Precise Measurement of Cargo Boxes for Gantry Robot Palletization in Large Scale Workspaces Using Low-Cost RGB-D Sensors

Yaadhav Raaj[1(✉)], Suraj Nair[1], and Alois Knoll[2]

[1] TUM CREATE, 1 CREATE Way, #10-02 CREATE Tower, Singapore 138602, Singapore
{raaj.yaadhav,suraj.nair}@tum-create.edu.sg
[2] Technische Universität München (TUM), Institüt für Informatik, Robotics and Embedded System, Munich, Germany
knoll@in.tum.de

Abstract. This paper presents a novel algorithm for extracting the pose and dimensions of cargo boxes in a large measurement space of a robotic gantry, with sub-centimetre accuracy using multiple low cost RGB-D Kinect sensors. This information is used by a bin-packing and path-planning software to build up a pallet. The robotic gantry workspaces can be up to 10 m in all dimensions, and the cameras cannot be placed top-down since the components of the gantry actuate within this space. This presents a challenge as occlusion and sensor noise is more likely.

This paper presents the system integration components on how point cloud information is extracted from multiple cameras and fused in real-time, how primitives and contours are extracted and corrected using RGB image features, and how cargo parameters from the cluttered cloud are extracted and optimized using graph based segmentation and particle filter based techniques. This is done with sub-centimetre accuracy irrespective of occlusion or noise from cameras at such camera placements and range to cargo.

1 Introduction

Motivation for this work was realized from having to build a fully automated palletizing/depalletizing gantry robot for the Air Cargo Terminal. Packing Cargo shipment into standardized containers/pallets is critical. One such pallet is the P6P (Fig. 1) measuring approx. (3 x 2.5)m, and other pallets can be found in Boeing documentation [2]. Cargo primarily consists of block-shaped shipments ranging from (0.3 to 2.0)m in 3 dimensions made of different materials from cardboard to wood, and are traditionally built up by hand onto the pallet. Our system aims to automate this process, by making use of a 4 axis Gantry Robot (Fig. 1) with approx. workspace dimensions (7.5 x 5 x 7.5)m, that can be further

Electronic supplementary material The online version of this chapter (doi:10.1007/978-3-319-54190-7_29) contains supplementary material, which is available to authorized users.

S.-H. Lai et al. (Eds.): ACCV 2016, Part IV, LNCS 10114, pp. 472–486, 2017.
DOI: 10.1007/978-3-319-54190-7_29

extended if needed be. The end effector (EF) has 4 degrees of freedom X, Y, Z, θ and can lift cargo using a vacuum suction gripper.

1.1 Robot Workspace

Cargo is placed in a measurement space (Yellow) of the robot, where it is measured/weighed before being placed in the pallet (Grey). A bin packing and path-planning software work together to palletize and depalletize the robot. However, this component requires sub-centimetre level precision in order to optimize it's packing and prevent collision as is evident in many bin packing planners [3]. Cargo boxes are introduced at random into this space, and we do not know the dimensions of these beforehand. This is why a robust vision system with a focus in precise measurement is required. This process is not trivial as the cameras cannot be placed vertically overhead as the actuation mechanism works along that area as with most gantry robots. Hence, multiple cameras have to be placed in an inclined manner along the pillars of the robot, so as to capture maximum possible information, from which depth and color information is sent to a server for processing.

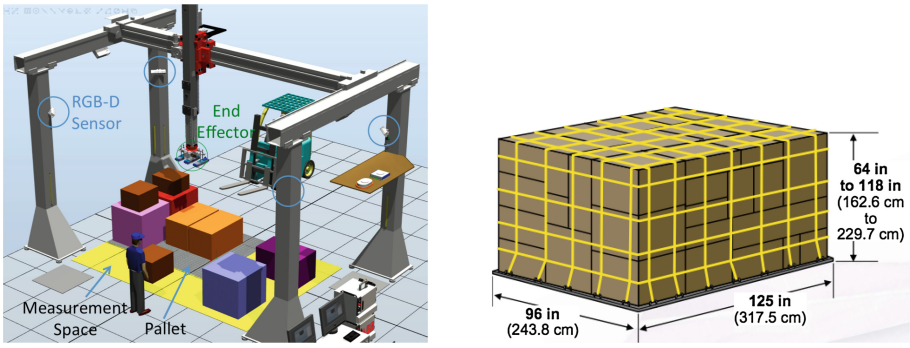


Fig. 1. Gantry Robot and P6P Pallet to be built up used in the Air Cargo Industry

Our objective in this paper, is to extract the measurements of unknown cuboid shaped cargo in the measurement space as accurately as possible, irrespective of occlusion due to the camera placement, or highly similar boxes being in close contact. Solving the problem of occlusion requires information from multiple sensors, hence having industry grade sensors may be too costly, hence we had to tailor design our method to work with commercial grade sensors that may have significant noise.

