Computing Box Volume from Images: An Implementation

a report presented

by

Gustavo Jose Crespo

to the Graduate School

in Partial fulfillment of the

Requirements

for the Degree of

Master of Science

in

Computer Science

State University of New York

at Stony Brook

July 19 2002

State University of New York
at Stony Brook

The Graduate School

Gustavo Jose Crespo

I, Manuel Menezes de Oliveira Neto as the advisor for the above candidate for the Master of Science in Computer Science degree, hereby recommend acceptance of this report.

Abstract of the Report

Computing Box Volume from Images: An Implementation

Gustavo Jose Crespo

Master of Science

In

Computer Science

State University of New York

At Stony Brook

2002

The calculation of volume has become an intrinsic part of the day to day business of several industries in the United States and abroad. In this report, we present a method for the efficient calculation of volumes of boxes from images. The method is independent of the box textures and size as we as orientation in the image. After an image has been acquired, it is process for the location of key features like corners and edges. Once this information is acquired, several techniques from projective geometry are used to get perspective in the image and compute the actual volume of the box from the image using this information.

Contents:

1 Introduction 1

1.1 Report Organization 3

2 Previous and Related Work 4

2.1 Dimension Finding 5

3 Background 9

3.1 Box Labeling 15

3.2 A note on Line Detection 15

3.3 Conclusion 16

4 Volume Computation 17

4.1 Edges, surface normal, and plane equations 17

4.2 Distance from the plane to the Origin 20

4.3 Three Dimensional vertex location and volume 24

4.4 Conclusion 25

5 Results 26

5.1 Results on Synthetic Images 28

5.2 Errors in Synthetic Images 31

5.3 Results on Real Images 32

5.4 Analysis of results of Real Images 35

5.5 Comparison of Results 39

6 Conclusion 43

Bibliography 44

List of Figures:

3.1	Edge detection and Textures	9
3.2	Box Geometry	11
3.3	Original Picture and its Binary Image	12
3.4	Outline of the Silhouette for Line Detection	12
3.5	Reconstructed Box	13
3.6	Labeling of the Box Elements	14
4.1	Geometry of Perspective Projection 1	20
4.2	Geometry of Perspective Projection 2	22
5.1	Two Images of Checkerboards Used for Camera Calibration	26
5.2	Images Created by the simulator program	29-30
5.3	Lines Detected contains errors; a better candidate is shown	32
5.4	Synthetic Box with Short Edges	32
5.5	Pictures of Real Boxes Used for Test	33-34
5.6	Laser Construction on the Tripod	36
5.7	Missing Edges	37
5.8	Error in Detection of Edges	38
5.9	Errors in Finding the Laser Dot on the Image	38

List of Tables:

5.1 Dimensions Computed for the Synthetic Images 31

5.2 Dimensions Computed for the Real images 35

List of Graphs:

5.1 Error Graphs 41

Acknowledgment:

I would like to express my most sincere gratitude to my advisor Professor Manuel Menezes de Oliveira Neto for his invaluable help and patients during the development of this project. I believe that his work ethic and his commitment made this exercise a full learning experience for me. I was able to learn not only about subject matter, but also about myself. I believe that after going thought this experience I have become a better computer scientist and a better person.

I would also like to express my deepest gratitude to my best friend Oi Ling Fong for her support and encouragement throughout my academic career. I think I have been able to try new things and to excel because of your motivation, help, and love. I do not think I would have been able to do all of this without you.

Chapter 1 Introduction

The need to determine dimensions of boxes on the fly has increasingly become a very important factor in the day-to-day business activities of certain industries in the United States and around the world. For example, despite the fact that the courier industry in the United States is a multi-billion dollar industry, there is no cheap and portable way for a courier representative to assigned prices based on box volume. Unless, of course, he/she measures these boxes by hand, which is not a practical approach. Most couriers provide their own boxes; and in the cases of custom boxes, tariffs are assigned on a prorated basis. It is not uncommon to go to a courier's web site to obtain information on cost and read the terms "box size" and "weight", and warnings as to the limitations and value of the prices given. Airline companies and storage facilities are also examples of industries that would greatly benefit by having a device that would allow them to determine such dimensions in cheap and efficient way. It would be cost effective and a logistical advantage for these industries to have a method for the calculation of volume in a flexible way and on real time. This information could then be use to determine optimal ways of storing the boxes while in transit, to determine the cost to the costumer and to the carrier itself, and for quality and logistical control. A storage facility could use this information to find optimal ways in which to store such items as to maximize space and safety. In fact, there are machines designed for the sole purpose of determining box dimensions. Nevertheless, they are too costly and their use is limited by the complexity of the technology used. There is a real need for the development of a cheap way to determine box dimensions and volume in a flexible, cheap, and reliable way. This report describes the theory and implementation of a system for computing box volumes from single images based on the method presented in [Oliveira 01].

With a modest investment on some basic instruments like camera and laser pointers, a new range finder is created to project a known distance onto pixel space. This information is later used to remove ambiguities inherent in perspective projection, and to calculate the relative positions of the corners of the box in 3-dimensional space. Once this information is known, the calculation of the volume follows easily from basic mathematics. Because of the limited use of equipment (camera and laser pointers) this technique can be used on a variety of settings with great flexibility. For example, a carrier representative could have such a device on his truck that would enable him to calculate volumes of boxes in the field. The size of such a device does not need to be of concern. With the implementation of the algorithm on a computer chip, a small scanner could be created. The only user interaction that is required is the projection of laser dots on the box and the pressing of a trigger. Another aspect that has been taken into consideration is the performance of this technique. It takes our software prototype less than 2 seconds to determine the volume of the box on a Athlon 1.3 MHz, with 448 MB of main memory. Image analysis, and vertex detection is the most time consuming part of the method, and can be accelerated with hardware implementation. However, once we have calculated all the vertices of the box the rest of the algorithm works very fast.

1.1 Report Organization

This report has been organized in the following way: Chapter 2 overviews relevant previous and related work and discusses why this new method represent an improvement with respect to the previous techniques. A brief description of the objectives and achievement of these methods is also introduced as well. In order to make this report self-contained, Chapter 3 introduces a brief review of some of the relevant concepts involved in the algorithm. These include: vanishing points, vanishing lines, basic calculus of plane equations, and a brief introduction to the computer vision techniques employed. Ideas like the camera's white balance, and why this is important are also introduced so that the reader understand the setting in which these experiments have been carried out. Chapter 4 goes on to explain the technique and algorithm used in the implementation discussed in this report. The main ideas explain are: vertex and box reconstruction from silhouettes, determination of 3D coordinates for vertices, and finally volume calculations. Chapter 5 presents the results of the program implemented based on the theory covered in Chapter 4. It is important to have an understanding on the accuracy of the technique and all of its implications in the calculation of the actual measurements. Hence, a brief description of the error, as well as their possible causes is presented. We cover ideas like approximation to missing vertex, approximation to missing edges, and the statistics used to determine such errors. Chapter 6 concludes this report with a summary of the main points covered.

Chapter 2 Previous and Related work

This chapter discusses some of the previous work relevant to the theory implemented in this Masters Project. It presents a brief account of the two main methods associated with image analysis and model parameterization for distance calculation. Some of the benefits and drawbacks of such techniques, as well as their computational costs and practical uses are presented. Several procedures use different approaches to analyze and extract object information from images. These procedures are summarized and studied for comparison. These ideas are important since the computation of distance and volumes usually depends on a priori knowledge about the shape being considered. Even though it is possible to obtain shapes and recognize certain objects in an image, this information is not enough to unambiguously determine "exact" dimensions in the real world. This follows from the fact that different objects may have the same projection on the image plane.

After some of the theory behind object recognition has been presented, we discuss two techniques for computing scaled dimensions of objects directly from images. While one of these two techniques is very general and applies to several settings and scenes from the real world, the other is very specific and very much related to the main topic of the work presented here. For some simple geometry, it is possible to compute the dimensions of an object up to a scaling factor [Taylor99]; however, one cannot determine precisely the dimensions without some knowledge about how distances in the real world relate to distances in the image.

Shape recovery and object recognition have been studied quite extensively in computer vision and artificial intelligence. Image processing techniques like edge detection [Canny, 86], and the Hough transform [Hough, 59] are used to extract basic geometric information from the images. Most of these approaches, like Kanade's method [Kanade 81], use the edges detected in the image to build geometric representation of the objects contained in the image. This approach gives an intuitive understanding of the shapes in the image; however, it is limited in the sense that no information regarding distances can be obtained. It is true that this approach would lead to the identification of arbitrary objects; however, this information is not enough to determine dimensions of any kind in the image. Thus for the application desired here, these methods are not practical. Hence, we review the two major methods relating to this work in the next section.

2.1 Dimension finding

The two techniques presented in this section exhibit a level of specialization regarding the ability to obtain dimensions of objects from images. While Taylor's method [Taylor 99] gives only up to scale approximations of the dimensions for object with simple geometry, Kefei Lu's project [Lu 01] gives up to scale values of the dimensions of a wide range of boxes.

The first method due to C. J. Taylor and D. Jelinek use projective geometry to obtain object dimensions, up to a scaling factor, as well as shape from the single still images [Taylor 99]. It is not clear from their presentation what is the performance of this method on images, or what computer system they used for their implementation. No error analysis was provided either, and therefore no conclusion can be made as to the accuracy or applications of this method. It is also not clear how this technique could be extended to allow for the computation of the actual dimensions of the object in the image. Nevertheless, this technique is very powerful and well suited for architectural model visualization and reconstruction. Since a scale version is available, this information can be used to create relational information in the image. This relational information could be use to rearrange objects in the image while preserving the dimension constrains among the objects. For example in the model of a house, the bathroom and dining room could be rearranged while preserving distances between them and other components of a house. This information can be a useful and cost effective way of giving customers a feel for the final look of a house being designed.

