Volumetrics - Measuring free volumes

André Carreira, Rodrigo Ventura, José Gaspar
Instituto Superior Técnico / UTL, Lisbon, Portugal
andre.rq.carreira@gmail.com, rodrigo.ventura@isr.ist.utl.pt, jag@isr.ist.utl.pt

ABSTRACT

Novel applications emerge as new, more precise and lesser expensive technologies of 3D scanning are becoming available in the market. The recently launched Microsoft-Kinect camera, based on the Prime Sense sensor, is an excellent example of such a 3D scanning technology. Despite having been designed for the computer games market, it can launch a number of successful, cost-effective and industrial applications.

The information provided by these depth cameras is commonly fused with well known tools, like the Iterative Closest Point (ICP) algorithm and volumetric computational algorithms. The objective of this work is therefore the application of these tools in the creation of two industrial applications (systems), using geometric information in order to speed up their computation and increase their precision.

The first application encompasses the development of a system that detects if the containers transported by a conveyor belt have some kind of residues. Normal information is used in this application for increasing computational time of the box identification step while image processing tools will adjudge its classification. In the second application, an automatic free volume estimator, normal information will be used in order to increase the system's precision as it will regulate the point matching step. A volume computational algorithm will be used and its precision will be analysed. The reliability of the applications will be studied based on a set of carried out experiments involving simulated and real data. Experiments have been carried out involving simulated and real data within both applications.

Keywords: Color-depth camera, 3D pose estimation, 3D volume estimation.

INTRODUCTION

Real time 3D recognition has become one of the most important and promising fields of robotics. Range images obtained from lasers and RGB-D cameras have growth in popularity and found numerous applications in fields including medical imaging, object modeling and robotics (obstacle avoidance, motion planning, mapping, etc.). One of the factors which led to this popularity was the appearance in the market of the Microsoft Kinect.

The Microsoft Kinect has completely changed the world of computer vision due to its unprecedented low price and its resolution 480x640 of video and depth data captured up to 30 fps. It came as a low accuracy alternative to the expensive laser scanners in applications where accuracy requirements are less strict. Kinect depth sensor is a structured light $3D$ scanner. Its infrared light source projects light patterns to the space ahead. The reflected light is then received by an infrared camera and processed to extract geometric information about object surfaces as their distance to the camera plane is computed. By merging this information a colored point cloud can be computed, containing up to 307200 points per frame. Still, Kinect's depth map resolution and accuracy can be a problem. In fact, the data provided present different noise-related issues which narrow its future applicability. The analysis of the systematic and random errors of the data is indubitably necessary to realize the full potential of the sensor for mapping and other applications.

The work described within this dissertation focuses on the analysis of the Kinect reliability as it is in the development and testing of two volume analysis systems of industrial purposes.

A. Motivation

The application of RGB-D technology for industrial purposes is recent. Companies are motivated and see this technology as a useful tool to control and support their information systems.

Two applications for this technology were purposed by two different companies:

Empty volume verification: a system that verifies if container is empty or partially filled when passing in a conveyor belt system. Conveyor belts are commonly used in many industries as they are able to safely transport materials from one level to another which, when done by human labor, would be strenuous and expensive. RGB-D cameras can be used in order to study efficiently and autonomously the properties of the passing objects. This application addresses the design of an automatic, fast and efficient system to adjudge if small containers moving in a conveyor system are empty.

Free volume computation: a system that allows a precise measurement of the free volume of a truck container. Transporting goods using trucks is ubiquitous. As Cargo transporters are constantly seeking ways of optimize their business by augmenting the transported volume per truck. Assessing the empty volume of a truck container is therefore a key tool for business optimization. This application addresses the design of an automatic free volume estimator applied to truck containers using recent high resolution cameras or depth sensors. For purposes of testing, the Kinect will be used in the first phase of the project.

Fig. 1: (a) Environment of application 1 (b) shows the concept of application 2

B. Related Work

In order to compute a colored cloud point, a previous calibration of the cameras is required since there's a physical distance between them. [2] provides Kinect calibration tools for a Matlab environment. [4] provides an insight into the factors influencing the accuracy of the data is provided and experimental results are shown, proving that the random error of depth measurements increases with increasing distance to the sensor, and ranges from a few millimeters up to about 4cm at the maximum range of the sensor. [7] proposes an algorithm for correcting gaps created due to errors in scanning.

Due to the heavy sheer volume of data generated by depth cameras, processing this data in real time has become a challenge. For this purpose, sampling methods are used in order to prune the non-relevant data from the depth image. [11] introduces the Fast Sampling Plane Filtering (FSPF), a method which samples points from the depth image, classifying local grouped set of points as belonging to planes in 3D. Other sampling methods were used in this work in order to filter points which belong to planes of interest. This sampling methods allow the use of computationally expensive 3D localization and model recognition algorithms. The most known of these algorithms, the Iterative closest point (ICP) algorithm is used in this work to compute the location of known objects in the observable environment, allowing to explore their volume properties thereafter [1], [3]. The ICP algorithm presented in the early 1990s has become the most popular algorithm to infer the change in 6D pose of a camera and the rigid transformation between two point clouds. It has been the target of intense research and many variants have been developed [9], [5], [8], [13], [12]. Some variants of this algorithm are hereby explained and compared in section 4.

C. Problem Formulation

Although being substantially different, the two applications studied and developed in our work have a significantly common overlap in terms of the required algorithms. In both cases the depth image provided by the RGB-D cameras is filtered for spurious data and the ICP algorithm is then applied to the result in order to identify the position of an open container. Finally, after having identified the position of the container, the volume properties are analyzed.

This thesis proposes a solution for both applications and analyzes the efficiency and precision of the used algorithms. The main steps of the work are therefore the following: (1) data acquisition of depth maps;(2) filtering the non-relevant data; (3) 3D recognition of an open container; (4) implementation of the empty box verification system (5) Precision analysis empty box verification system (6) Implementation of the free volume estimator (7) Precision analysis of the free volume estimator.