There has been not much prior work on achieving centimetre level precision of unknown models at ranges beyond 3 to 4 m using commercial grade sensors before this, with the constraint of not having top-down cameras. Hence we aim to fill this gap to achieve our objectives above. Our algorithm works with Kinect

2 RGBD cameras, by fusing point cloud data from multiple cameras, fitting models to this data, segmenting it using graph based approaches and refining this further by using edge information from the RGB cameras. This is further improved by using a 9DOF particle filter approach. Results from these are then used by the bin-packing and path-planning software to optimize and build up the pallet. The system may also be used to depalletize the pallet.

2 Related Work

2.1 Industrial Applications

There are already a significant number of industry players/systems in the automation space, such as Bosch, Universal Robotics and Z Automation. A simple search on PackExpo would give a hundred more. However, most of these systems have been designed for conveyor line scanning, where the object's model can be extracted (Fig. 2). Universal Robotic's [4] solution works with unknown models, but it's commercially available solutions advertise small workspaces, with robots having an eye-in-hand configuration (Fig. 2). There is no known solution for large gantry workspaces, where cargo can be placed anywhere in the workspace for palletization.

2.2 Research Applications

Significant work has already been done in the space of extracting known objects through feature/template matching either in the point cloud or image space, and is a well-defined problem. This is evident in many robotic depalletizing and bin packing papers, such Drost's [5] and Holz's paper [6], where he proposes comparing object surfel models with sensor data for object/pose detection, in a small constrained workspace. The objects studied there are geometrically feature rich as well, unlike our cargo boxes.

For unknown objects, Ryan Lloyd [7] suggests the use of a single RGB-D sensor, by extracting the top surface of a point cloud, after applying euclidean

Fig. 2. SICK LMS Line Scanning and Universal Robotic's Depalletization

clustering and point cloud smoothing with Moving Least Squares (MLS) [8]. Features of the top surface are then extracted and used to train an SVM to classify different primitives, including quads. Dimensions are extracted using a Minimum Volume Bounding Box (MVBB) projected to the ground, and centimetre level precision is achieved. In this paper however, experiments are carried out with an organized point cloud due to the single camera metrology, at ranges below 1m.

Also, in Richtsfeld's seminal paper [9], he uses a combination of SVM and graph cut based segmentation to segment and get dimensions of primitive objects in cluttered scenes. The focus of this paper was not measurement, but rather precise segmentation of objects based on their color and shape information irrespective of occlusion. Much of the experiments were also conducted at ranges below 1 m with a single camera, hence spurious point cloud noise was not much of an issue.

Alternatives to SVM/Graphcuts such as SIFT matching [10] and Primitive Shape Matching [11] were also studied. Many similar papers including those above performed pose estimation using an initial SVD/PCA based seed, with an iterative refinement step. In all of these papers, a single structured light based RGB-D camera such as the Asus Xtion or Kinect 1 is used, and were generally designed for smaller workspaces such as tabletops. Many of these algorithms were also designed for single camera inputs with organized point clouds. Our workspace exceeded 5 m, and an unorganized point cloud was generated due to multiple cameras. At such ranges, the depth image also had significant Z noise, resulting in noisy point clouds, hence noisy edges.

This meant that the above methods of using MVBB, SVD/PCA to perform pose and dimension estimation or using Richtsfeld's method to perform model fitting and segmentation were challenging. To solve this issue, edge based fusion methods were also studied. Anwer's [12] paper describes using depth image refinement techniques such as depth normalization and bilateral filters. Aouada's [13] paper describes using RGB images as guidance images, where holes are filled referencing edges found in the higher resolution RGB image. These techniques are later referenced and implemented in our paper.

Lastly, large scale point cloud acquisition and measurement techniques are also studied in the form of LIDAR and Aerial Imagery fusion. It is challenging to find large scale point cloud processing methods without exploring the LIDAR domain. In Wang's [14] paper, he describes how a graph is constructed from the noisy aerial LIDAR data, how edges are extracted using iterative RANSAC techniques, and how these edges are further refined by projecting these points into the 2D Aerial camera and corrected them to edges and corners found.

3 Sensors and Systems

3.1 Camera System

A range of commercial and industrial 3D systems were explored. Industrial solutions consisted of Structured Light systems by GOM and SICK, Laser systems by Faro and Osela, and Time of Flight (TOF) systems by Basler. The GOM,

SICK and Faro systems had sub-millimetre accuracy, but had a limited range of below 1m, and were costly in large numbers. The Basler TOF system was reasonably priced, touted centimetre accuracy, and had a range of up to 5 m but it was yet to begin sales during testing time. Commercial systems tested included the Asus Xtion Pro, Intel RealSense, Microsoft Kinect 1 and 2. In the end, we decided to go with the Kinect 2, which is a TOF system with a range of up to 4.5 m, and could output 512 x 424 pixel Depth images and Full HD RGB images.