The second method is due to Kefei Lu [Lu 01] who showed an iterative algorithm to obtain up to scale dimensions of a box. The basic algorithm starts extracting lines from the image corresponding to the edges of the box. This step uses standard techniques from image processing like noise removal, brightening and smoothing the image, and finally detecting edges corresponding to the boundaries of the box. Lines can then be found using the corresponding edges. It must be noted here that the number of smoothing and brightening passes used in this technique are determined manually. This becomes a major drawback because these smoothing and brightening factors may change from image to image. If a user were to use this method, he/she would have to be trained on how to determine these factors. A better method would determine these factors on its own so that no user interaction is needed. After all lines are located, they are organized and the box vertices are located. Once this step is completed, a box outline is created and the up to scale dimensions of the original box are calculated. This algorithm was designed to produce an answer within 10 second. If on some input the algorithm takes longer than the preset time limit, the procedure is interrupted and a possibly incorrect answer is provided or an error message is given as output. One more reason for failure is a poor image input.

Lu’s method shows a good solution for a well-defined practical problem in computer vision. However, some of the limitations inherent in the technique limit it from being used in applications like that of a scanner for box volume. For instance, the distance from the camera to the object needs to be known a priori for the method to yield the actual dimensions of the box. If a scanner were to be built implementing this method, the requirement of knowing the distance of the camera to the box becomes impractical. While finding the vertices of the box several candidates are considered. The process of eliminating false vertices can be very costly. If this cost were to be too high on a good input image, the program will produce a failure where it should have worked. Once the vertices are found (several possible candidates), the outline testing takes place. Here, again there are a big combinational number of outlines to test and this also has a high computational cost. For this reason, not all outlines are tested and a weighted criteria is used to eliminate some of them. No error analysis is provided on the results for comparison or estimation.

So far, two main approaches to the problem of volume calculation have been presented. However, neither of this technique solves all the questions relating shape and dimension of boxes on an image. At this point, there is no single method that with the push of a button or the input of an image will produce the desired results without any further user interaction. While the task of finding the dimension of a box may seem a rather limited result, the applications are very broad.

Chapter 3 Background

One of the most basic steps toward the identification of a box in a picture is to locate its edges and corners. One of the major limitations of the use of conventional edge-detection algorithms is their sensitivity to textures. Thus, for instance, labels on a box, like that of FedEx, may create false edges corresponding to the outline of the label on the box (see figure 3.1.) If this traditional approach was to be pursued to obtain the edges of the box, a considerable amount of computational time would have been spend sorting out and eliminating all false edges. This follows from the fact that some of the artificial edges would be located in places where disambiguating them from the real ones would required a combinatorial process. Another major consideration was the fact that different lighting situations give rise to bogus edges as well. This follows from the fact that edge detectors are designed to identify high frequencies in the image.

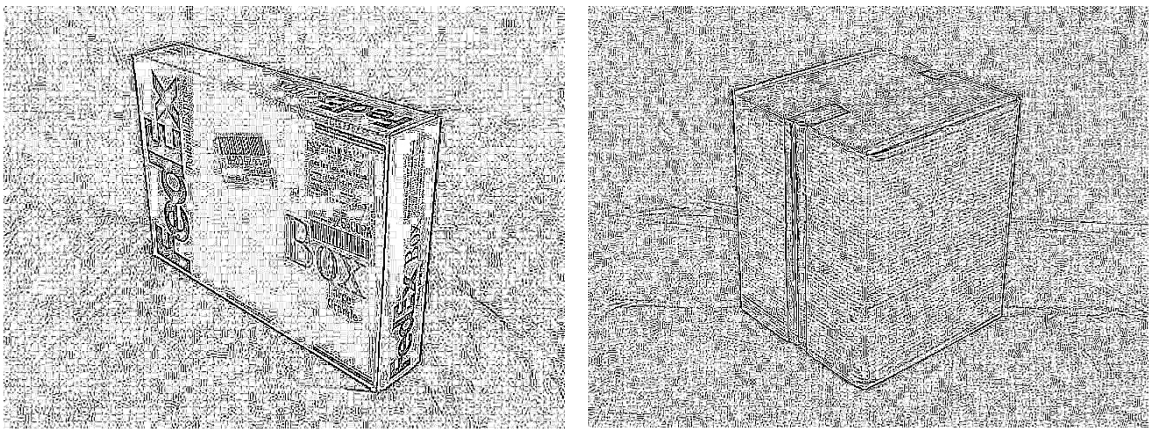


Figure 3.1. Edge detection and textures

Workarounds can sometimes be found by providing different threshold to the edge detector program. However, this would imply that the user would have to change this threshold depending on the background, texture and lighting condition given by different boxes. As it can be seen, this scheme is not very practical at all. A more general method that would be free of calibration, easy to use, and more flexible was therefore sought.

In order to find this new method, we decided that the geometry of the box could help in the design of an input free technique for the edge detection in this particular setting. Observing the shape of a box, it can be seen that a lot of the geometry is known beforehand (see figure 3.2). A major step towards using the known geometry of the box for its reconstruction is to find its silhouette of the box. With the silhouette, at least 6 of the corners of the box are obtained. We will obtain exactly 6 if only two of the faces of the box are visible, and 7 if three faces are visible. It must be noted that by recovering the silhouette of a box, information about the vertex located in the interior of the box's outline and about the edges incident to it are temporarily lost. However, using the relationship of the lost vertex with three other vertices of the box the location of the missing vertex can be recovered. To obtain this silhouette, a background was introduced to eliminate unwanted noise and false vertices provided in the backdrop of the image (see figure 3.3)

By providing such a background, color can then be used to extract the outline of the box very efficiently. This is done by looking at the ratios between the Red, Green, and Blue channels of the colors at every pixel in the image. It is known, before hand, what is the appropriate ratio for the background; hence, by setting every pixel satisfying this ratio to black and the rest of the pixels to white, a binary image is created (see figure 3.3). A particular shade of color green used as background was selected because it is not a common color present in boxes. However, this color could be changed to accommodate getting the silhouette of boxes with green color or texturing on them. Since color is used, lighting becomes a very important factor in this process.

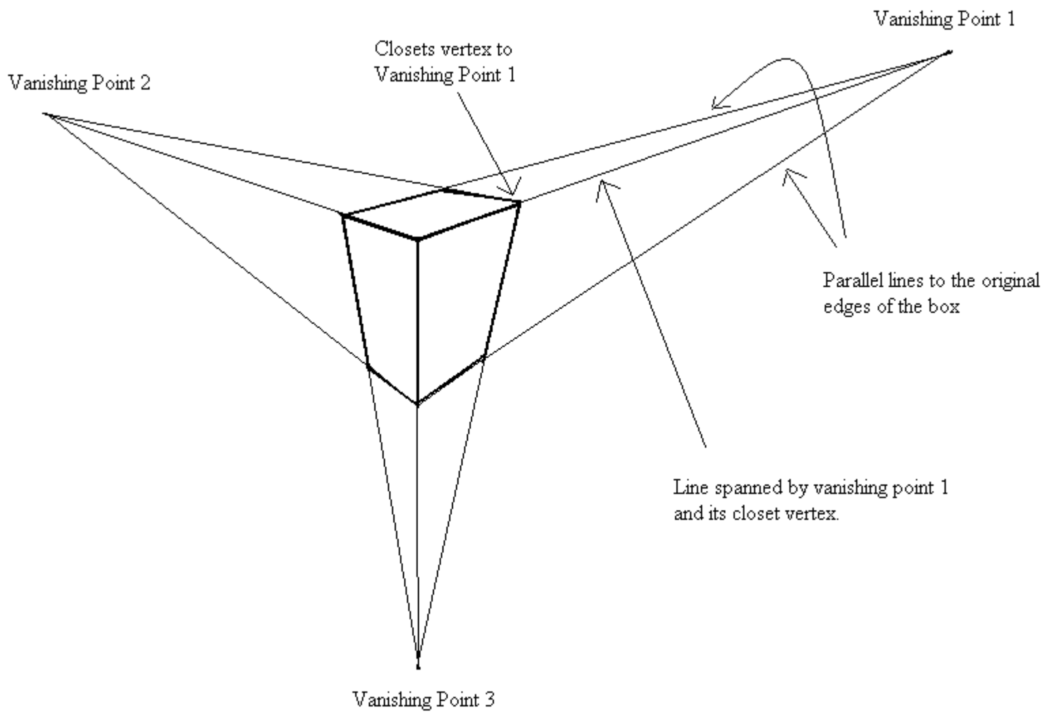

Vanishing Point 1
Closets vertex to Vanishing Point 1
Vanishing Point 2
Parallel lines to the original edges of the box
Line spanned by vanishing point 1 and its closet vertex.
Vanishing Point 3

Figure 3.2. Box geometry

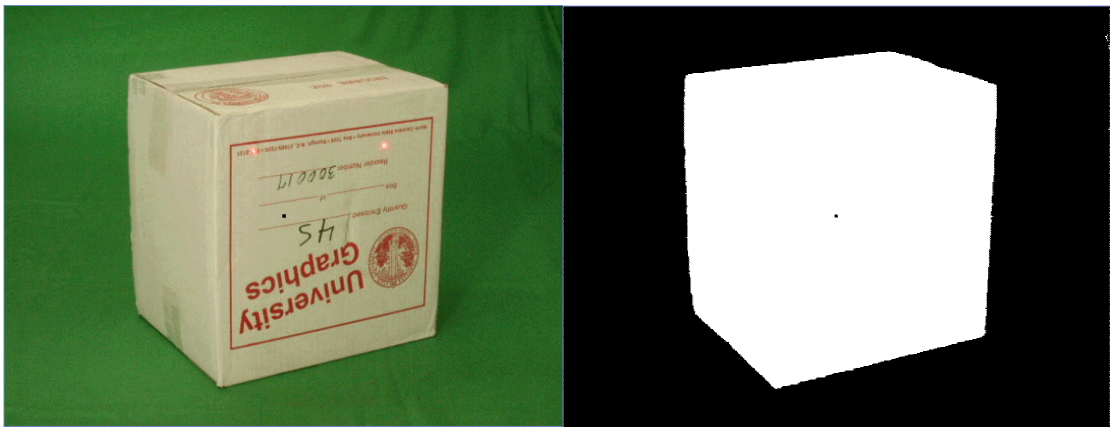


Figure 3.3. Original picture (left), and its binary image (right)

Once a binary image (see figure 3.3) is created, the box can be rebuilt from the silhouette by using a simple geometric procedure. The first step in this procedure is to recover the lines corresponding to the edges of the box. . Since the image consists of only two colors corresponding to the silhouette and its background, a high pass filter can be use to generate a contour of the outline of the box (see figure 3.4.) This contour image consists of only the edges contained in the outline of the box. From these the lines, all visible edges of the box can be recovered.

