distance between the points in the source and their closest points in the target. When using ICP, it is important to exclude points from the source point cloud that we cannot expect to have a correspondence in the target. We have chosen to use the acquired 3D scene as the source and the template as the target. The scene may be cluttered with points from the background and other objects which should be excluded. One way to achieve this is to use a projection grid, much like described in the previous section. In this way, we can compare the projections of the scene and the template in a plane that is perpendicular to the line of sight of the sensor, and we select only points from the scene that overlap with the projection of the template in the hypothesized pose.

V. EXPERIMENTS

In order to test the performance of the system, experiments were performed with the prototype cell described in section III. In these simple experiments we chose to move the robot arm with constant velocity along a linear path, fixed with respect to the robot base. A complete sweep could then be assembled by simply adding an offset to each individual scan line in the point cloud based on their associated timestamp and the scanning velocity. The transformation between the sensor frame and the base frame of the handling robot was calculated by the method discussed in section III-B.2 by using a set of four calibration objects in the form of half spheres on top of cylinders with different heights.

A. Recognition of Various Parts

Our first experiment was carried out to show how the system is easily configured in order to recognize several different parts. Dense point cloud models were generated from the CAD models of three different items: the seat, the back and the armrest of an office chair, by sampling points on their exterior surfaces as discussed in section IV.

3D point clouds of scenes containing each of these items in unknown poses were captured by the vision robot. Due to the relatively dark appearance of the black plastic material on these chair parts, an exposure time of around 10ms was used for the PhotonFocus camera, which gave us a scanning speed of about 100 profiles/second. A sweep consisting of 350 scan lines was then carried out in about 3.5 seconds. Typically, one sweep provides us with around 100.000 valid 3D points (in total for the pallet and the chair part.) Surface normals were estimated on the full dense point clouds before they were subsampled. For the chair seat and the seat back we used a point distance of 20mm, which reduced the amount of points to around 1000 before they were sent to the matcher module. For the arm rest we used a point distance of 10mm, since this part is much smaller in area.

Figure 4 shows the result of the 3D object matching when applied to three different parts of the chair: the seat, the arm rest and the back. Normal estimation for each of these 3 objects was performed in around 0.4 seconds (for the full point cloud), the matching was performed in around 0.8 seconds and the clustering in just 5 ms, in total around 1.2 seconds on an Intel Core i7 1.6GHz laptop computer. Optionally, the pose estimates were further refined by running the ICP algorithm, which in this case had an additional running time of around 0.5 seconds. The running times can be further reduced by increasing the point distances such that fewer points from the scene are sent to the matcher module. However, this will negatively affect the accuracy and the robustness of the pose estimation.

Note that the algorithms have not been optimized yet, and they currently run only in a single thread on a single CPU core. A trivial way to speed up the normal estimation would be to calculate surface normals only for the subsampled points. We are currently working on a parallellized implementation for multi-core CPUs or GPUs, as discussed briefly in the further work section.

Even though we have only presented results for three different objects in this paper, we have successfully applied

the system for localization of a large variety of objects in our laboratory.

B. Accuracy Test for Chair Seat

In the second experiment, we performed recognition, grasping and packaging of chair seats by feeding poses obtained from the recognition system to the handling robot. A series of 10 recordings was done, where the chair seat was lifted up and placed down in random positions and orientations in its pallet. The chair seat was displaced up to ± 100 millimeters in X and Y, and tilted up to ± 25 degrees around the X and Y axis and ± 180 degrees around the Z axis. To provide a metric for the accuracy of the matching, we have estimated a ground truth for the center of the cylindrical hole on the chair seat in each of the 10 recordings. (This cylindrical hole is where the handling robot grasps the chair seat.) The ground truth estimation was done by fitting 3D circles to the circular opening in each of the acquired point clouds by using the Raindrop Geomagic software for point cloud analysis.

The results of these experiments are summarized in Table I. (X, Y, Z) are the detected positions of the centric hole of the chair seat in millimeters. The matching algorithm also provided us with values for the orientations of the chair seat, but these values were left out of the table for clarity. Δ_1 shows the difference in the detected position of the centric hole and the ground truth after coarse localization with oriented point pairs. Δ_2 shows the difference to the ground truth after ICP has been run for pose refinement. As we can see, the mean error after localization with oriented point pairs is around 10.7 mm, and the standard deviation is 4.5 mm. This was not accurate enough for reliable grasping. When ICP was run as a successive algorithm for pose refinement, the mean error was reduced to 1.8 mm with a standard deviation of 0.9 mm. When the results after ICP refinement were fed to the handling robot, it managed to successfully grasp the chair seat and drop it into the cardboard box in all of the experiments.