D. Report Outline

In Section 1 we start by introducing the objective of this dissertation and by presenting a short introduction of the state of the art on depth cameras. Next, in Section 2, we proceed to present the pinhole model as the model considered in this work. Here we describe the relation between 3D spatial coordinates of an object and its projection unto the screen coordinates. Section 3 describes the methods used to filter the point cloud in order to identify the points that are likely to belong solely to the top, down, right and left side of the container. Here, the depth image blank spaces are filled by applying a simple filling algorithm, the points with desirable normals are identified and their 3D coordinates are obtained. Furthermore, after having pruning the points of interest, in Section 4 we present an overview of the ICP algorithm as well as some variations used to identify the container in the frame. Here it is described the first application, which finds a container in space and adjudges if it is empty. An analysis of its performance is here also presented. The algorithm used in order to compute the free volume is presented in Section 5, as well as the experiment results and precision analysis. Section 6 summarizes the work performed and highlights the main achievements in this work. Moreover, this section proposes further work to extend the activities described in this document.

I. Color-depth Cameras

Color-depth cameras, also know as RGB-D cameras, provide real-time color and depth data which can be useful in many applications. The depth data is returned as a matrix whose entries represent the distance between the image plane and the corresponding point in the 3D environment.

In this section the geometric relationship between the coordinates of a 3D point and its projection onto the image, screen coordinates, is explained using the pinhole model.

A. The Camera Model

Although real cameras have a lens that focuses light and therefore are better approximated by the thin lens camera model, throughout this work we will consider a pinhole camera model, in which no lenses are used to focus light. The pinhole camera model describes a mapping between the 3D coordinates of a feature and the feature's projection into the image plane of an ideal pinhole camera, where the cameras aperture is described as a single point (optical center) which every light ray goes through. This model, even though disregarding effects like geometric distortions of real cameras, is a good approximation to how the projection works.

To simplify and avoid confusions, it is common in computer graphics to consider an image plane that lies between the

optical center and the 3D scene. This way the result is an unrotated image. This virtual image plane is placed so that it intersects the Z axis at f, focal distance, instead of at -f. In this model, by triangle symmetry we get the relations:

$$\frac{X}{Z} = \frac{x}{f} \text{ and } \frac{Y}{Z} = \frac{y}{f} \tag{1}$$

or in a more practical characterization:

$$\begin{bmatrix} \lambda x \\ \lambda y \\ \lambda \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}^C \tag{2}$$

where λ is the distance of the point to the image plane. This can be expressed a homogeneous matrix form and

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}^C = \widetilde{M}^C$$

is the coordinates of the point in the camera coordinates system expressed in Cartesian coordinates. λ represents a scale factor and it is needed in order to compute the real coordinates. This is done by simply dividing the left side of the equation by λ.

1) From a World Coordinate System: In order to compute the coordinates of a point in the image not from the world coordinates system W, one needs to apply the transformation (see eq.3) to the coordinates in the world referential.

$$\begin{bmatrix} X^C \\ Y^C \\ Z^C \\ 1 \end{bmatrix} = \left[\begin{array}{c|c} R_W^C & t_W^C \\ \hline 0 & 1 \end{array}\right] \begin{bmatrix} X^W \\ Y^W \\ Z^W \\ 1 \end{bmatrix} \quad . \tag{3}$$

By using eq.2 and eq.3 we contain the relationship between 3D point (in world coordinates) of a feature of an object and the coordinates of its projection into the image frame. Here $\widetilde{R_W^C}$ and $\widetilde{t_W^C}$ represent the rotation and translation between the referentials and $R_W^C \in \mathbb{R}^{3\times3}$, $t_W^C \in \mathbb{R}^{3\times1}$ and the 0 represents a zero matrix $in\mathbb{R}^{1\times3}$. Thus, the relationship between 3D point (in world coordinates) of a feature of an object and the coordinates of its projection into the image frame is given by:

$$\begin{bmatrix} \lambda x \\ \lambda y \\ \lambda \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \left[\begin{array}{c|c} {}^C R_W & {}^C t_W \\ \hline 0 & 1 \end{array}\right] \begin{bmatrix} X^W \\ Y^W \\ Z^W \\ 1 \end{bmatrix} \quad . \tag{4}$$

The image frame coordinates are here given in meters, the same unit used for the 3D coordinates. We will now proceed to compute its coordinates in pixels.

2) From Meters to Pixels: In order to compute the image plane projection in pixels, some adjustments are needed. Here the parameters necessary to link the pixel coordinates of an image point with the corresponding coordinates in the camera reference frame do not depend on the camera's location, but on its own specifications, like Focal lenght, CCD dimensions and principal point. If the origin of the 2D image coordinate system does not coincide with where the Z axis intersects the image plane (principal point), we need to translate the projection of the point P_0 to the desired origin. This translation will be defined by (u_0, v_0). Next, its u and y coordinates will be multiplied by the scale factors α_u and α_v respectively so as to obtain the point in pixels.

Considering the skew coefficient, γ, we can express these relations in matrix form as:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad . \tag{5}$$

We can now update eq. 4 so that we obtain the projection equation from world coordinates to the image frame pixels:

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha_u f & 0 & u_0 \\ 0 & \alpha_v f & v_0 \\ 0 & 0 & 1 \end{bmatrix} \left[\begin{array}{c|c} R_W^C & t_W^C \\ \hline 0 & 1 \end{array}\right] \begin{bmatrix} X^W \\ Y^W \\ Z^W \\ 1 \end{bmatrix} \quad . \tag{6}$$

Considering the projective camera, the column, which multiplies with and the line itself of the rotation and translation matrix, can be occluded to simplify the model itself. This equation is often mentioned in its most simplified form:

$$\lambda m = PM \quad . \tag{7}$$

Where $P \in \mathbb{R}^{3\times4}$ is the projection matrix composed from the intrinsic matrix specific to every camera and the extrinsic camera which defines the location and orientation of the camera reference frame with respect to a known world reference frame, $\left[\begin{array}{c|c} R_W^C & t_W^C \end{array}\right]$.