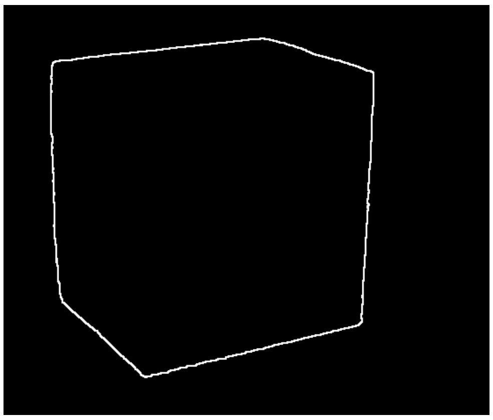


Figure 3.4. Outline of the silhouette for line detection

Once all lines have been recovered, we look for vanishing points. A vanishing point can be described as the point in which two parallel lines seem to converge in a perspective drawing (see figure 3.2.) . Another way to think of vanishing points is to realize that they are the projection of points at infinity. Vanishing points are use to get cues, like position and/or orientation, from an image. At this point, all the edges and the vanishing points of the image are known, and from this information the missing corner and edges can be rebuilt. The missing edges can be recovered by observing that the lines corresponding to parallel edges of the original box intersect at the vanishing point (see figure 3.2.) Therefore, if we take the intersection of two of the lines corresponding

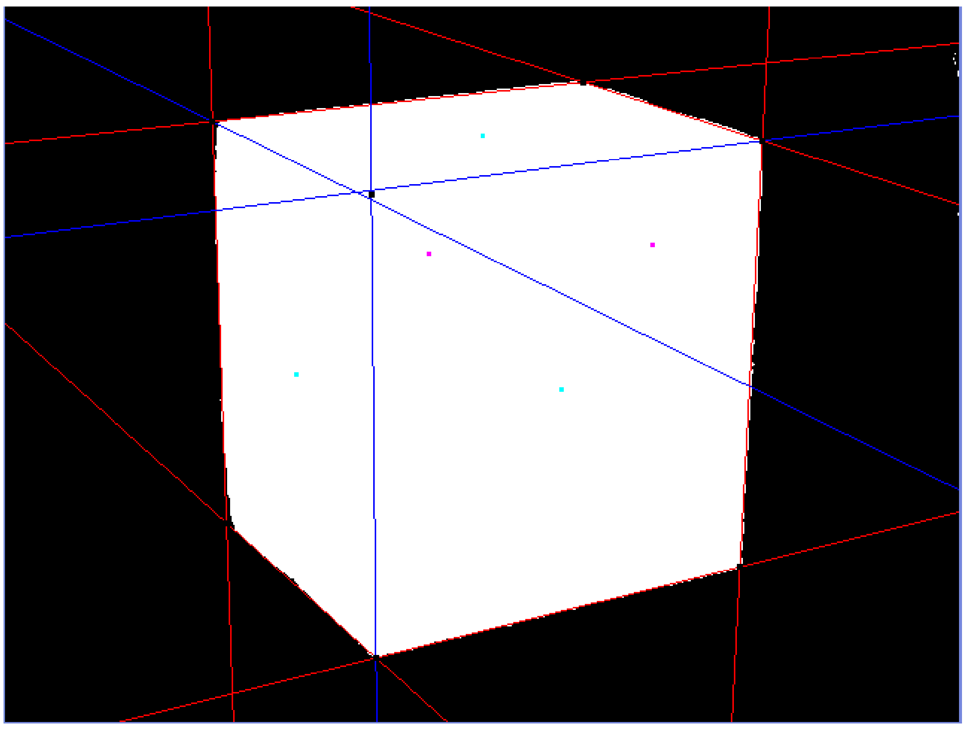


Figure 3.5. Reconstructed box

to parallel edges of the box in 3-dimensinal space, and draw a line from this intersection to the nearest corner of the box, the missing edge is obtain (see figure 3.2). This can be done for the three different vanishing points (see figure 3.2) providing the three missing edges. The intersection of these three edges corresponds to the missing corner of the box, visible in the original picture but lost by the creation of the binary image. It must be observed that three intersections will occur and that they may not be exactly the same. A least squared method was used to find the best fit for the point. Figure 3.5 provides an example of the result of the reconstruction of the box.

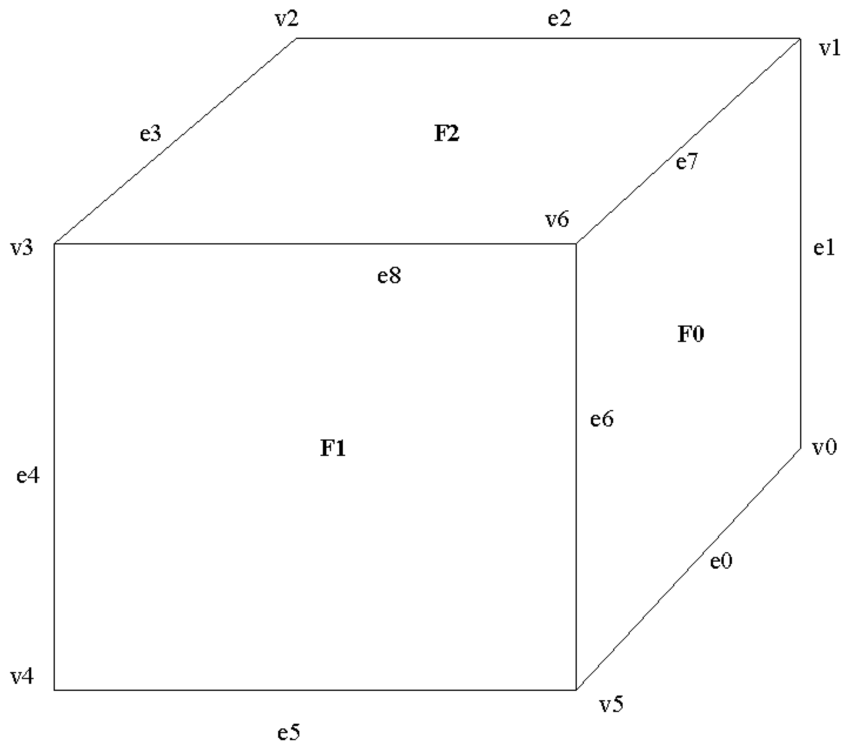


Figure 3.6. Labeling of the box elements

3.1 Box labeling

At this stage, the box has been completely reconstructed and the all edges have been located. It would be convenient to have a sense of order for the set of vertices and edges on the outline of the silhouette. Therefore, a labeling of the faces, edges, and vertices of the box is introduced at this point. Vertices are labeled counterclockwise from 0 to 6 (see figure 3.6). The first 7 edges are labeled similarly; their numbering follows the value of the right end point or vertex. To avoid any ambiguities the last remaining edges are numbered 8 and 9 as in figure 3.6. The faces, F0 to F2, are labeled clockwise.

3.2 A note on line detection

A Hough transform is used to recover the lines from the contour of the outline of the box. However, the protocol used to apply the Hough transform was also of importance. Because of the nature of the line detection, several lines would be detected that would correspond to only one edge of the figure. This follows from the fact that some points corresponding to one edge vote for several possible lines. The votes on these lines can sometime outnumber the votes of other possible edges. Hence, the result would be that two lines are obtained for the same edge. To eliminate this sort of mistake in the line/edge detection process, all those points that vote for one line are eliminated and the edge detection proceeds in what is left of the outline of the box. Once all the lines are detected, we can then find the corners of the box as the intersection of two adjacent lines. Once the corners of the box have all been identified, the missing corner and edges are located.

3.3 Conclusion

Thus far, several concepts and ideas used in the implementation of this technique have been discussed. These ideas will be fundamental in the understanding of the theory of box volume detection, and the implementation code. The ideas discussed here introduce a corner detection pipeline for the box in which the edges are detected first, then the corners corresponding to line intersection, and finally that box reconstruction. Once the box has been reconstructed, we can proceed to the next chapter for the discussion of the main theory behind the implementation.

Chapter 4 Volume computation

In this chapter, the mathematics and underlying method for volume finding will be presented. To remove perspective ambiguities, two parallel laser pointers are directed to one of the faces of the box. This information will then be used to provide a constrain that will allow the computation of the actual box dimension from an image. In this report, we will adhere to the standard notation used in computer vision for points in the image and points in 3-dimensional space. That is, 3-dimensional coordinates will be represented using capital letters, while image coordinates will be represented in small letters.

4.1 Edges, surface normal, and plane equations

As discussed in Chapter 3, the result of the Hough transform on the contour of the box (see Figure 3.4) is the set of lines corresponding to the silhouette a box. With these lines, we reconstruct the missing edges and missing corner using the vanishing points in the X, Y, and Z directions. The next task is to identify the corners of the box. This can be done by observing that they correspond to the intersection of the lines given by the Hough transform. Thus to find the corners, we find the intersection of consecutive lines in either clock or counterclockwise order. It is easy to think of these lines as the edges on the box (see figure 3.6). Hence, while e0 and e1 are consecutive counterclockwise, e0 and e2 are not since there is an edge in between.

The lines given by the Hough transform are converted from polar coordinates $\rho = i\cos(\theta) + j\sin(\theta)$ to Euclidean coordinates $y = mx + b$, where $m = -\frac{\cos(\theta)}{\sin(\theta)}$, and

$b = \frac{\rho}{\sin(\theta)}$. Assume that $y_1 = m_1 x + b_1$, and $\mathrm{y}_2 = m_2 x + b_2$ are two such lines. For example, y_1 can correspond to e0 in the box, and y_2 correspond to e1 (see figure 3.6). Then the intersection point or corner will have coordinates $x = \frac{b_2 - b_1}{m_1 - m_2}$, and $y = m_1\left(\frac{b_2 - b_1}{m_1 - m_2}\right) + b_1$ (I). Now, the vanishing points associated with the X, Y, and Z directions in 3D can be obtained by computing the intersection of pairs of parallel lines along these directions in 3D. For example, e5 and e2 provide a vanishing point in the X direction, e1 and e4 yield the Y direction, and e0 and e3 give the vanishing point in the Z direction. These vanishing points can be calculated by finding the intersection of the lines corresponding to these edges and formula (I). After we have found the vanishing points we can proceed in the same manner finding the coordinates of the 6 vertices or corners corresponding to the box in the silhouette. After the lines corresponding to the edges of the contour of the image have been sorted, it is straightforward to find the intersection of all lines that are consecutive to each other. These intersections will correspond to the 6 corners of the box visible in the silhouette. At this point we have 6 of the corners of the box, the 7th is obtained using the vanishing points as described in Chapter 3.