#	X [mm]	Y [mm]	Z [mm]	Δ_1 [mm]	Δ_2 [mm]
1	-93.79	252.62	95.29	10.61	0.51
2	-89.19	275.13	96.61	9.02	2.43
3	-43.71	267.34	94.91	10.21	0.94
4	-60.92	274.22	96.51	2.23	1.51
5	-77.16	256.83	95.62	10.76	2.50
6	-42.53	218.27	97.10	11.11	2.30
7	-49.24	212.47	124.14	20.94	2.05
8	-58.24	217.04	114.98	13.76	3.55
9	-1.66	221.57	119.09	7.37	1.34
10	-42.55	307.54	126.91	10.88	0.99
			μ [mm]	10.69	1.81
			σ [mm]	4.48	0.87

TABLE I

LOCALIZATION RESULTS FOR CHAIR SEAT

VI. SUMMARY AND CONCLUSIONS

A flexible system for scanning and localizing workpieces in 3D for use in robot based handling operations has been presented. The system consists of a robot manipulator with a wrist-mounted laser triangulation sensor. Because the sensor is mounted on a robot, optimal viewpoints and scanning trajectories can be chosen depending on the task we would like to execute.

An essential component of the system is a module able to recognize and localize objects in 3D with six degrees of freedom. Since the only inputs to the object matcher are a 3D measurement of the scene and a 3D template, it is easily set up to recognize a variety of different objects. The templates can be either CAD models or point clouds acquired from 3D scanning. We would like to emphasize that our 3D object localization module in principle works together with any sensor that provides measurement data in the form of dense 3D point clouds, such as laser scanners, structured light systems, dense stereo cameras, time-of-flight cameras and Microsoft Kinect.

The matching algorithm that we have used is based on oriented point pair features and a Hough-like voting scheme. Even though the original authors of the algorithm have demonstrated a high recognition rate[1], their results were not accurate enough for reliable grasping and manipulation. We have proposed several improvements including a robust clustering method as well as methods for pose verification and pose refinement. These additional steps help to ensure that the pose estimation results are sufficiently robust and accurate to be useful in an industrial setting.

It has been demonstrated how the vision robot is easily set up to scan and recognize different parts for a manufacturer of office chairs. When used in combination with the ICP algorithm for pose refinement, the obtained accuracy was more than good enough to successfully grasp and manipulate the parts with a separate handling robot.

VII. FURTHER WORK

In the experiments described in section V, the sensor was moved on a linear path, fixed with respect to the robot base, at constant velocity in order to simplify the calibration procedure. We are currently working on more sophisticated eye-in-hand calibration methods (see section III-B). This will allow us to move the sensor along arbitrary trajectories, since the vision robot then will be able to output data in the static reference of its base.

The experiments were conducted with the original factory calibration of joint offsets for the vision robot. It has been identified that the eye-in-hand calibration currently does not achieve the expected precision. The deficiency is ascribed to the fact that the joint offsets of the vision robot were not properly calibrated. It is now a work in progress at our laboratories to conduct experiments with such calibration methods as described by [19], [20], [21] to ensure that the vision robot operates with an adequate level of accuracy.

On the object recognition side, we are working on a GPU implementation of the matching algorithm based on oriented point pairs. This will allow us to increase the point density and hence also the robustness and accuracy of the algorithm without sacrificing speed. The speed gain will also allow

(a) Result for chair seat (b) Result for arm rest (c) Result for seat back

Fig. 4. Pose estimation result for the chair seat, arm rest and seat back. In orange color: point clouds captured by the scanner. In gray color: CAD models of the corresponding part in the detected position and orientation.

us to perform simultaneous searches for multiple models in more complex scenes.

We are also working on more flexible solutions for grasping, in particular we have started the work on a dual arm manipulation system composed of two robot arms, each with a simple gripper. We expect that the combination of a generic 3D object localization method with more flexible manipulators will give the system the ability to both recognize and manipulate a large diversity of product parts and components without requiring a large range of specialized gripper tools.

ACKNOWLEDGMENT

This work is part of the project "Next Generation Robotics for Norwegian Industry"[22]. The project is funded by the Norwegian Research Council and the following end users from the industry: Statoil, Hydro, Tronrud Engineering, Scandinavian Business Seating, Glen Dimplex Nordic and RobotNorge.

REFERENCES

[1] B. Drost, M. Ulrich, N. Navab, and S. Ilic, "Model globally, match locally: Efficient and robust 3d object recognition," in *Proceedings of CVPR 2010, IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2010.