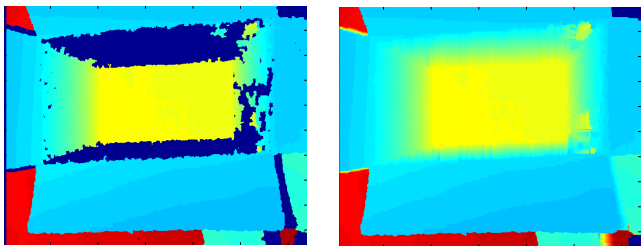
GETTING THE POINT CLOUD OF INTEREST

B. Overview

In this section we explain the mean to obtain a filtered point cloud which contains the points of interest which will be analyzed later in order to find the correct box position. The output of the Ms Kinect camera consist of a matrix in which each pixel value is the distance between the corresponding 3D point and the camera plane. Errors in readings by the Ms Kinect cameras can be presented as areas of no information on the depth image. Some preprocessing is needed.

An estimation of its lost information must be obtained in order to compute the volume. Inpaiting algorithms for these shadows have been a subject of research [7].

A simple estimation method based on the specific geometry of the problem will be here applied. Later, in order to apply recognition methods we need a point cloud of the area of interest as such, in this section we explain how to obtain a point cloud from the depth matrix. Finally, in order to find areas of interest, we filter the point cloud by normal properties.



Raw depth image Filling algorithm output

Fig. 2: Filling algorithm results

C. Preprocessing the Depth Image

Structured light sensors measurements, such as the Microsoft Kinect's, are affected by noise due to multiple reflections, transparent objects or scattering in particular surfaces. This contributes to the presence of regions for which the camera is not able to estimate the depth.

For each set of points in the shadow, the length of both the horizontal and vertical intersecting vectors, which contain no information, was computed. By choosing the direction in which the vector was shorter, a linearly spaced vector would take its place taking into account the depth intensity of both extremities and its size. This application was experimented on boxes which present continuous slope along its sides and do not contain highly irregular volumes, these facts contributed for the good results of the used algorithm.

D. Depth Image to Point Cloud

Cameras project 3D points into the image plane. This transformation is not invertible unless we have some additional information about the point location in space.

Depths cameras provide the depth value for every pixel. This depth information, along with the camera intrinsics (horizontal field of view f_h, vertical field of view f_v, image width w and height h in pixels), can be used to reconstruct a 3D point cloud.

The pin-hole model introduced in section 2 will be considered as the model throughout all the project.

Let the depth image of size $w \times h$ pixels provided by the camera be d, where $d(u, v)$ is the depth of a pixel at the location (u, v). We intend to find the corresponding 3D point $p = (p_x, p_y, p_z)$.

Using eq. 2 and eq. 5 we obtain the projective system which relates the pixel position in the image frame with its position in the 3D coordinate system of the camera itself.

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X^C \\ Y^C \\ Z^C \\ 1 \end{bmatrix} . \quad (8)$$

Let for each pixel (u, v), $\lambda = d(u, v)$. Hence:

$$\begin{cases} X = \frac{du - u_0 Z}{\alpha_u f} \\ Y = \frac{dv - v_0 Z}{\alpha_v f} \\ Z = d \end{cases} \quad (9)$$

The 3D points coordinates were created in three different matrices in order not to lose the relative projected spatial information. These were reconstructed using the depth value $I(d)$:

$$\begin{aligned} X(u, v) &= d(u, v) \frac{u - u_0}{\alpha_u f} \\ Y(u, v) &= d(u, v) \frac{v - v_0}{\alpha_v f} \\ Z(u, v) &= d(u, v) \end{aligned} \quad (10)$$

E. Computing the Normal

As most computers cannot process the full 3D at a frame rate, a pruning of non-relevand data is required in order to apply the matching algorithm in the next section. The points that are likely not to belong to the box's sides will be removed. This pruning takes into account their expected geometric properties, namely, their normal vectors. Considerating that the box is in a stable position, its sides are relatively vertical while its upper and bottom are relatively horizontal. Therefore, the normal vector of these sides will surely have a big horizontal or vertical component, respectively.

1) Partial Derivatives Method: By computing the gradient of the Z matrix in two orthogonal directions, the resulting values u and v will correspond to the magnitude in both directions tangent to the surface. In 3D a cross product between two vectors will return their orthogonal complement. Applying a cross product to u and v will then give us the intensity value of the Z coordinate of each point. In order to Obtain all the normal coordinates, this method is applied to X and Y. Although this method does not provide us with the knowledge of planes in the image as would the Fast Sampling Algorithm [11], it has proven to be faster and more precise.

F. Filtering the Point Cloud

Having found the normal vectors of the points in the frame, we can now identify the points of interest. In this work we considered them to have a normalized horizontal or vertical component of the normal vector bigger than 0.9. This intensity of the main normal component allows some versatility of the accepted angles.

The results of the used algorithms can be observed in fig. 3. The white points of the image represent the points of interest, points which present a normal horizontal or vertical component higher than the accepted threshold.

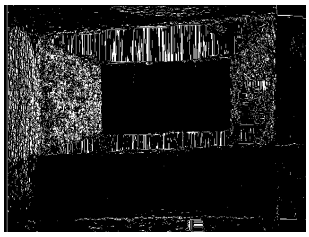


Fig. 3: Points of interest (in white) given by the used algorithm

The subsampling algorithm considered has proven to be a faster and more accurate alternative to the FSPF algorithm for

finding the normals of all the frame points. However, the last algorithm used proved to be more precise and even faster.