Let vp_1, vp_2, and vp_3 correspond to the vanishing points found using the previous argument in the X, Y, and Z directions. According to the duality principle of projective spaces [Bouguet 99] the cross product of two points in homogeneous coordinates is the line joining them. Hence to obtain the normal vector to the faces of the box we use the standard cross product to obtain the following results:

$$N_{F0} = vl_1 = vp_1 \times vp_2$$
$$N_{F1} = vl_2 = vp_2 \times vp_3$$
$$N_{F2} = vl_3 = vp_2 \times vp_1$$

where the components of the normal vectors can be written as:

$N_{Fi} = (A_{Fi}, B_{Fi}, C_{Fi})$ for $i =$ 0, 1, 2.

Hence, the equations of the three planes corresponding to the faces of the box have expressions of the form:

$A_{Fi}X + B_{Fi}Y + C_{Fi}Z + D_{Fi} = 0$ for $i =$ 0, 1, 2   (II)

Since the normal N_{Fi} are computed with respect to the camera coordinate system, the relative position and orientation of the faces of the box with respect to the camera can be obtained by solving (II) for D_{Fi}. This can be done with the introduction of some constrain relating distance in 3D to some distance in the image. Since no distances in 3D are known a priori, such a constraint can be obtained by projecting a known pattern on the scene. Two dots projected on one of the faces of the box by a pair of parallel lasers beams can be used to satisfy this constraint [Oliveira 01].

Let δ be the distance, in the real world, between the two parallel laser pointers. Then, the distance in 3D between the two projected dots on a face of the box (F0 to F2, see figure 3.6) will be given by $\Delta = \delta / \cos(\theta)$ where θ is the angle the normalized projection of the normal of the face onto the XZ plane makes with the Z axis of the camera [Oliveira 01]. Such a projection is obtained by simply dropping the y component of the face normal vector, followed by a normalization procedure. Thus, $\cos(\theta) = -CVD \bullet NP$ where CVD is the camera view direction, i.e. $(0,0,1)$.

4.2 Distance from the planes to the Origin

Throughout this discussion, it is assume that a pinhole camera model is used. Points in 3D space are projected onto a plane known as the image plane. The center of projection (COP) is considered to be at the origin and the distance from the plane to the COP (center of projection) is known as the focal length f (see Figure 4.1.)

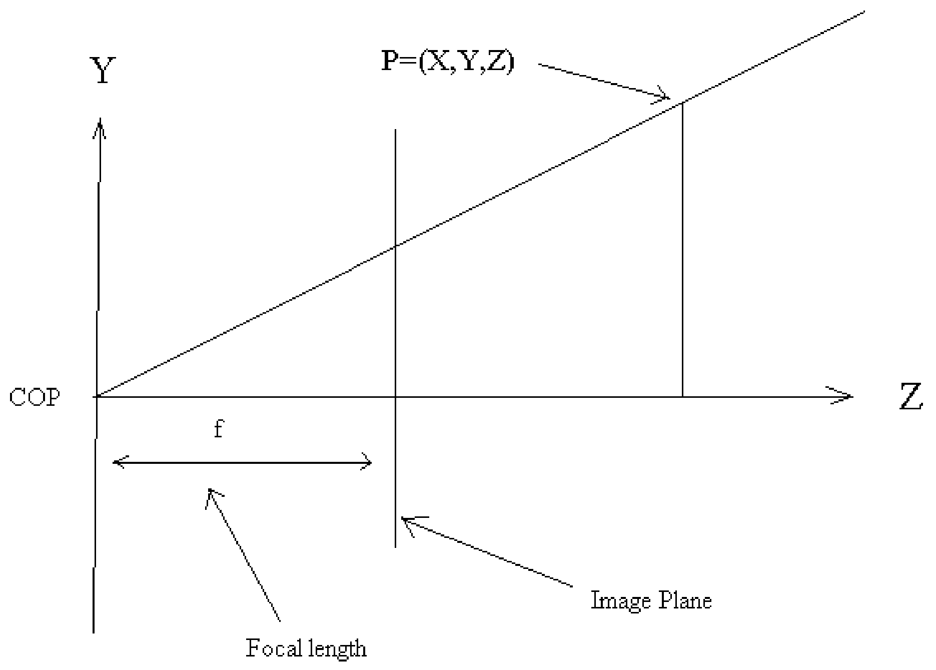


Figure 4.1 Geometry of perspective projection 1

Under the pinhole camera model a point P with coordinates $(x, y, z)^T$ is projected onto a point in the image plane with homogeneous coordinates $X^{'} = (f\frac{x}{z}, f\frac{y}{z}, 1)$. This central projection can be expressed as a linear map

$$\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} fx \\ fy \\ z \end{pmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}.$$

At this point, the assumption is made that $f = f_x = f_y$, which is valid since most modern cameras used square CCD pixels. . To transform the points to a focal length of unit size all that needs to be done is to normalize the points using the matrix

$$K = \begin{bmatrix} 1/f & 0 & 0 \\ 0 & 1/f & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

which performs the division by the focal length.

Let $p_i = [x_i, y_i, 1]^T$ be the pixel coordinates of a point in the image. Further assume that the coordinates of the principal points have been subtracted from these pixel coordinates, and that they have been multiplied by the inverse of the focal length in pixels. Now let $\overline{p_i} = (x_{pi}, y_{pi}, 1)$ be the coordinates of this points after subtraction of the optical center only. Then, p_i can be expressed using linear operation involving focal length. Namely,

$$p_i = K\overline{p_i}$$

K is also known as the intrinsic camera matrix containing the intrinsic parameters. Now, using the known geometry (see figure 4.1 and figure 4.2) of how the points project to the image plane, the point coordinates can be expressed as

$$X_i = x_i Z \qquad \text{(III)}$$
$$Y_i = y_i Z$$

This follows since the focal length has been normalize to be equal to one. Now, given two arbitrary points $p_1 = (x_{p1}, y_{p1}, z_{p1})$ and $p_2 = (x_{p2}, y_{p2}, z_{p2})$ on a face of the box for which the distance is known (these points correspond to the laser dots projected on the face of a box), the following relationship is satisfied:

$$A_{Fi}x_{p1} + B_{Fi}y_{p1} + C_{Fi}z_{p1} + D_{Fi} = 0$$
$$A_{Fi}x_{p2} + B_{Fi}y_{p2} + C_{Fi}z_{p2} + D_{Fi} = 0$$

From which we can obtain

$$A_{Fi}x_{p1} + B_{Fi}y_{p1} + C_{Fi}z_{p1} = A_{Fi}x_{p2} + B_{Fi}y_{p2} + C_{Fi}z_{p2} \text{ (IV)}$$

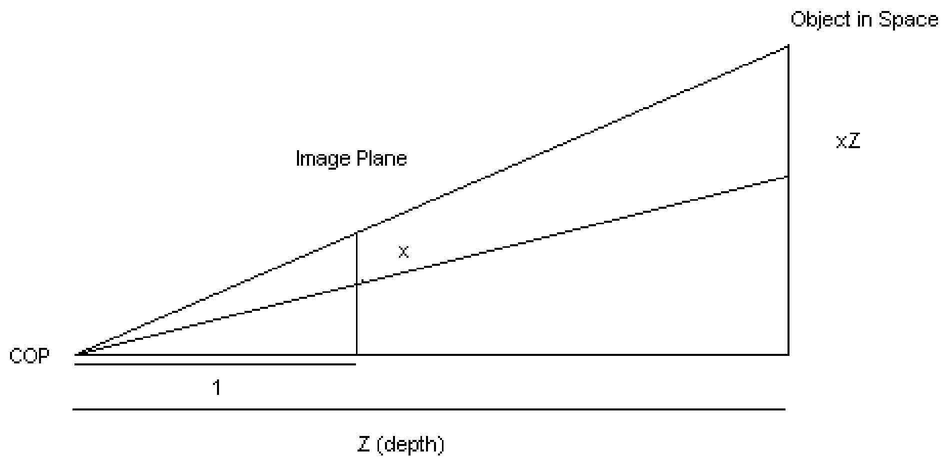


Figure 4.2 Geometry of perspective projection 2

By using the relation II and applying it to points in III with

$$\begin{aligned} x_{p1} &= X_{p1} z_{p1}, \\ y_{p1} &= Y_{p1} z_{p1}, \\ x_{p2} &= X_{p2} z_{p2}, \text{ and} \\ y_{p2} &= Y_{p2} z_{p2} \end{aligned}$$

the following relationship is then derived:

$$\begin{aligned} A_{Fi}x_{p1} + B_{Fi}y_{p1} + C_{Fi}z_{p1} = A_{Fi}x_{p2} + B_{Fi}y_{p2} + C_{Fi}z_{p2} &\Leftrightarrow z_{p1}(AX_{p1} + BY_{p1} + 1) = z_{p2}(AX_{p2} + BY_{p2} + 1) \\ &\Leftrightarrow z_{p1} = \frac{(AX_{p2} + BY_{p2} + 1)}{(AX_{p1} + BY_{p1} + 1)} z_{p2} \\ &\Leftrightarrow z_{p1} = \Gamma z_{p2}, \text{ where} \qquad \text{(V)} \end{aligned}$$

$$\Gamma = \frac{(AX_{p2} + BY_{p2} + 1)}{(AX_{p1} + BY_{p1} + 1)}$$

Now, since the distance $d = \|p_1 - p_2\| \Leftrightarrow d^2 = (p_1 - p_2)^2$ between the two points is known, the expressions:

$$\begin{aligned} D^2 = (P_1 - P_2)^2 &\Leftrightarrow D^2 = (X_{p1} - X_{p2})^2 + (Y_{p1} - Y_{p2})^2 + (Z_{p1} - Z_{p2})^2 \\ &\Leftrightarrow D^2 = (x_{p1}z_{p1} - x_{p2}z_{p2})^2 + (y_{p1}z_{p1} - y_{p2}z_{p2})^2 + (z_{p1} - z_{p2})^2 \\ &\Leftrightarrow D^2 = z_{p1}^2(x_{p1}^2 + x_{p1}^2 + 1) + 2z_{p1}z_{p2}(x_{p1}x_{p2} + y_{p1}y_{p2} + 1) + z_{p2}^2(x_{p2}^2 + y_{p2}^2 + 1) \\ &\Leftrightarrow D^2 = az_{p1}^2 + 2bz_{p1}z_{p2} + cz_{p2}^2, \text{ where} \qquad \text{(VI)} \end{aligned}$$

$$\begin{aligned} a &= (X_{p1}^2 + Y_{p1}^2 + 1) \\ b &= (X_{p1}X_{p2} + Y_{p1}Y_{p2} + 1) \\ c &= (X_{p2}^2 + Y_{p2}^2 + 1) \text{ are obtained.} \end{aligned}$$