3.2 Experimental Setup

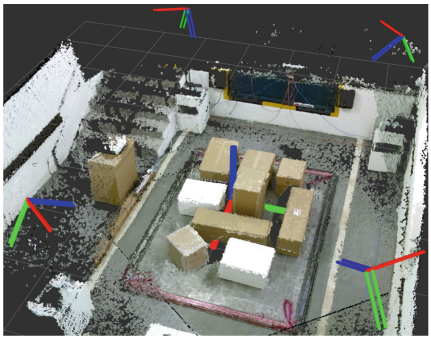
Our experimental setup consists of 4 NUC/Kinect units with a measurement space of approximately (4 x 3.5)m, with the Kinects at a height of 2.5 angled at 45 degrees. The highly reflective P6P pallet is placed in the centre of the setup and cargo boxes ranging from 30 cm to 1 m in either dimension, consisting of cardboard, styrofoam and shrink-wrapped boxes were used. Our objective is to simultaneously measure and track the boxes that are initially placed flat against the ground (though they may be in close contact), to simulate the situation where a cargo item of unknown dimensions or pose has been introduced into a robot workspace, and needs to be worked upon.

3.3 Data Extraction

RGB and Depth imagery is extracted via the libfreenect2 [15] driver through the iai-kinect2 [16] ROS (Robot Operating System) [17] interface. This driver performs several pre-processing steps. Firstly, it allows one to perform intrinsic and extrinsic calibration of the Kinect sensor by solving for intrinsics and radial distortion using Zhang's checkerboard method [18], and solving for the transformation between the IR and RGB sensors. Secondly, edge aware and bilateral filters are used to enhance the depth image. Finally, a registration step is performed to generate a depth image in the RGB frame. All of this is performed on a 6th Generation Intel NUC i5 with GPU optimization on the registration step, allowing image rates of up to 20 Hz. Each camera sends a 16 bit upscaled (540 x 960) depth image and full HD JPEG compressed RGB image to the main server via a gigabit switch.

3.4 Point Cloud Fusion

We use the approximate time scheduling policy implemented in ROS to extract a joint set of data from all cameras. Since, ${}^{R}T_{W}$ is known, points can be generated on the server, and transformed to the world frame, after which thresholding can be applied to eliminate points outside the scope of the workspace. This is done on all 2 million over points at once on the GPU. Since there are redundant points due to overlapping views, a Voxel Grid filter is applied with a distance threshold. Euclidean clustering is then applied to eliminate spacial noise, and to break



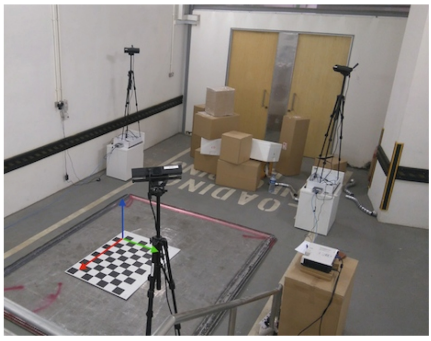


Fig. 3. ROS Integration and Experimental Setup

the point cloud into chunks that can be analysed in parallel for performance boost. These steps are done using PCL (Point Cloud Library) [20]. Finally, a Moving Least Squares (MLS) filter is applied on these chunks [8]. This is a good pre-processing step [7], as it allows for better thresholding based on normal vector values, and smoothes out noise in the point cloud spatially. Bruce Merry's GPU implementation of MLS [21] is used. This allows us to generate clean $(R, G, B, X, Y, Z, nx, ny, nz)$ point clouds of the entire 3D scene at up to 10 Hz, where (nx, ny, nz) represents the point surface normals (Fig. 3).

4 Algorithm

4.1 Model Extraction

With the point cloud made available, we are able to extract pose, dimensions and features from this. The items we are dealing with are primarily cuboid shaped cargo, that are placed against the floor of the workspace initially. These constitute 80 % of the cargo, and allows for the best packing density.

$$ax + by + cz + d = 0\ (Plane) \quad (x, y, z) = (x_0, y_0, z_0) + t(a, b, c)\ (Line) \tag{1}$$

$$m = \frac{1}{N}\sum_{i=1}^{N}(p_i) \qquad c = \frac{1}{N}\sum_{i=1}^{N}\left((p_i - c)(p_i - c)^T\right) \tag{2}$$

$$C = \left(\sum_{j=1}^{N}\lambda_j p_j : \lambda_j \geq 0\, for\ all\ j\ and\ \sum_{j=1}^{N}(\lambda_j = 1)\right) \tag{3}$$

There are several methods to extract the pose and dimensions from cuboid objects. Applying Iterative RANSAC works for ellipsoids, cylinders, planes and lines, but not for cuboids since there is no mathematical model for it. Iteratively finding planes (Eq. 1) and their intersections through a least squares approach fails in our case as boxes being densely packed removes outward facing planes.

Applying methods such as PCA (Principle Component Analysis) by computing the mean/covariance of a set of points p_i and extracting it's min/max eigenvectors (Eq. 2), or MVBB (Minimum Volume Bounding Box) [1] by optimizing a bounding box based on various heuristics as used in [6,7] works poorly in our case (Fig. 4). This is because such methods optimize for all points, including outliers, and don't take the object model into account. Measurements derived through this method had an μ error of up to 5 cm from the ground truth, which is unacceptable for the bin packing algorithm.