As a result of applying the filter, points which do not meet the geometric specifications were pruned and their 3D coordinates will not be considered for the 3D matching presented in the next section. The Depth image captured by the Kinect contains 307200 points (480×640). In figure 3 the total points of interest identified are 30868, which corresponds to approximately 10% of the total number of points. This will increase considerably the speed of the match algorithm.

II. Finding Boxes in Point Clouds

A. Overview of the ICP Algorithm

Introduced by Chen and Medioni [3] and Besl and Mckay [1], Iterative Closest Point (ICP) is an algorithm employed to find the rigid transformations, which minimizes the difference between two clouds of points that relies on an initial estimate of the relative pose between the two point clouds in order to achieve the global solution, which minimizes the error metric.

By giving as inputs to the ICP, a world scan, a 3D model of an object and an estimate of its initial position, ICP will compute the rigid transformation between them which minimizes the error metric, providing us with an hypothesis of the object position in the world (model based tracking). [9] provides a recent survey of many variants of the algorithm. A good initial estimate of the position of the model in the world is essential to the convergence of the algorithm to a global minimum of the cost function.

The ICP algorithm can be decomposed in three main stages:

1. **Selection** of used points in both meshes;
2. **Matching** the points between both point clouds;
3. **Minimizing** the error metric in order to achieve a estimate of the rigid transformation given the matched points.

The transformation which minimizes the error metric is then applied to the model point cloud and this process is repeated cyclically until a stopping criteria is met.

The key to the solution of this problem is to achieve the correct correspondences between point clouds since their correct relative Rotation/Translation can then be computed in closed form.

A more detailed analysis of its stages can be observed bellow.

ICP starts by applying an initial estimation to the transformation T between two point clouds.

Assuming one wants to find a correspondence of the point cloud $B \in \mathbb{R}^{3\times n}$ in the point cloud $A \in \mathbb{R}^{3\times k}$, the matching consists of finding for each point $T \times B_i$ the nearest point in A, m_i. Considering a minimum distance, the found correspondence will only be considered if its distance is less than the specified threshold.

Finding a transformation between matched points will consist in minimizing the cost function. This cost function varies, the simplest form is an euclidean distance between two points (as seen in the algorithm above). Other alternatives are considered in this work taking into account spacial properties of these matched points.

Algorithm 1 regular ICP Algorithm

$T \leftarrow T_0$
while (not converged) **do**
  for $i = 0$ to $serverIndex.length$ **do**
    $m_i \leftarrow FindClosestPointInA(T \cdot b_i)$
    if $\|m_i - T \cdot b_i\| \leq d_max$ **then**
      $m_i \leftarrow 1$
    else
      $m_i \leftarrow 0$
    end if
  end for
  $\arg\min_x \{\sum_i w_i \|T \cdot b_i - m_i\|^2\}$
end while

With the first estimation of the transformation computed, the transformation is applied to B, the point cloud we desire to match, and the iterative process repeats until a satisfying condition is met. Although the weights, $w \in \mathbb{R}^{1\times k}$, which reenforce some correspondences, are considered in the algorithm presented above, these were not considered in the work.

B. Stages of the ICP Algorithm

As mentioned previously, the ICP algorithm can be broken down in three main stages which have their own variants, as they have been focus of intense research. We will now proceed to describe in detail the variants used in these applications.

1) Point Selection: While [1] uses all available points in both meshes for the ICP algorithm, [10] performs an uniform sampling of all the available points and [6] performs a random sampling of the existing points in each iteration. In order to avoid a combinatorial explosion of the number of possibilities, *a priori* knowledge about the object's position and its color [12] or spatial [8] properties can be used. In both applications the point selection phase is accomplished by the filter of non-relevant points. In application (i) the points considered are achieved through a background subtraction while, in application (ii), only the points which obeyed the spatial condition of belonging to vertical or horizontal planes were selected. The model was hereby considered bottomless so it can be used even when the box is partially filled.

2) Point Matching: In each iteration, for each point $T.B_i$, we want to find the corresponding closest point in A in order to compute the new transformation estimative, where $B = [p_1, p_2, ..., p_n] \in \mathbb{R}^{3\times n}$ is the model point cloud, $A = [p_1, p_2, ..., p_m] \in \mathbb{R}^{3\times m}$ is the world point cloud, result of the previous pruning, and T is the estimation of the transformation between meshes.

Depending on the number of points generated by the range sensor, it might be reasonable to make a selection of these points, so as to compute the box location in real time.

It also turns out that some matches are more suitable than others as it is more likely to identify a correct match for them. Which method to use is this selecting is therefore strongly dependent on the kind of data being used and should be considered for each specific problem.

Since in the previous section the filtered point cloud only contains points of two classes: points with normalized

horizontal normal coordinate > 0.9 and points with normalized vertical normal coordinate > 0.9. Following a somehow similar aproach to [8], points with normals nearly orthogonal were not allowed to be compared. The computational time required by this method is greatly lower than the Exhaustive search alone. It is worth pointing out that even though the speeds of convergence increases so will the precision, since unlikely matches will decrease in number.

3) Computing the Rigid Transformation: After the points have been selected and matched, a suitable error metric will be assigned and minimized in order to find the transformation estimate. The point to plane metric [3], which takes into account the normals of each corresponding point, proved to be better suited for these applications than the more straightforward point-to-point metric [1]. Identification algorithms will not be used in both applications since a rough initial alignment is always present due to their specifications.

C. Point-to-Plane

Taking into consideration the normal properties of the plane in the cost function, the object of minimization of this error metric is the sum of squared distance between a point and the tangent plane at its corresponding point.

$$T = \underset{T}{\operatorname{argmin}} \sum_{i=1}^{n} \|\eta(a_i - (Tb_i))\|^2 \tag{11}$$

This makes particular sense when we are dealing with planes since it lets flat regions slide along each.

There are no closed-form solutions for this metric. One way of solving the least-squares equation is by linearized the problem, assuming that incremental rotations are small ($\sin(\theta) \sim \theta$ and $\cos(\theta) \sim 1$) between both input surfaces ([5]).