Putting III and IV together, the result

$$z_2 = \pm\sqrt{\frac{d^2}{a\Gamma^2 + 2b\Gamma + c}} \qquad \text{(VII) is derived.}$$

Since the box is in front of the camera, z_2 should be positive; therefore only the positive value is considered.

At this point, all the variables of the plane equations (II) are known except that the D coefficients of the three planes need to be determined. To obtain these coefficients, one of the three planes is selected and the values for the coordinate of the vertex shared by the three planes *(v6)* are substitute in the equations, which yields:

$$D_{Fi} = -(A_{Fi} X_{pi} z_2 + B_{Fi} Y_{pi} z_2 + C_{Fi} z_2)$$

At this point, the 3-dimensional coordinates of the share vertex are known. Hence, using the other two plane equations we can determine the other D values for the two planes by substitution of the share vertex.

4.3 Three dimensional vertex location and volume

In the previous section, the three plane equations (II) corresponding to the faces of a box were uniquely determined. With this information all that it is required is to obtain the 3D coordinates of the box, and from them the length, height, and width of the box. The three-dimensional coordinates of the vertices v0, v1, v5, and v6 can be computed using the place equation corresponding to F0. v2 will be computed using F2, and v4 will be computed using F1. Since computations are the same for all the vertices only one of the vertices computation will be presented here. All that needs to be done to compute the rest is adjust for the right face in the computation.

Take $V0 = (VX, VY, VZ)$, and let $V0_{space} = (VX_{space}, VY_{space}, VZ_{space})$ represent the three-dimensional coordinated of v0 (to be determine), with the normal and plane equation given by F0. Then, $\overline{v0} = Kv0 = (vx_{norm}, vy_{norm}, vz_{norm})$

$$\begin{aligned} vz_{space} &= -\frac{D_{F0}}{A_{F0}vx_{norm} + B_{F0}vy_{norm} + C_{F0}vz_{norm}} \\ vx_{space} &= vx_{norm}vz_{space} \\ vy_{space} &= vy_{norm}vz_{space} \end{aligned}$$

Assuming that these computations have been carried out for all vertices using the, then width, height, and length can be calculated as:

$$\begin{aligned} width &= \left\| v5_{space} - v4_{space} \right\|, \\ height &= \left\| v4_{space} - v3_{space} \right\|, \\ length &= \left\| v5_{space} - v0_{space} \right\| \end{aligned}$$

As you can see, we only need to calculate the 3D coordinates for 4 vertices; however, having all of them allows us to compare the width, height and length using different vertices. This is a good tool for error analysis.

4.4 Conclusion

The method described here makes no assumptions as to the orientation of the box, with the exception that 3 of the faces of the box have to be visible. Since the volume computation only requires 4 vertices, it can be performed even when only two faces of the box are visible [Oliveira 01]. In this case, a slightly different approach should be used when extracting the edges of the box from the image. Since only 6 of the vertices defining the silhouette of the box are visible.

Chapter 5 Results

This Chapter discusses the results obtained using an implementation of the technique presented in Chapter 4. A second program for creating simulated views of boxes with arbitrary dimensions and orientations was also implemented. This simulation program played an important role towards the final development and validation of the method.

The experiments were carried out using both synthetic and real images. In all cases, the resolution of the images were 640×480 pixels. The simulator was used to create synthetic images was written in C++ and OpenGL. The resulting images were then saved using the JPEG format. The real images were taken using an Olympus C-3000 Zoom Camera with the following parameters: ISO400 for the film, Shutter speed $1/8$, the F-Stop was f11, and manual focus set at infinity. The camera parameters (focal length and optical center) were obtained using Bouguet's calibration toolkit for Matlab [Bouguet year]. Figure 5.1 shows two of the images used for camera calibration.

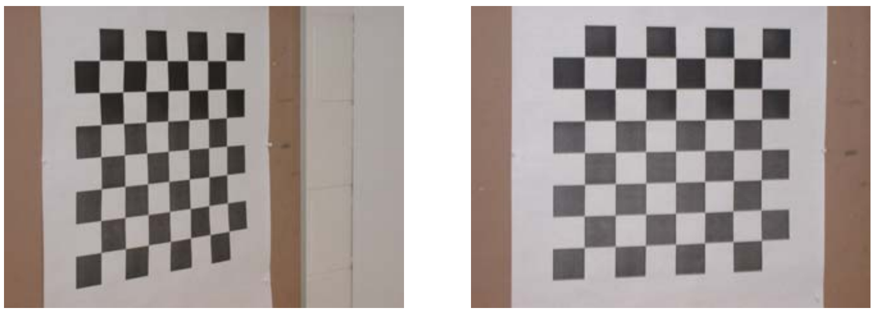


Figure 5.1. Two images of a checkerboard used for camera calibration

No flash was used and the light setting was set to fluorescent light (balance setting in the camera menu). The Olympus proprietary software (Camedia) was used to remove radial distortion from the images.

To ensure that these results were independent from the distance from the camera to the box, this distance was also changed in both cases. Since it was possible to change the distance between the laser pointers in the simulator, this was done to ensure that the method works for different distance between the laser pointers. In the case of the real images, it was not possible to change this distance because the laser construction was constructed for a specific distance. That is, there was no means of calibrating the laser construction for different distances. One more aspect to consider is the location of the laser dots. The first step towards location of the laser dot is to identify the pixels with almost saturated red component. Then, the centroid of this cluster of pixels is considered as a laser dot in the image.

The jpeg format was used to save both the real and synthetic the images. Because of compression, some errors were introduced. This effect is most apparent while detecting edges or finding the laser dots in the images. In some cases, the laser dots were not detected, while in others the edges got muddled and a little smudge on the image. In most cases, however, the quality of the images was sufficient for the technique to be validated.

The errors were calculated using the following formulas:

$$\textit{Volume}\ \mathrm{E}\textit{rror} = \frac{\left|\text{Computed Volume } - \text{Actual Volume}\right|}{\text{Actual Volume}} \times 100, \text{ and}$$

$$\textit{Dimention error} = \frac{|\text{Computed Dimention} - \text{Actual Dimention}|}{\text{Actual Dimention}} \times 100$$

Where computed volume is the volume computed by the program, and actual volume is the one calculated using the measures obtained with a measuring tape, or computed in the synthetic images by setting the vertices to the desired distance in the simulator. Similarly for the dimension error, the computed dimensions are the ones computed by the program.

5.1 Results on synthetic images

Some of the images created by the simulator are presented in figure 5.2. Several combinations of dimensions and distance between the simulated laser dots were considered. The relative position of the box in the image was also changed. All images were created to model the dimensions and possible positions of a real box. The first image in the mosaic of synthetic images show the simulated laser dots on the simulated box. Table 5.1 presents the measurements of these synthetic boxes along with the corresponding dimensions computed by the program.