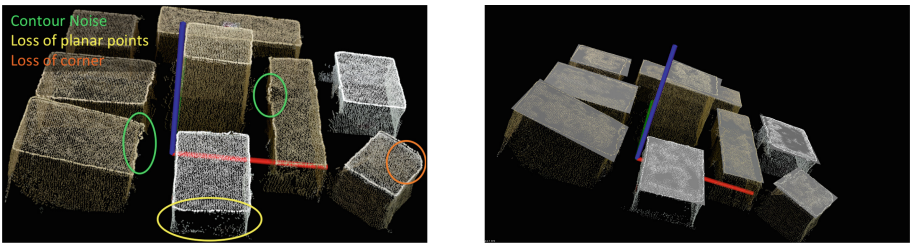


Fig. 4. Contour extraction and PCA/MVBB fitting

Hence, we use a top surface contour extraction approach, to simplify measurement and pose extraction. We select only top surface points ($nz > 0.5$), fit a plane to it via RANSAC, and compute the convex hull C for points p_i (Eq. 3). A Sobel operator is applied over sections of the RGB image that C projected onto each RGB image occupies, and it's contours C_{proj} are extracted where $C_{proj} \subseteq C$. We then iteratively fit these to 3D line models (Eq. 1). Working in a multi-camera point cloud domain ensures that occlusion is not much of an issue (Fig. 5).

4.2 Segmentation Using Node Graphs

If cargo boxes are separated spatially, or have different heights, the above clustering and segmentation approaches work. However, if they are packed closely, have the same height and the same color (very likely scenario in depalletization situations), they end up getting clustered together (Fig. 6). In both cases, the contours were extracted from the same euclidean cluster. Hence splitting them requires some kind of geometric analysis.

We solve this problem graphically. First, a bi-directional connected graph $G_c = (E, V)$ is constructed. G_c contains vertices $(v_i ... v_n)$, where each vertex v_i contains a corresponding point p_i, where $deg(v_i) \geq 1$. To construct this graph, we begin with our edges represented as vectors $(\overrightarrow{e_i} ... \overrightarrow{e_n})$ with a point passing through them defined through the line model $(x, y, z) = (x_0, y_0, z_0) + t(a, b, c)$. An edge $\overrightarrow{e_i}$ has it's min/max points $p_i^{(min)}, p_i^{(max)}$. We construct nodes n_i by intersecting every edge $\overrightarrow{e_i} \bigcap \overrightarrow{e_j}$ $(i \neq j)$. Repeat nodes are inevitable, so these

Fig. 5. Boxes here are spatially classified into one cluster due to proximity, hence requiring methods to split them up

are eliminated by averaging neighbours using a Oct-Tree search. A node v_i is created if the distance between it's parent edges $\overrightarrow{e_i}, \overrightarrow{e_j}$ is minimized, and if the angle between them are either close to 90 or 0 degrees (Eq. 4).

$$G_c = \left(\sum_{i=0}^{N} \sum_{j=i+1}^{N} \left(v_i = \overrightarrow{e_i} \bigcap \overrightarrow{e_j} \; (i \neq j) \right) \; where \; min \left(f(v_i, \overrightarrow{e_i}, \overrightarrow{e_j}) \right) \right)$$

$$f(v_i, \overrightarrow{e_i}, \overrightarrow{e_j}) = \left(\| v_i, closer(p_i^{(min)}, p_i^{(max)}) \|_2 - 0.1 \right)$$

$$+ \left(\| v_i, closer(p_j^{(min)}, p_j^{(max)}) \|_2 - 0.1 \right) + min \left((\angle(\overrightarrow{e_i}, \overrightarrow{e_j}) - 90), (\angle(\overrightarrow{e_i}, \overrightarrow{e_j}) - 0) \right) \tag{4}$$

Once this graph has been constructed, we simply have to locate inner cycles $cycles_c$ within it. Such a problem can result in exponential search time, but with geometric constraints this is no longer true. This is done in a Depth-First-Search (DFS) approach. Given a set of searched connected cycles where each set N_i contains a set of nodes $N_i \epsilon (n_i^0 ... n_i^n)$ that contain their edges $E_i \epsilon (\overrightarrow{e_i^0} ... \overrightarrow{e_i^{n-1}})$, we ensure that the *angle* between them are close to 90 or 0 degrees, their $l2norm$ is maximized, the vectors move in a consistent clockwise/counter-clockwise direction and that they are connected (Eq. 5).

$$cycles_c = min \left(\sum_{i=0}^{M} \sum_{j=0}^{N} \left(g(E_i \epsilon (\overrightarrow{e_i^j} ... \overrightarrow{e_i^{n-1}}) \; where \; \overrightarrow{e_i^j} = \overrightarrow{e_i^{n-1}} \right) \right)$$

$$g(E_i) = \left(0.1 - \| \overrightarrow{e_i^j} \|_2 \right) + min \left(\left(\angle(\overrightarrow{e_i^j}, \overrightarrow{e_i^{j+1}}) - 90 \right), \left(\angle(\overrightarrow{e_i^j}, \overrightarrow{e_i^{j+1}}) - 0 \right) \right)$$

$$+ \left((\overrightarrow{e_i^j} \times \overrightarrow{e_i^{j+1}}).z - (\overrightarrow{e_i^{j+1}} \times \overrightarrow{e_i^{j+2}}).z \right) \tag{5}$$