III. Application 1: Testing Empty Boxes

As a viable solution a human based alternative, this automatic system consists of an empty volume binary classifier, intended to be installed on top of a conveyor belt in which small containers are being transported. These containers present sometimes residues which need to be extracted when present. Velocity of computation and precision of the results are mandatory.

In order to obtain the model needed to the ICP computation, a digitization of a container is required. This is achieved by taking two depth images: one background image and one image with an empty container. By using a simple background subtraction, followed by a light erosion to erase outliers one obtains the container model. The points of the depth image $d(u, v)$, which belong to the box, are identified. Lastly, the identified points are transformed into a point cloud using eq.9 (fig. 4 (b)).

1) Error Analysis: Taking into account that the containers are passing through the camera in a line shaft driven roller conveyor, the background can be considered static, and therefore, a simple background subtraction is enough to identify the points which belong to the container.

Due to the imprecision of the Kinect camera the result off this subtraction presents points in the background with non zero value, in order to eliminate them in this filtering process, a slight erosion of the result can be applied. This erosion is used to filter small group of isolated points, but will consequently partially filter some of the containers' points. The precision of the ICP result will not be affected by the erosion and its convergence speed is actually going to increase since less points of the box will be considered in each iteration.

One way to avoid outliers during the autonomous activity of the system would be considering the use of a protective shell to enclose the camera view and the boxes which pass through it. This enclosure would also prevent errors caused by external movement in the surroundings of the containers and would have other advantages which will be afterward explained in the error analysis section.

After having filtered the Depth image, we can now use the ICP algorithm in order to successfully identify the box position in the world. The ICP algorithm requires an initial transformation, which correspond to the initial estimate of location of the container in the image. For this purpose, due to its good localization, the same Depth image, in which the model was obtained, was used as a reference for initial position estimation just by converting it to a point cloud - the 3D point cloud of the model was already in the initial position estimate, as such, the initial rotation considered is an identity matrix and the translation will be zero.

At this point the corners of the interior part of the box were chosen from the Depth image, their 3D coordinates were obtained and saved. These points are going to be used once the container's position is found. Considering that no exterior noise was provoked due to movement in the vicinity of the conveyor, there are three different situations which can be concluded based on the results:

1- If the points of the image obtained after the background subtraction do not meet the minimum amount, the camera is not capturing a box or only a small part of the box is appearing in the frame. In this case another scan is immediately executed.

2- If the error of the ICP exceeds a given threshold, it can be concluded that once again not all the box was found in the image and another scan is then performed. A high error indicates that part of the model could not be correctly overlaid onto the resulting world point cloud and therefore the box is partially occluded or the object in the frame is not similar to the container model.

3- If the number of filtered points meet a minimum and the error of the ICP is low enough, the box was found and both translation and rotation, which map the model into its position in the world, are obtained.

By applying the obtained rotation and translation to the initially saved position of the interior corners of the model we achieve their final position in the identified space.

As we intend to analyze the interior of the box, not in the

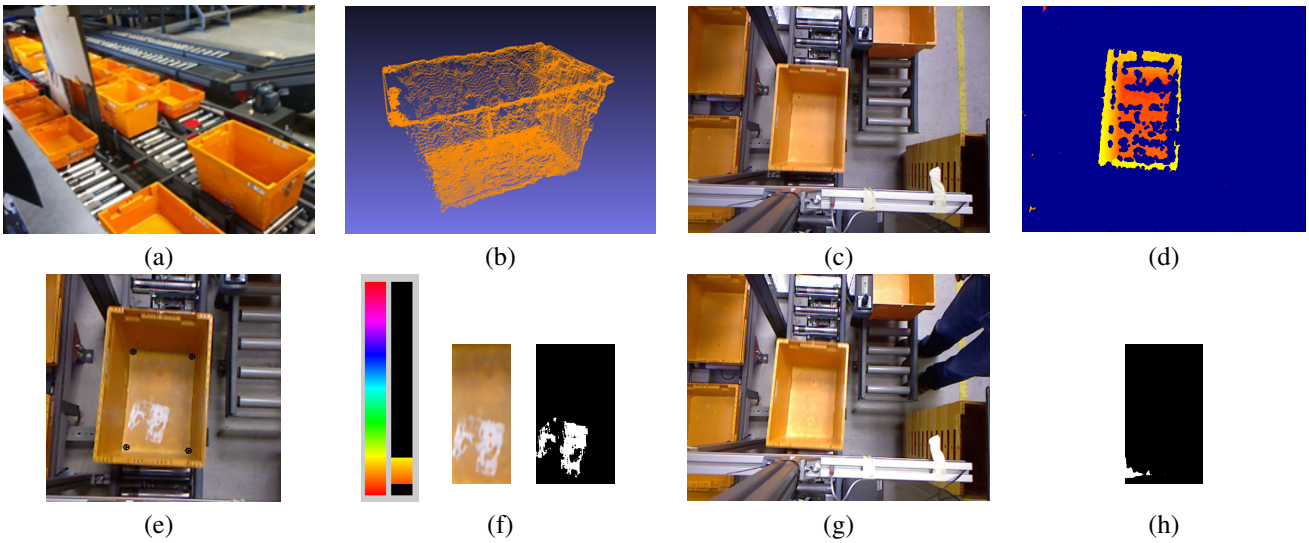


Fig. 4: (a) shows the application 1 environment, (b) illustrates the container's model, (c) RGB image retrieved by the kinect, (d) shows the filtered depth image, In (e) is the result of the model's identification, (f) shows the admitted hue and, in white, the identified object, in (g) a wrong classification due to a strong light, (h) shows the wrongly identified object of that classification

Depth image but in the RGB image, a transformation between their image coordinates is required. Using the transformation (rotation and translation) between cameras obtained in their calibration [2], the 3D position of the points in the RGB coordinates will then be computed and a projection using the intrinsic parameters of the RGB camera will provide us with their coordinates in the image. Fig. 6 shows a similar approach between the world 3D coordinates and the Depth image position. Having obtained the points corresponding to the inside corners of the container in the RGB image (fig.4 (e)), by performing a crop and a projective transformation, we obtain only the inside of the container, as seen in fig.4 (f).