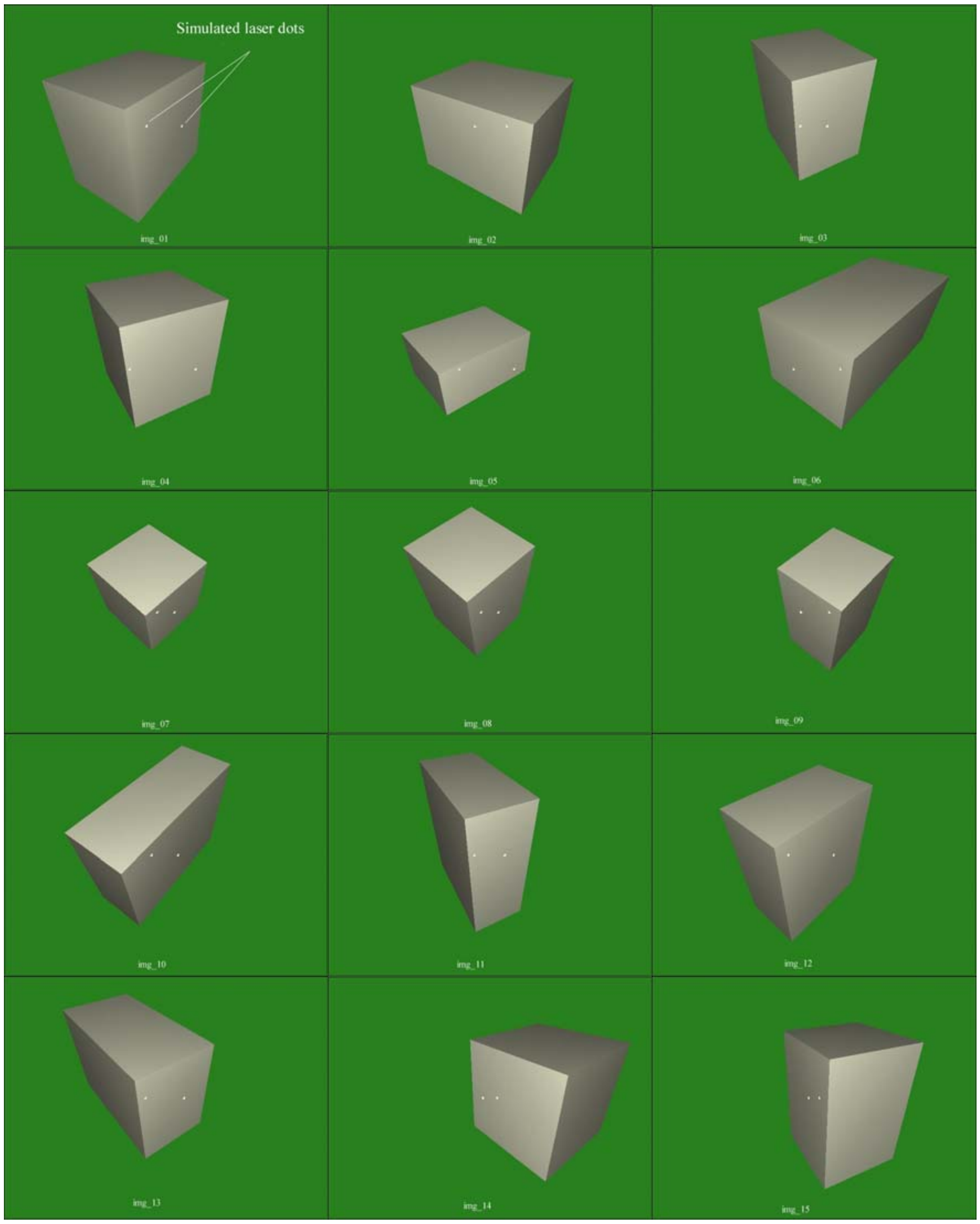


Figure 5.2. Images created by the simulator program cont.

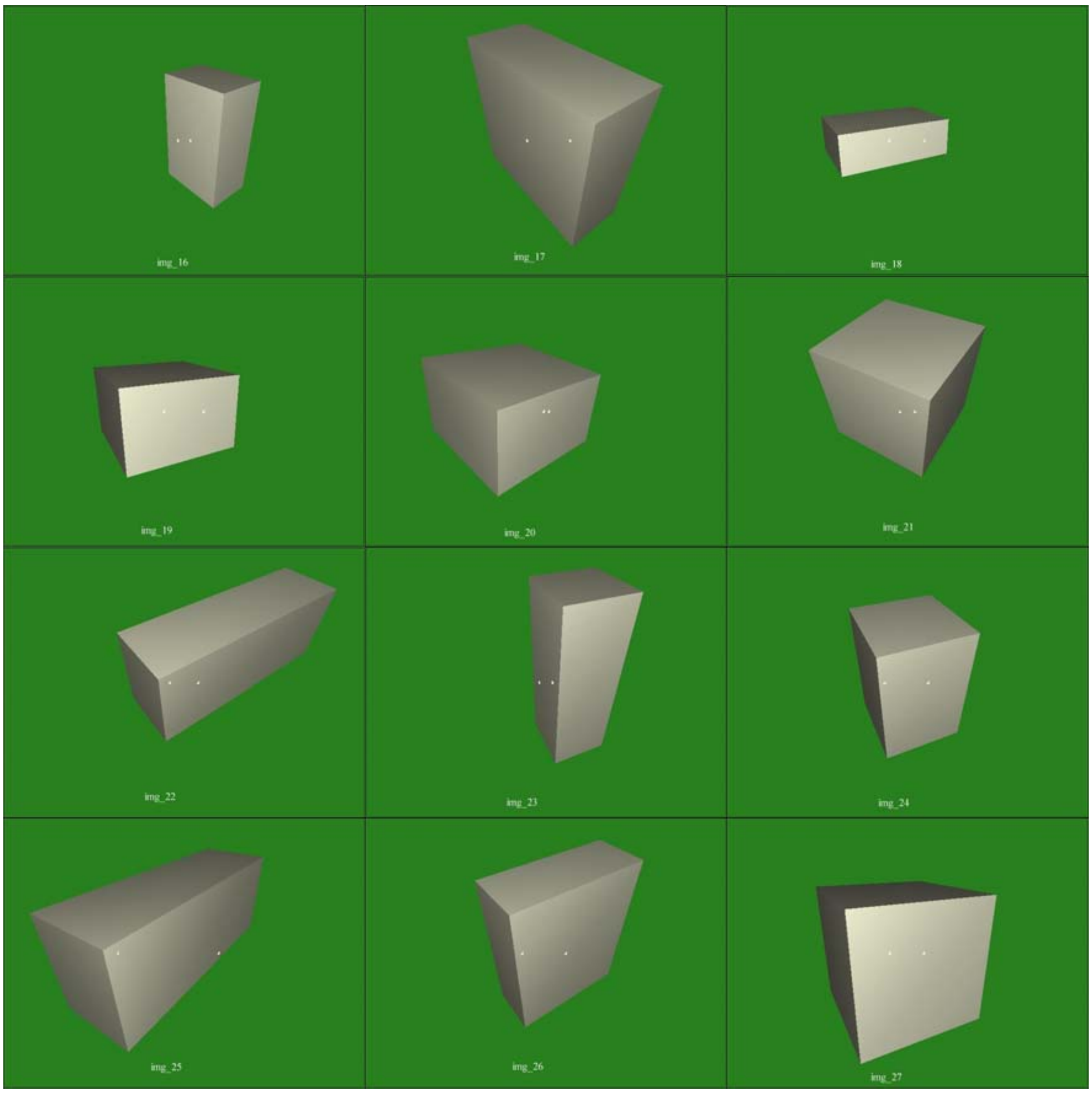


Figure 5.2. Images created by the simulator program

	Actual Measurments				Recovered Mearsurements			Errors (%)			
BOX	Height	Width	Length	Lasers	Height	Width	Length	Height	Width	Length	Vol Error
img_01	4.00	3.00	4.00	1.00	3.90	3.00	4.00	2.500	0.000	0.000	2.500
img_02	4.00	5.00	4.00	1.00	4.05	5.04	4.09	1.250	0.800	2.250	4.356
img_03	3.00	2.50	2.00	0.60	3.01	2.55	1.99	0.333	2.000	0.500	1.828
img_04	3.00	2.50	2.50	1.50	3.01	2.55	2.51	0.333	2.000	0.400	2.749
img_05	2.00	3.00	4.00	2.00	2.02	3.04	4.01	1.000	1.333	0.250	2.603
img_06	7.00	8.00	15.00	3.00	7.10	7.80	14.80	1.429	2.500	1.333	2.426
img_07	2.00	2.00	2.00	0.40	1.95	2.07	2.05	2.500	3.500	2.500	3.435
img_08	3.00	2.00	2.00	0.40	3.02	1.94	2.04	0.667	3.000	2.000	0.400
img_09	26.00	14.00	16.00	5.00	26.20	13.60	16.07	0.769	2.857	0.438	1.682
img_10	3.00	1.50	4.00	0.60	2.98	1.49	3.93	0.667	0.667	1.850	3.154
img_11	2.00	5.00	4.00	0.80	1.92	4.94	4.11	4.000	1.200	2.750	2.544
img_12	23.00	11.00	19.00	6.00	23.50	11.10	18.90	2.174	0.909	0.526	2.560
img_13	23.00	32.00	17.00	7.00	23.20	32.70	16.80	0.870	2.188	1.176	1.863
img_14	3.00	3.00	3.00	0.40	2.98	2.96	2.95	0.667	1.333	1.667	3.625
img_15	3.00	2.00	2.00	0.20	2.93	2.06	1.97	2.333	3.000	1.500	0.912
img_16	5.00	3.00	2.00	0.50	4.70	3.01	1.98	6.000	0.333	1.000	6.630
img_17	25.00	25.00	10.00	5.00	24.60	24.90	10.03	1.600	0.400	0.300	1.700
img_18	1.00	2.00	3.00	0.90	0.96	2.12	3.03	4.000	6.000	1.000	2.778
img_19	2.00	3.00	3.00	0.90	1.93	3.30	2.90	3.500	10.000	3.333	2.612
img_20	2.00	3.00	3.00	0.10	1.99	3.06	3.04	0.500	2.000	1.333	2.843
img_21	20.00	20.00	20.00	2.00	20.60	20.10	20.10	3.000	0.500	0.500	4.033
img_22	20.00	15.00	50.00	5.00	19.90	14.78	51.40	0.500	1.467	2.800	0.786
img_23	30.00	10.00	10.00	2.00	30.50	10.10	9.50	1.667	1.000	5.000	2.451
img_24	15.00	10.00	10.00	4.00	14.99	10.60	9.80	0.067	6.000	2.000	3.811
img_25	10.00	8.00	25.00	8.00	9.76	7.90	25.10	2.400	1.250	0.400	3.234
img_26	30.00	10.00	25.00	7.00	30.50	9.90	24.80	1.667	1.000	0.800	0.155
img_27	2.00	2.00	2.00	0.40	1.99	2.08	1.99	0.500	4.000	0.500	2.963

Table 5.1 Dimensions computed for the synthetic images

5.2 Errors in synthetic images

Despite the fact that the synthetic images were created in a perfectly controlled environment, some errors are still present. Most notably in some occasions, lines detected using the Hough transform are not the best fit for the points that voted for them (see figure 5.3.) In these cases, false lines may be used instead of the actual ones. Another issue related to the line detection is the Hough transform threshold. According to the current implementation, for a line to be considered in the process, at least 35 votes are required. This means that short edges (see figure 5.4) fail this test, resulting in incorrect results.