This is done by using cross-product and normalized dot-product of a set of connected edges generated from the nodes as a constraint. Strongly connected component algorithms from Tarjan, Tiernan and elementary circuit discovery methods from Johnson [22] were studied and integrated for the search steps. These graph generation and analysis steps can be visualized (Fig. 6). In the left

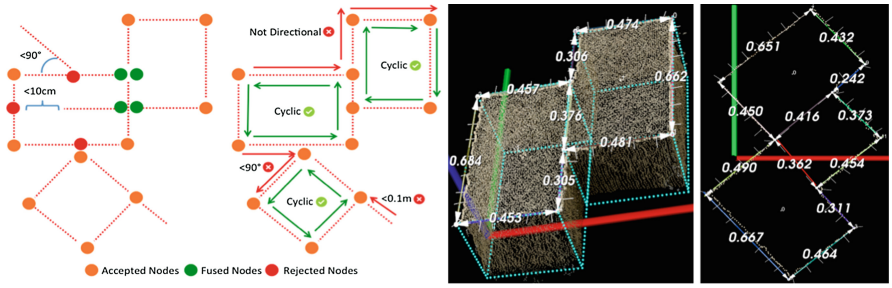


Fig. 6. Graph construction and Analysis (Color figure online)

image, one observes how a node isn't created if it's intersection with another edge isn't close enough, or it's angle doesn't satisfy the above constraints (in red). Nodes created that are too close are fused together (in green). In the right image, we can see how the search occurs, and how cycles are detected. Measurements derived through this method had an μ error of up to 2 cm from the ground truth, which is unacceptable for the bin packing algorithm.

4.3 RGB Camera Sobel Edge Fusion

One can observe the above error in measurements by projecting the fitted line into one of the camera RGB images (Fig. 7). You can see that the line doesn't fall exactly on the edges found on the box. This results in deviations from the ground truth measurement. Hence, it is observed that it might be worth using the RGB image to "correct" the contour points before fitting line models to it. This method is inspired by Wang's paper [14], where the initial model is superimposed onto a sobel edge image from the RGB camera, and then corrected. However, his solution is constrained to working from a top down view, but our scenario involves sensors that are placed inclined to the workspace, since downward facing cameras cannot be used due to obstruction with the robotic gantry components. This means that selecting a sensor to correct these contour points is not trivial, due to occlusion, and distance to the sensor. Such occlusions can be observed (Fig. 7).

Hence, the following correction algorithm is proposed. This is applied before the iterative RANSAC line search and graph generation. Given a contour $C \epsilon \{p_i..p_m\}$ with a set of points, a set of hypothesis for each p_i is generated with a 2D normal constraint in the search space (Eq. 6), where t varies to generate $p_i \epsilon (p_i^0 ... p_i^n)$.

$$p_i \epsilon \Big(\{p_i^j .. p_i^m\} \; for \; all \; j \; (p_i^0 + p_i^j.normal * t) \Big) \tag{6}$$

$$min \left(\sum_{j=0}^{M} \frac{1}{N} \sum_{c=0}^{C} \Big(255 - intensity(p_i^j, I_c^{RGB}) + distance(p_i^j, I_c^{Depth}) \Big) \right) \tag{7}$$

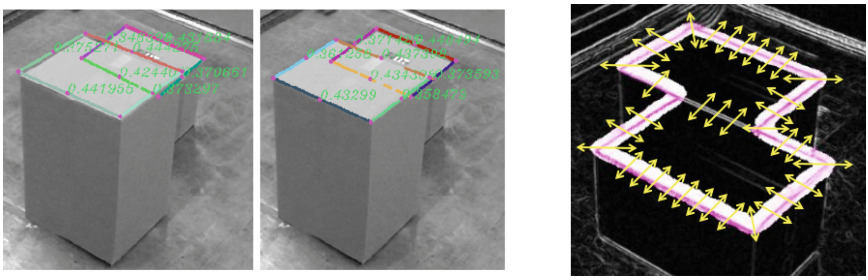


Fig. 7. Before and After Sobel Edge Fusion, and Correction of points along contour normals

Each hypothesis p_i^j is transformed and projected into both the RGB Sobel Edge I_c^{RGB} and Depth Images I_c^{Depth} for each camera. We then attempt to select the best p_i^j to replace p_i that minimizes the distance to the camera, the Z value on I_c^{Depth} to prevent converging on a edge/solution that is being occluded, and maximize a point falling on a edge on I_c^{RGB} (Eq. 7). We can observe the correction in (Fig. 7). Measurements derived through this method had an μ error of up to 0.5 cm from the ground truth, which is deemed suitable. You will notice that the height of the box wouldn't be optimized this way, but we found this was already made accurate during the planar RANSAC fit.

4.4 Model Constrained Pose and Sobel Edge Refinement

At this point, one can already extract the measurements from the various cargo boxes. However, we found that there were situations where several points on the contour may converge at a highly deviant position, leading to a failure in RANSAC line fitting. This is because we corrected each point individually instead of constraining it to a line model. We needed to further refine the pose in 6D and measurements and make use of the Sobel Edge information to do that in a box model constrained manner, while also making use of multiple cameras and accounting for occlusion. We also needed to track small perturbations/movements in the cargo boxes in case a worker may shift them. We found that the ideal solution to this would be a particle filter based optimization approach [23] (Fig. 8).