Lastly, we intend to determine if a container is empty. For that two metrics were used and both took into account one of the main attributes of color - the hue. The hue of an RGB image can be computed through a HSV transformation. Knowing the box's color, a short interval of accepted hues can be established in order to filter objects of different colors (see fig. 4 (f)).

The percentage of space occupied by points of hue which do not belong to the specified interval (points in white in fig. 4 (f)) is computed and then compared a threshold. This threshold represents the maximum percentage of occupied space admitted for boxes classified as empty.

In case the occupied space percentage is less than the used threshold, a second verification is performed, in which the total sum of intensity of the gradient of the hue is taken into account. This second threshold was considered in order to detect transparent plastics, which even though they create changes in hue, this values remain inside the admitted interval of acceptance.

In this work, by convention, a filled box is declared as being a positive classification and an empty box was formally known as a negative classification. As previously mentioned, not only the percentage of space occupied by points of different hue value, but also the sum of the gradient of the hue were considered to the analysis of the cropped image. Therefore, the values of these two thresholds influence the classification. The performance of the presented binary classifier system, as its discrimination threshold is varied, was tested through data of three sets of test scans. As it was empirically observable, the second threshold (sum of hue gradient) did not increase the accuracy of the system when the first threshold - percentage of occupied space - was well established. Based in these three performed tests, an optimal model point was selected which correspond to the used thresholds. This point was selected with the purpose of obtaining a perfect classification of occupied boxes while simultaneously decreasing the false positive (empty box detected as being filled) and the false negative (filled box detected as being empty) ratios. In the tests, 5536 frames were analysed, from which 561 frames showed 56 individual fully visible containers. No boxes with garbage were classified as being empty (false negatives) although in the third test four empty boxes were considered filled. The false positives events were found to be consecutive and in all a strong light was present. This light, caused by an external event, dragged the hue of part of the striked area to values outside the admitted interval (fig. 4 (g) and (h)).

The idea of a protective shell, introduced in the ICP section, would bring another advantage as it would avoid the sensitivity to external light, therefore, increase the reliability of the system. Although further testing is advised, the Kinect has proven to be a reliable sensor for such an application and the method introduced satisfied the requirements.

IV. Application 2: Computing Volumes

In this case of study of industrial applications for the RBG-D technologies the objective is to study volume properties of known objects, specifically, truck containers. Subsequently study the accuracy of a possible system to determine the

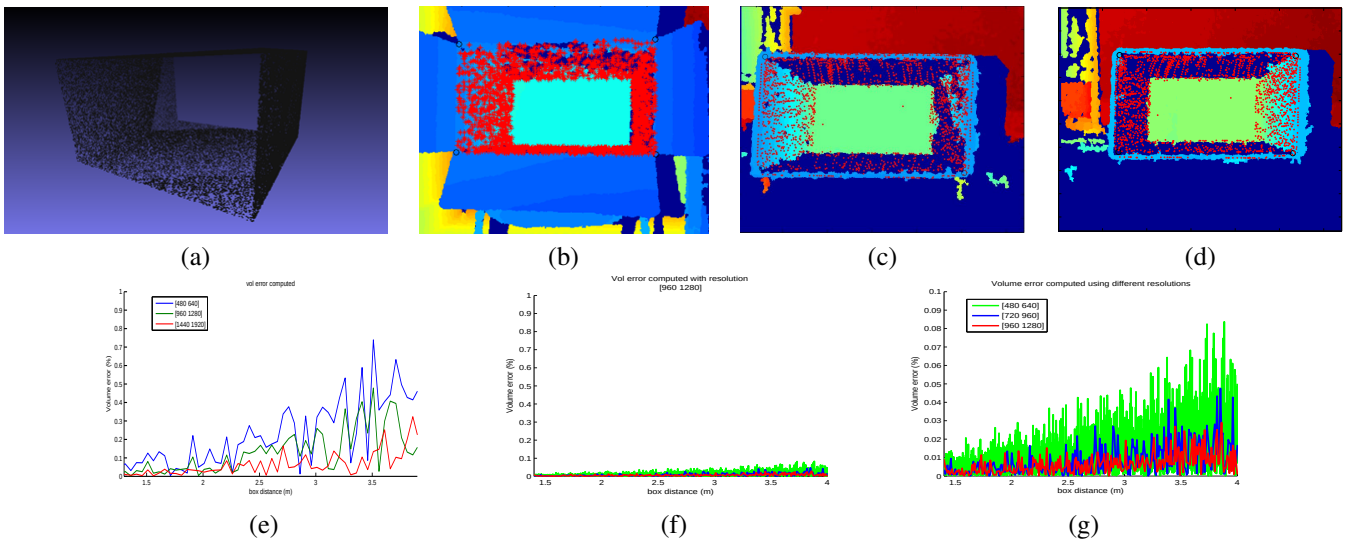


Fig. 5: (a) The used model for application 2 (b) an identification example (c) distance=50cm, Volume error=0.697% (d) distance=60cm, Volume error=0.734% (e) The effect of resolution on volume computation (f) The effect of noise on volume computation at the same scale has the resolution effect graph (g) Computational error using depth with noise with different resolutions

volume of the container in a specific frame captured in a depth map. The solution to this application presents itself as having the same main structure as the Testing Empty Volumes application. This application follows similar approach to the previous: a 3D model of the object is digitalized (1); The capture of the RBG-D image is followed by pruning of non relevant points (2); the object's location is given by the ICP (3); and its content will at last be analyzed (4).