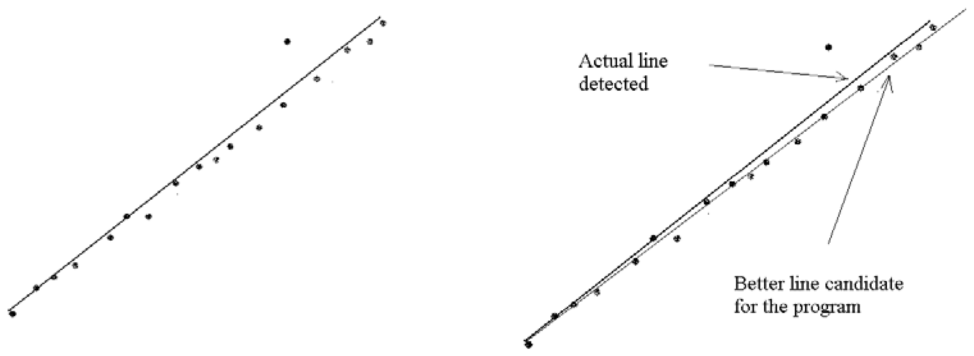


Figure 5.3 Line detected (left) contains errors; a better candidate is shown (right)

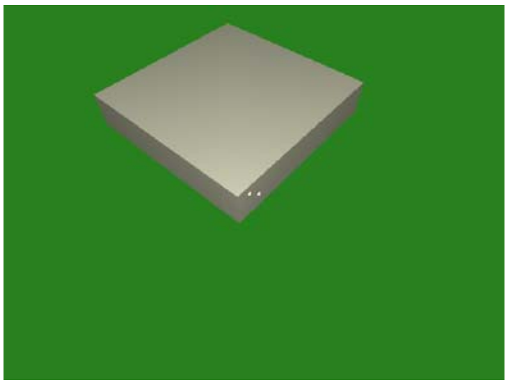


Figure 5.4. Synthetic box with short edges

The average percentage error for the dimensions of all the synthetic boxes considered is 1.7% for height, 2.2% for width, and 1.4% for length.

5.3 Results from real pictures

This section discusses the results obtained with the use of real images. The results are summarized in table 5.2. The images are presented in Figure 5.5.

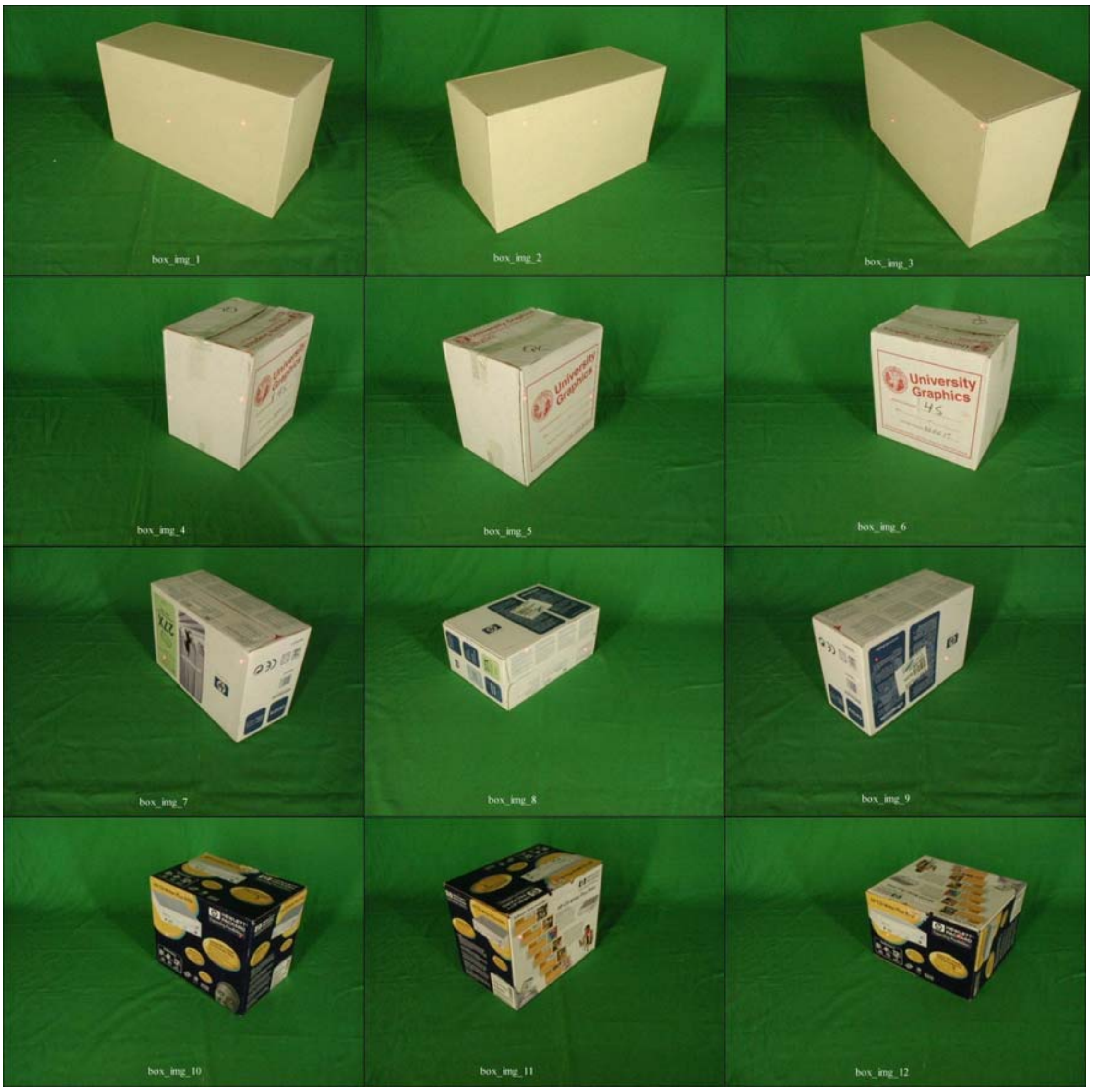


Figure 5.5. Pictures of real boxes used for test cont.

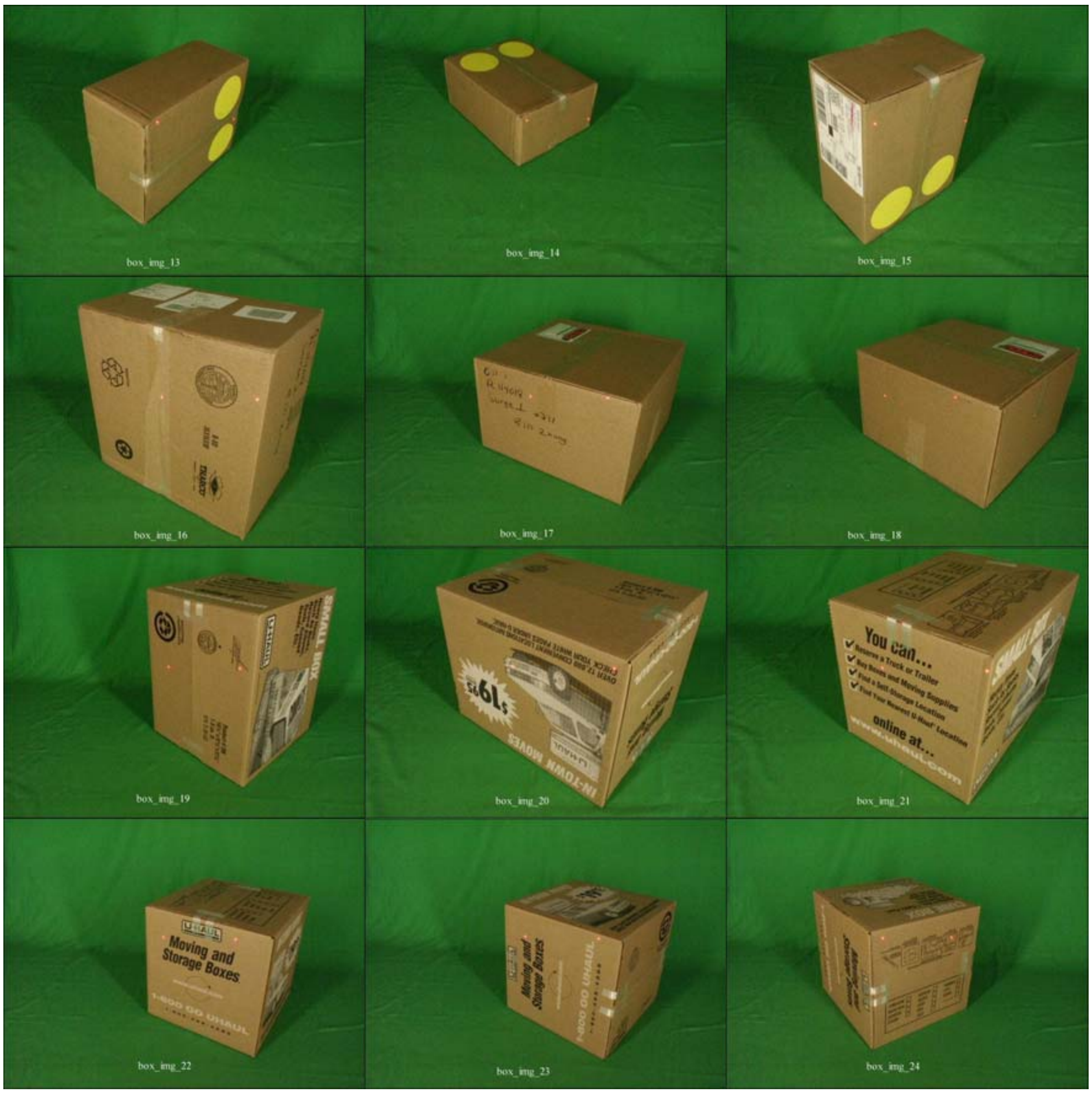


Figure 5.5. Pictures of real boxes used for test.