A particle filter with a 9 dimensional state $X_{n|n-1}^{(k)}$ is setup to represent a cuboid with position (x_w, y_w, z_w) about the centre of mass, dimensions (l_w, w_w, h_w) and 6-dof orientation $(\alpha_w, \beta_w, \gamma_w)$ in the world frame for a given particle k with 500 particles. A Brownian motion model based based on additive gaussian noise w_{n-1} is used (Eq. 8). Our weights described later are calculated using a multi-variate gaussian distribution model represented in a log-likelihood function for faster computation (Eq. 9).

$$X_{n|n-1}^{(k)} = [x_w \;\; y_w \;\; z_w \;\; l_w \;\; w_w \;\; h_w \;\; \alpha_w \;\; \beta_w \;\; \gamma_w] \tag{8}$$

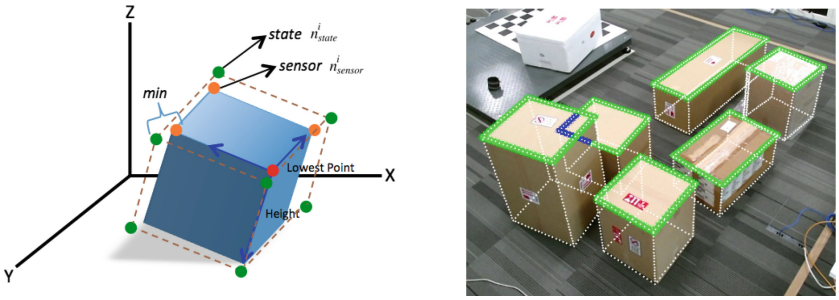


Fig. 8. Cargo Particle Filter based State optimization

$$
\begin{aligned}
X^{(k}_{n|n-1} &= X^{(k)}_{n-1|n-1} + w_{n-1} \\
P(Y_n|X^{(k)}_{n|n-1}) &= \frac{1}{\sqrt{2\pi\sigma^2}} \cdot exp(-\frac{d^2}{2\sigma^2}) \\
log(P(Y_n|X^{(k)}_{n|n-1})) &= -log(\sigma\sqrt{2\pi}) - \frac{0.5}{\sigma^2} \cdot d^2
\end{aligned} \tag{9}
$$

We have our nodes which were generated from the previous step represented as n^i_{sensor}. We generate a box model from a given hypothesis k, and we extract 4 nodes from it represented as n^i_{state}. We then attempt to minimize the $l2norm$ distance from each node i to a matching nearest neighbouring node, with the graph constraints in place (Eq. 10). This is done by reducing the weight through a multiplier β_{l2} of a particle whose set of matches don't belong to the same inner cycle set.

$$
q^{(k)}_{l2W} = \sum_i^N \left(-log(\sigma_{l2}\sqrt{2\pi}) - \frac{0.5}{\sigma^2_{I2}} \cdot \|(n^i_{state}, n^i_{sensor})\|^2_2 \right) * \beta_{l2} \tag{10}
$$

We wished to also integrate the edge information from the sobel edge images I^{RGB}_c of each camera in our particle filter step in case there were errors in the Sobel Fusion step, or if we wanted to avoid that step. This is done by sampling the top surface contour of a box $C\epsilon\{p_i..p_m\}$ generated from a given particle k, and projecting it into each camera since ${}^RT_W$ is known. We then attempt to maximize the intensity of each projected point in the sobel image space (Eq. 11). We reference the depth image I^{Depth}_c and penalize the weights of points which are occluded by other boxes through a multiplier β_I.

$$
q^{(k)}_{IW} = \sum_i^N \left(-log(\sigma_I\sqrt{2\pi}) - \frac{0.5}{\sigma^2_I} \cdot (p^i_{contour} - p^i_{sobel})^2 \right) * \beta_I \tag{11}
$$

To do this, once a particle k has reached a state where $\|(n^i_{state}, n^i_{sensor})\|_2$ is less than 1 cm for all n^i_{state}, we turn on additional weights that make use of the edge information. We found that this made our measurements more robust

to occlusion from other cargo boxes. We then add up the weights from both $q_{l2W}^{(k)}$ and $q_{IW}^{(k)}$, apply the exponential operation on them and resample with replacement (Eq. 13).

$$q_n = \frac{P(Y_n|X_{n|n-1}^{(k)})}{\sum_k P(Y_n|X_{n|n-1}^{(k)})} \quad k\epsilon\mathbb{N} \tag{12}$$

$$X_{n+1|n}^{(k)} = h(X_{n|n}^{(k)}|q_n) \quad k\epsilon\mathbb{N} \tag{13}$$

Once the dimensions of the box have been optimized, we can simply lock (l_w, w_w, h_w) in place if we want to track pose. This can be applied for multiple boxes in the scene.