[2] S. Fuchs, S. Haddadin, M. Keller, S. Parusel, A. Kolb, and M. Suppa, "Cooperative Bin-Picking with Time-of-Flight Camera and Impedance Controlled DLR Lightweight Robot III," in *IEEE/RSJ 2010 International Conference on Intelligent Robots and Systems (IROS 2010)*, 2010.

[3] J. Pauli, A. Schmidt, and G. Sommer, "Vision-based integrated system for object inspection and handling," *Robotics and Autonomous Systems*, vol. 37, pp. 297–309, 2001.

[4] C. Son, "Intelligent control planning strategies with neural network/fuzzy coordinator and sensor fusion for robotic part macro/micro-assembly tasks in a partially unknown environment," *International Journal of Machine Tools and Manufacture*, vol. 44, pp. 1667–1681, 2004.

[5] S. Rusinkiewicz and M. Levoy, "Efficient variants of the icp algorithm," in *3-D Digital Imaging and Modeling, 2001. Proceedings. Third International Conference on*, 2001, pp. 145–152.

[6] C. Zhang and T. Chen, "Efficient feature extraction for 2d/3d objects in mesh representation," in *Image Processing, 2001. Proceedings. 2001 International Conference on*, vol. 3, 2001, pp. 935 –938 vol.3.

[7] A. E. Johnson and M. Hebert, "Using spin images for efficient object recognition in cluttered 3d scenes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 5, pp. 433–449, 1999.

[8] A. S. Mian, M. Bennamoun, and R. Owens, "Three-dimensional model-based object recognition and segmentation in cluttered scenes," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 28, no. 10, pp. 1584–1601, 2006.

[9] T. B. Moeslund and J. Kirkegaard, "Pose estimation using structured light and harmonic shape contexts," in *Advances in Computer Graphics and Computer Vision*, 2008, pp. 281–292.

[10] O. Skotheim, J. T. Thielemann, A. Berge, and A. Sommerfelt, "Robust 3d object localization and pose estimation for random bin picking with the 3dmama algorithm," in *Proceedings of Three-Dimensional Image Processing (3DIP) and Applications*, A. M. Baskurt, Ed., vol. 7526, no. 1. SPIE, 2010, p. 75260E. [Online]. Available: http://link.aip.org/link/?PSI/7526/75260E/1

[11] A. Transeth, O. Skotheim, H. Schumann-Olsen, G. Johansen, J. Thielemann, and E. Kyrkjebo, "A robotic concept for remote maintenance operations: A robust 3d object detection and pose estimation method and a novel robot tool," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, oct. 2010, pp. 5099 –5106.

[12] J.-Y. Bouguet, "Camera calibration toolbox for Matlab," http://www.vision.caltech.edu/bouguetj/calib_doc/index.html.

[13] B. K. P. Horn, "Closed-form solution of absolute orientation using unit quaternions," *Journal of the Optical Society of America A*, vol. 4, no. 4, pp. 629–642, 1987.

[14] K. Strobl and G. Hirzinger, "Optimal Hand-Eye Calibration," in *International Conference onIntelligent Robots and Systems*, Oct. 2006, pp. 4647 –4653.

[15] F. Park and B. Martin, "Robot Sensor Calibration: Solving AX = XB on the Euclidean Group," *IEEE Transactions on Robotics and Automation*, vol. 10, no. 5, pp. 717 –721, Oct. 1994.

[16] F. Dornaika and R. Horaud, "Simultaneous Robot-World and Hand-Eye Calibration," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 4, pp. 617 –622, Aug. 1998.

[17] SINTEF Applied Mathematics, "GoTools geometry toolkit," http://www.sintef.no/Projectweb/Geometry-Toolkits/GoTools/.

[18] M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration," in *International Conference on Computer Vision Theory and Application VISSAPP'09)*. INSTICC Press, 2009, pp. 331–340.

[19] M. Ikits and J. M. Hollerbach, "Kinematic Calibration Using a Plane Constraint," in *International Conference on Robotics and Automation*, vol. 4. IEEE, Apr. 1997, pp. 3191–3196.

[20] H. Chen, T. Fuhlbrigge, S. Choi, J. Wang, and X. Li, "Practical Industrial Robot Zero Offset Calibration," in *International Conference on Automation Science and Engineering*. IEEE, Aug. 2008, pp. 516–521.

[21] M. Lind, "Automatic Robot Joint Offset Calibration," in *International Workshop of Advanced Manufacturing and Automation*, K. Wang, O. Strandhagen, R. Bjartnes, and D. Tu, Eds. Trondheim, Norway: Tapir Academic Press, June 2012.

[22] I. Schjølberg, "Development of robot solutions for oil/ gas, aluminium and manufacturing industry," in *IEEE Conference on Emerging Technologies and Factory Automation (EFTA)*, 2012.