An analysis of its results using the Kinect camera will be described in the end of this section. Due to the characteristics of the project, in a realistic scenario, the Depth image would need to be captured by a camera with long range accuracy (e.g. laser sensors). In the created testing environment, an empty box made of corrugated fiberboard was used as a container.

An estimate of information lost in the shadows areas of the depth image is required. The filling algorithm introduced in section 3.2 was used.

After having computed the 3D points, the normal matrices can now be computed using the method of the partial derivatives since this proved to be fast and precise. As explained in the Section 3.5, in order to speed up the ICP algorithm, only points which obey to the imposed spatial and color specification will be considered while the points of the bottom of the box were not considered as to avoid possible errors in the ICP matching step caused by objects in the container. Due to its higher convergence speed the matching step of the ICP had into account the normal of the points, not allowing points with different main normal components to be compared while the Point-to-Plane was the minimization metric used.

The ICP algorithm stops once a condition is satisfied. This condition can be either a threshold of relative error reduction between iterations or a threshold of error itself. In this application a maximum number of iterations was specified. Once the stopping criteria has not been met in this number of iterations, the ICP stops, the volume computation algorithm is not applied and another frame is analyzed.

If, by the contrary, the box is identified by the algorithm and the stopping criteria is met, then the volume is, at last, computed.

A. *Total Volume as a Sum of Pyramids*

Having as an objective to create an automatic volume estimator applied to the truck containers, we are going to focus now on the method used to compute the volume, having already identified the entrance of the truck container in the image.

Applying a Delaunay triangulation to a set of points in a plane, allows us to find a combination of sets of points that form non intersected triangles.

Having the information about the distance of three points to the camera plus the information about the distance between each other, the volume of the pyramid created by these four references can be computed. The sum of the volume of these pyramids will be used for the computation of the total volume (fig. 6). In order to have the volume of the container alone, we have to subtract to the previous sum, the volume of the quadrangular pyramid in which the base is the entrance of the container.

The Delaunay triangulation is computed on the points in the depth plane and, for each set of three adjacent points given by the triangulation, we will compute the volume of the triangular pyramid that has these three points as a base and the camera has a remaining vertex.

Considering A, B, C the points of the base, we know that the base area, $A_{\triangle}$, is given by:

$$A_{\triangle} = \frac{\|\overrightarrow{AB}\|\|\overrightarrow{AC}\| \sin\theta}{2} = \frac{|\overrightarrow{AB} \times \overrightarrow{AC}|}{2} \quad . \qquad (12)$$

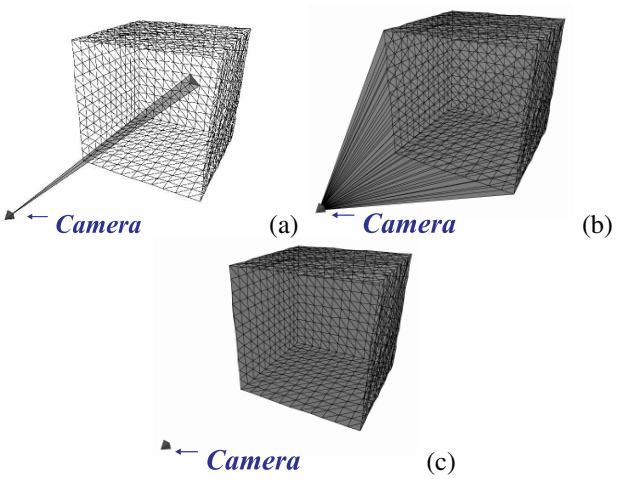


Fig. 6: (a) shows volume of a single pyramid, (b) illustrates their sum, (b) shows the final result after the subtraction of the final volume

In order to obtain the volume, we need now to compute the height of the pyramids. The normal vector to the plan ABC is given (normalized) by:

$$\vec{n} = \frac{\overrightarrow{AB} \times \overrightarrow{AC}}{\|\overrightarrow{AB} \times \overrightarrow{AC}\|} \tag{13}$$

The height h is given by the magnitude of the projection of $\overrightarrow{AC}$ over the direction of $\vec{n}$:

$$h = | < \vec{OA}, \vec{n} > | = | < A, \vec{n} > | \quad . \tag{14}$$

We can now compute the volume of each pyramid i

$$V_{p_i} = \frac{A_{\triangle_i} h_i}{3} \tag{15}$$

The volume of the square pyramid formed by the camera and the entrance vertices of the container will be subtracted to the sum all the volume of these (N) pyramids so as to compute the free volume of the container:

$$V_T = \sum_{i=1}^{N} V_{p_i} - V_o \tag{16}$$

B. Error Analysis

As we analyze the experimental results of the volume computation method described in the previous sections, we are confronted with systematic errors in the obtained volume. Some examples of experiments' results are illustrated in the figures beneath.

In these experiments the error between the computed and the measured volume was systematically higher than 0.5%, reaching in some cases values higher than $3,5\%$ and, throughout the experiments, has proven to be proportional to the distance between the identified object and the camera.

Three causes were identified for the obtained error: the dataset reliability, the precision of the depth data and the Kinect resolution.

1) Dataset Reliability: This errors are caused by the characteristics of the used object itself. In order to decrease their influence, the material of the used object should be unbendable, since this can cause unreliable corner information and not absorb IR light, which can cause information gaps in the Depth image.

2) Precision of the Volume Computation Algorithm: In order to study the Precision of the Volume Computation Algorithm, a perfect data model was computed at various distances and used in the following tests. This way, the influence of the dataset reliability is erased.

The precise 3D information of the corners of the box was included in the set of points considered in the triangulation step to explore the volume computation under perfect identification conditions. This will prevent deviations caused by the resolution, which will later be explained.

The result of analysis of computationally generated dataset, as the entrance of the box distances itself, proved that the used algorithm is precise, showing only a residual error (1.2168×10^{-011} at the distance 4m) caused by precision limits in the calculus.