5 Results

5.1 Experimental Setup

Testing was conducted on 3 different types of cargo boxes (Fig. 9), a standard cardboard box (0.665 x 0.445 x 0.370)m, a styrofoam box with rounded edges (0.460 x 0.605 x 0.300)m and a shiny shrink wrapped box (0.360 x 0.370 x 0.440)m. They were placed in the centre of the measurement workspace [4 x 3.5]m where the kinect cameras were 2.5 m high and at a 45 degree angle. Several boxes were also placed around this to simulate occlusion (Fig. 9) where at least one camera had the box covered partially.

Fig. 9. Test boxes used and tracking of multiple boxes in a pallet

In the first experiment, we generated the corner nodes n_{sensor}^{i} for all 3 boxes using the MVBB method, the Node Graph Construction method with line model fitting, and the Node Graph + Sobel Edge Fusion method. We then extracted 60 samples from the most deviant dimension from the ground truth (was always in the length/breath domain) for all 3 methods. In the second experiment, we ran our particle filter algorithm on all 3 cargo boxes initialized with data extracted from the Graph Node step, one with edge fusion turned on and one without, and extracted the dimensions from the filter.

5.2 Measurement Results

For the first experiment, from (Fig. 10), we are able to see that the MVBB approaches on our system produced mean results that deviated from the ground truth by up to 5 cm, and had significant noise especially in the plastic shrink-wrapped cargo possibly due to reflectivity from the Kinect sensor. Using a RANSAC line fitting and node graph construction approach reduced that both error and noise down significantly, but still had up to a 2 cm error in some cases. Finally, the Sobel Edge Fusion pre-processing step before application of the node graph step, reduced the error down to 0.5 cm for all types of cargo, and noise to even lower levels. This level of precision is within the bounds of allowance of the gantry robot bin-packing algorithm, and can aid in further algorithms that may track the state of boxes while they are in the pallet.

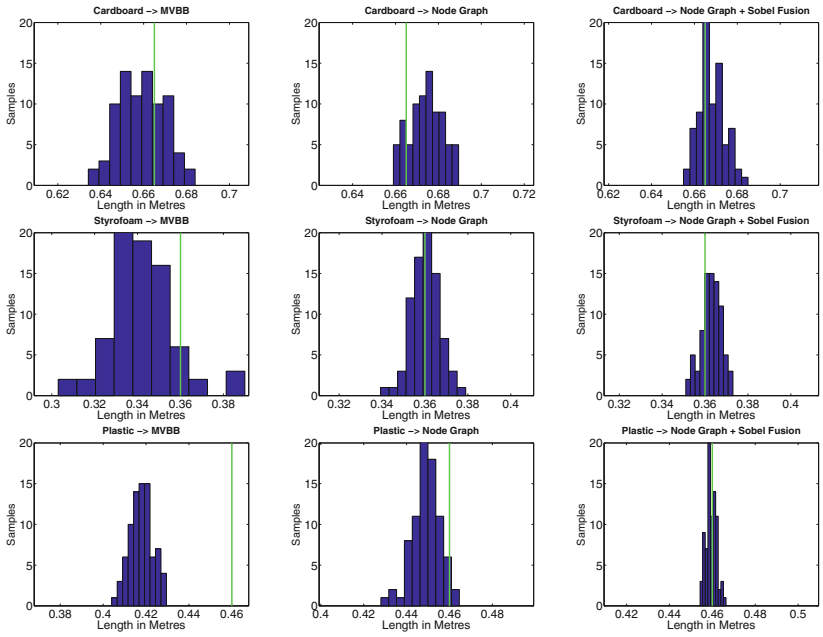


Fig. 10. Histogram of 60 samples extracted from the various methods described

For the second experiment, from (Fig. 11), you can see that for the Cardboard and Plastic cargo boxes, we see an improvement in the measurement from the ground truth in both length and breath (height was found to be always accurate). For the Styrofoam box, it was not as noticeable, possibly due to the fact that the box did not have as distinct edges in the sobel edge image. In all 3 cases however, we were able to achieve our accuracy requirement of 1 cm or below.

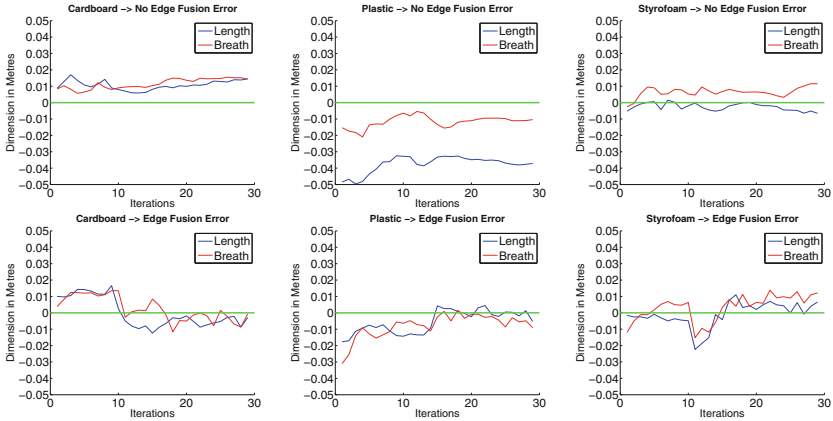


Fig. 11. Running particle filter optimization with and without edge fusion

5.3 Conclusion and Future Work

While using the Node Graph + Sobel Edge Fusion step was more accurate with enough samples taken, the Node Graph + Particle Filter with Edge fusion was more robust. Our solution allows one to place unknown cargo boxes of various materials inside a scene, and achieve sub-centimetre level precision pose and dimension extraction even with commercial grade sensors at a range of over 3.5 m, and allows a gantry robot and bin packing software to work with minimal constraints. These algorithms were run on an Intel Core i7-4790 with 4 physical cores and a NVIDIA Geforce 760 Ti. We were able to optimize up to 12 boxes in the scene at approximately 4 Hz. In the future, it may be possible to extract features from the cargo boxes, and use that to track the cargo box pose more robustly.

Acknowledgement. This work is funded by the Civil Aviation Authority of Singapore (CAAS) under the Aviation Challenge 2 grant.

References

1. Har-peled, S.: A practical approach for computing the diameter of a point set. In: SCG 2001 (2001)
2. Boeing: boeing pallets @ONLINE (2012). http://www.boeing.com/resources/boeingdotcom/company/about_bca/pdf/CargoPalletsContainers.pdf
3. Viegas, J.P.L., Vieira, S.M., Sousa, J.M.C., Henriques, E.M.P.: Metaheuristics for the 3D bin packing problem in the steel industry. In: Proceedings of the 2014 IEEE Congress on Evolutionary Computation, CEC 2014, pp. 338–343 (2014)
4. Robotics, U.: Universal robotics @ONLINE (2015). http://www.universalrobotics.com/applications/
5. Drost, B., Ulrich, M., Navab, N., Ilic, S.: Model globally, match locally: efficient and robust 3D object recognition (2012)

6. Holz, D., Behnke, S., Holz, D., Topalidou-kyniazopoulou, A.: Real-time object detection, localization and verification for fast robotic depalletizing verification for fast robotic depalletizing. In: IROS (2015)
7. Lloyd, R., McCloskey, S.: Recognition of 3D package shapes for single camera metrology. In: 2014 IEEE Winter Conference on Applications of Computer Vision, WACV 2014, pp. 99–106 (2014)
8. Alexa, M., Behr, J., Cohen-Or, D., Fleishman, S., Levin, D., Silva, C.T.: Computing and rendering point set surfaces. IEEE Trans. Vis. Comput. Graphics **9**, 3–15 (2003)
9. Richtsfeld, A., Morwald, T., Prankl, J., Zillich, M., Vincze, M.: Segmentation of unknown objects in indoor environments. In: IEEE International Conference on Intelligent Robots and Systems, pp. 4791–4796 (2012)
10. Wu, K., Ranasinghe, R., Dissanayake, G.: A fast pipeline for textured object recognition in clutter using an RGB-D sensor. In: 2014 13th International Conference on Control Automation Robotics and Vision, ICARCV 2014, pp. 1650–1655 (2014)
11. Somani, N., Cai, C., Perzylo, A., Rickert, M., Knoll, A.: Object recognition using constraints from primitive shape matching. In: 10th International Symposium on Visual Computing (ISVC 2014) (2014)
12. Anwer, A., Baig, A., Nawaz, R.: Calculating real world object dimensions from Kinect RGB-D image using dynamic resolution. In: Proceedings of 2015 12th International Bhurban Conference on Applied Sciences and Technology, IBCAST 2015, pp. 198–203 (2015)
13. Aouada, D., Ottersten, B., Mirbach, B., Garcia, F., Solignac, T.: Real-time depth enhancement by fusion for RGB-D cameras. IET Comput. Vision **7**, 335–345 (2013)
14. Wang, H., Zhang, W., Chen, Y., Chen, M., Yan, K.: Semantic decomposition and reconstruction of compound buildings with symmetric roofs from LiDAR data and aerial imagery. Remote Sens. **7**, 13945–13974 (2015)
15. libfreenect2 @ONLINE (2013). https://github.com/OpenKinect/libfreenect2
16. iai-kinect2 @ONLINE (2015). https://github.com/code-iai/iai_kinect2
17. Quigley, M., Conley, K., Gerkey, B.P., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A.Y.: ROS: an open-source robot operating system. In: ICRA Workshop on Open Source Software (2009)
18. Zhang, Z.: A flexible new technique for camera calibration (technical report). IEEE Trans. Pattern Anal. Mach. Intell. **22**, 1330–1334 (2002)
19. Bradski, G.: Opencv. Dr. Dobb's journal of software tools (2000)
20. Rusu, R.B., Cousins, S.: 3D is here: Point Cloud Library (PCL). In: IEEE International Conference on Robotics and Automation (ICRA), Shanghai, China (2011)
21. Merry, B., Gain, J., Marais, P.: Moving least-squares reconstruction of large models with GPUs. IEEE Trans. Vis. Comput. Graphics **20**, 249–261 (2014)
22. Johnson, D.B.: Finding all the elementary circuits of a directed graph. **4**, 77–84 (1975)
23. Fox, D., Burgard, W., Dellaert, F., Thrun, S.: Monte carlo localization: efficient position estimation for mobile robots dieter fox, wolfram burgard. In: 16th National Conference on Artificial Intelligence (AAAI99), pp. 343–349 (1999)