3) Resolution of the Kinect Depth Data: The error obtained when computing the volume without identifying the corners was studied (fig. 5 (e)). Here, having no influence of depth precision, since computationally generated data has been used, the discretization caused by the resolution is then solely responsible for the obtained error. Fig. 7 shows two consequences of this discretization.

Figure 7 illustrates a basic scheme of the projections of depth information on the depth frame. A part of an open box is illustrated in black. The dashed lines in the figure represent pixels' bounds and in the center of each pixel a red line identifies the registered depth measurement for that pixel.

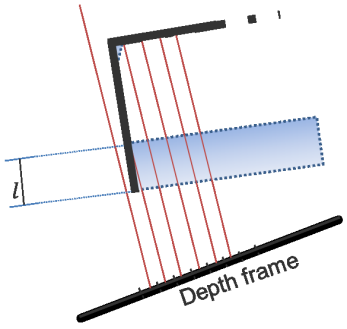


Fig. 7: Error caused by low resolution

As seen in fig. 7, the 3D information of the full length l of the box side is lost, as is the volume in the corner, incrementing therefore, the overall error.

It can be theoretically concluded that the bigger the relative pixel size, the bigger the influence of these factors in the overall computation. This has been sustained by the results of same algorithm once resolution rises (fig.5 (e)).
This effect explains the empirically observed gain of the error

obtained while increasing the distance of the object to the camera.

4) Precision of the Depth Measures: For each measure the Kinect camera possesses an uncertainty which rises with the distance. In [4] a theoretical error analysis is presented based and a theoretical random error function for the Kinects deviation. By applying this deviation to the computationally generated depth map we obtain the expected result of data retrieved by the camera. This simulated depth image will be used to study the overall influence of the Kinect depth inaccuracy over the total error. As we intend to eliminate the error components caused by an incorrect identification, once again conditions of a perfect identification will be simulated, where the, also deviated, corners are included in the triangulation. A result of the volume computation of the randomly deviated data is shown in green in Fig 5 (f) and (g). As the added noise for each distance d is a Gaussian distribution with deviation $\sigma(d)$ and expected value d, it can be considered white noise and its influence over the total error should decrease with the number of points computed. This effect can be observed in the figure 5(f) and (g).

Having then proved that the limiting factor of the kinect for volumetric measurements is the resolution and not the deviations in the depth measurements. This is only valid since no significant constant bias is present in the Kinect specifications.

V. Conclusion and Future Work

The work described in this thesis was focused in using, and assessing the applicability of, a low budget RGB-D camera, the MS Kinect, to develop two industrial applications: empty boxes verification and free volume computation. The proposed methodologies for the two applications have a significantly common basis: (1) selecting 3D object-points using pose and shape priors, (2) finding the pose of the object of interest using ICP and (3) using pose and shape to determine that a box is empty, or not, or computing the free volume within the box.

In the application 1, testing empty boxes, the camera specifications proved to suffice the requirements as the autonomous binary classifier proved to be reliable. This classifier, after using ICP in order to identify the inside of the containers, adjudge based on the hue property. The only false classifications registered were due to a strong light which could be avoided in a more controlled environment.

In application 2, free volume computation, the Kinect is used within a mockup scenario. In particular is known that the maximum range of the Kinect is significantly lesser than the length of some truck containers whose free volumes are to be estimated. Nevertheless, the proposed methodology for computing the free volume, the results obtained and the analysis of uncertainty provided already some interesting insights.

One insight is that the necessary 3D models, obtained a priori of the free volumes computation, can effectively be built from experimental data. As real objects, i.e. truck containers, may have some deviations with respect to have perfect geometric forms, makes all sense to build on line and/or adapt the necessary 3D models.

Another insight obtained is the importance of a very precise estimation of the boundary of the container. Any small deviation from the open face of the container implies a large error in the free volume computation. With the Kinect was found that the uncertainty in the estimation of the position of the open face of the container is much more relevant than the uncertainty (noise) of the depth measurements.

Further testing using more general datasets and the addition of a cost function optimization step, for increasing the reliability of the computed box localization, would yield further insights about the Kinect reliability towards volume computation. Nevertheless, the results already obtained shown that the Kinect is effectively a versatile tool and justifies the global attention from researchers and developers that it is has received in the last years, and will certainly continue to receive.

VI. Acknowledgments

This work has been partially supported by the FCT project PEst-OE / EEI / LA0009 / 2013, and by the FCT project PTDC / EEACRO / 105413 / 2008 DCCAL.

References

[1] P. Besl and N. McKay. A method for registration of 3-d shapes. *Pattern Analysis and Machine Intelligence*, 14(2), 1992.
[2] Jean-Yves Bouguet. Camera calibration toolbox for matlab. http://www.vision.caltech.edu/bouguetj.
[3] Y. Chen and G. Medioni. pages 2724–2729, 1991.
[4] K. Khoshelham and S. Elberink. Accuracy and resolution of kinect depth data for indoor mapping applications, Sensors 2012, 12, 1437-1454; doi:10.3390/s120201437.
[5] K. Low. Linear least-squares optimization for point-to-plane icp surface registration. 2004.
[6] T. Masuda, K. Sakaue, and N. Yokoya. Registration and integration of multiple range images for 3-d model construction. *Pattern Analysis and Machine Intelligence*, 1996.
[7] R. Parker. Kinect depth inpainting and filtering.
[8] K. Pulli. Multiview registration for large data sets. 1999.
[9] S. Rusinkiewicz and M. Levoy. Efficient variants of the icp algorithm. 2001.
[10] G. Turk and M. Levoy. Zippered polygon meshes from range images. 1994.
[11] M. Veloso and J. Diswas. Depth camera based indoor mobile robot localization and navigation. 2012.
[12] S. Weik. Registration of 3-d partial surface models using luminance and depth information. 1997.
[13] wolfram. point to plane distance. http://mathworld.wolfram.com/Point-PlaneDistance.html.