	Actual Measurments				Recovered Mearsurements			Error (%)			
BOX	Height	Width	Length	Lasers	Height	Width	Length	Height	Width	Length	Vol error
box_img_1	29.90	21.00	51.00	15.80	28.30	22.10	51.09	5.35	5.24	0.18	0.218
box_img_2	29.90	21.00	51.00	15.80	28.60	18.90	48.70	4.35	10.00	4.51	17.795
box_img_3	29.90	21.00	51.00	15.80	29.50	22.60	48.40	1.34	7.62	5.10	0.766
box_img_4	26.00	23.00	30.50	15.80	26.00	22.30	29.30	0.00	3.04	3.93	6.858
box_img_5	26.00	23.00	30.50	15.80	26.90	23.80	28.80	3.46	3.48	5.57	1.093
box_img_6	26.00	23.00	30.50	15.80	28.10	23.10	29.60	8.08	0.43	2.95	5.344
box_img_7	24.00	15.50	35.50	15.80	22.80	16.60	33.80	5.00	7.10	4.79	3.130
box_img_8	24.00	15.50	35.50	15.80	23.30	14.60	34.10	2.92	5.81	3.94	12.160
box_img_9	24.00	15.50	35.50	15.80	22.20	14.90	34.60	7.50	3.87	2.54	13.335
box_img_10	25.30	19.50	30.00	15.80	24.50	19.70	28.70	3.16	1.03	4.33	6.408
box_img_11	25.30	19.50	30.00	15.80	22.70	19.50	28.80	10.28	0.00	4.00	13.866
box_img_12	25.30	19.50	30.00	15.80	19.20	23.60	29.50	24.11	21.03	1.67	9.685
box_img_13	25.30	14.60	32.50	15.80	23.30	14.60	31.60	7.91	0.00	2.77	10.455
box_img_14	25.30	14.60	32.50	15.80	25.30	13.00	32.30	0.00	10.96	0.62	11.507
box_img_15	25.30	14.60	32.50	15.80	30.00	14.10	24.00	18.58	3.42	26.15	15.434
box_img_16	41.30	26.50	41.50	15.80	39.20	26.50	39.90	5.08	0.00	3.86	8.744
box_img_17	41.30	26.50	41.50	15.80	38.00	24.90	40.00	7.99	6.04	3.61	16.670
box_img_18	41.300	26.500	41.500	15.80	39.500	25.200	40.000	4.358	4.906	3.614	12.338
box_img_19	33.50	32.50	42.00	15.80	32.00	30.90	41.00	4.48	4.92	2.38	11.343
box_img_20	33.50	32.50	42.00	15.80	31.00	31.90	41.90	7.46	1.85	0.24	9.387
box_img_21	33.50	32.50	42.00	15.80	31.10	32.50	41.70	7.16	0.00	0.71	7.827
box_img_22	31.10	31.30	31.10	15.80	30.30	30.40	28.90	2.57	2.88	7.07	12.068
box_img_23	31.10	31.30	31.10	15.80	29.10	30.30	29.20	6.43	3.19	6.11	14.954
box_img_24	31.10	31.30	31.10	15.80	29.85	28.70	29.82	4.02	8.31	4.12	15.614

Table 5.2 Dimensions computed for real images

5.4 Analysis of results of real images

In order to understand the meaning of the numbers in Table 5.2, several factors have to be taken into consideration. The first of these factors is the fact that boxes are not perfect parallelepipeds. For instance, discrepancy errors of up to 1cm were observed when two parallel edges of the same face were measured using a tape. This happens when the edges of the box are bended or when the measurements of the box are not consistent from edge to edge due to imperfections in the box fabrication.

The average percentage error per dimension is 6.74% for height, 4.69% for width, and 4.77% for length with no apparent justification for these differences. In most cases, the results obtained on real boxes are quite good. Errors are bound to be less than 17% in the worst case of the calculation of the volume. In our experiments, the dimensional error is much less than the presented average suggests. In the worst case, a 26% error is found in the measurement of length. However, this was rather an outlier event and not the norm or average behavior. For most dimensions, measurements fall in a range of 0.1% to 6.0%. It must be noted that the error calculated for the volume is the result of propagation of the errors calculated for the dimensions of the box. Consider for instance two dimensions of the box for which error has been calculated; so that one of the calculated dimensions is bigger than the actual dimension, and in the other case smaller both by a similar amount. The error computed for the box would be somewhat small because the error in the two measurements would cancel out each other on the calculation of the volume. For example, box_img_1 and box_img_3 are examples of this situation. In particular, observe how in box_img_3 the width and length are above and below the actual measurements yet the error computer is relatively small.

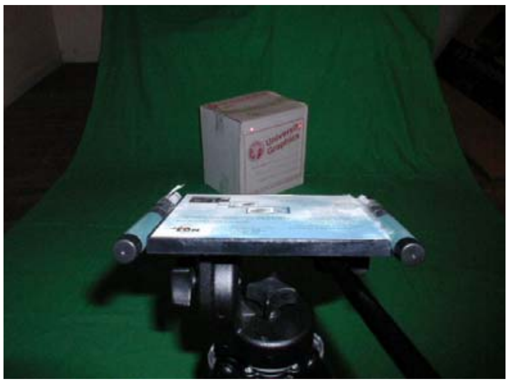


Figure 5.6. The laser construction on the tripod

There are several inconsistencies in real boxes that form part of the analysis of error. For instance, bended edges, different sizes for two parallel edges of the same face, warped corners, etc. The geometry of the box is somewhat different from that of a parallelepiped because of the imperfections introduced during the box construction. Because of these inconsistencies the missing corner (v6, refer to figure 3.6 in Chapter 3) of the box may be misplaced or wrongly located. In this case, since this point is used to find dimensions of the box, some error will be introduced as well (see figure 5.7). The magnitude of the error in these situations depends directly on the distance of the camera to the box. This follows form the fact that there are errors in the pixel location of the corner, and the distance that we found in the program multiplies this error thus increasing the error magnitude. Hence, the further away the camera is, the bigger the error factor that we obtain in the calculations Another factor that contribute to the location of a false missing corner is the Hough transform.

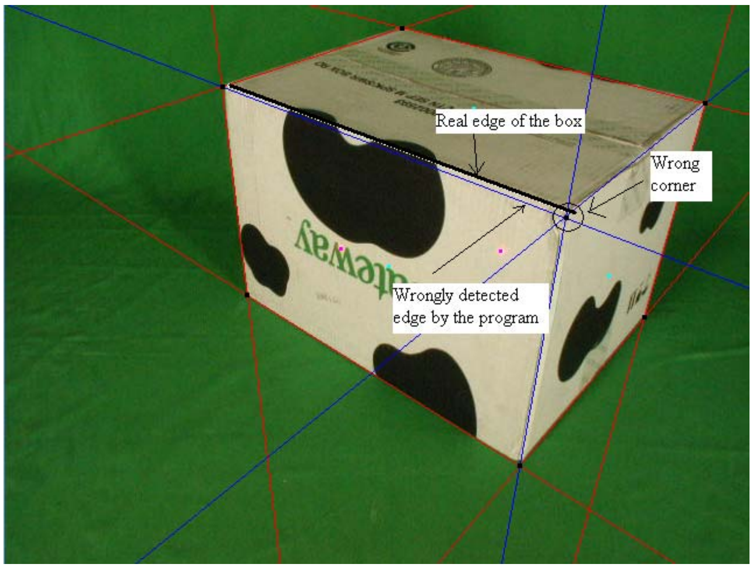


Figure 5.7 Missing edge

Similar to the case of synthetic images discussed above (see section 5.2 for a discussion on this issue), if the boxes for which this method is applied have short edges, then errors and failure are likely to happen. One example of this behavior is the FedEx box (see figure 5.8.) This happens because the threshold in the Hough transform is not low enough to detect these short edges.

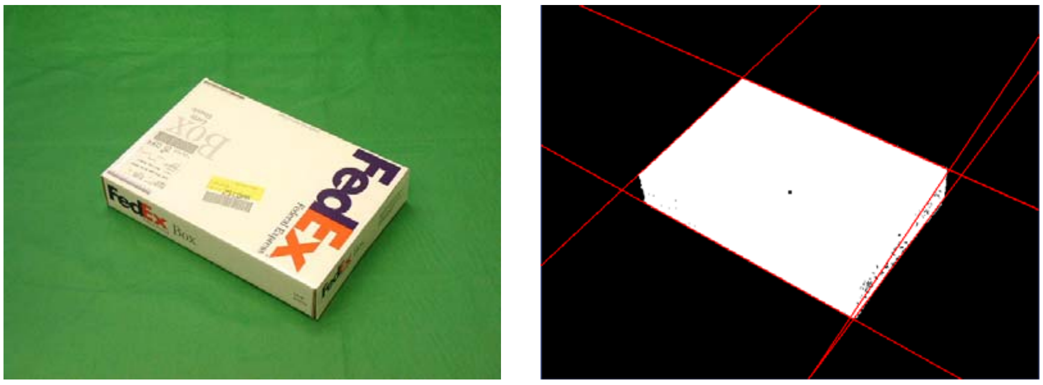


Figure 5.8. Error in detection of edges

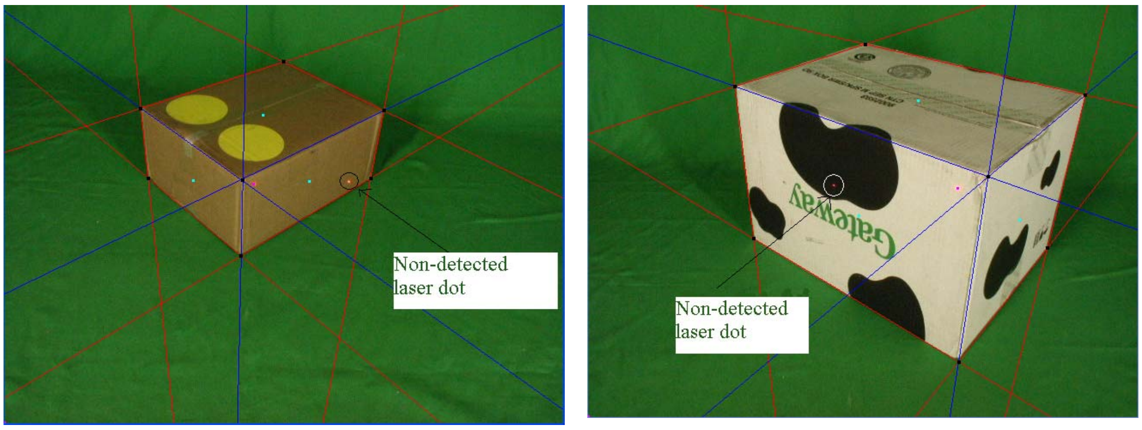


Figure 5.9. Errors in finding the laser dot on the image

Another factor that makes the method completely fail is the lack of detecting the two laser dots projected on the image. This situation can happen if the lighting condition is not appropriate or the surface on which the lasers are being projected has some texture/color combination that makes it difficult to distinguish them (see figure 5.9).

5.5 Comparison of results

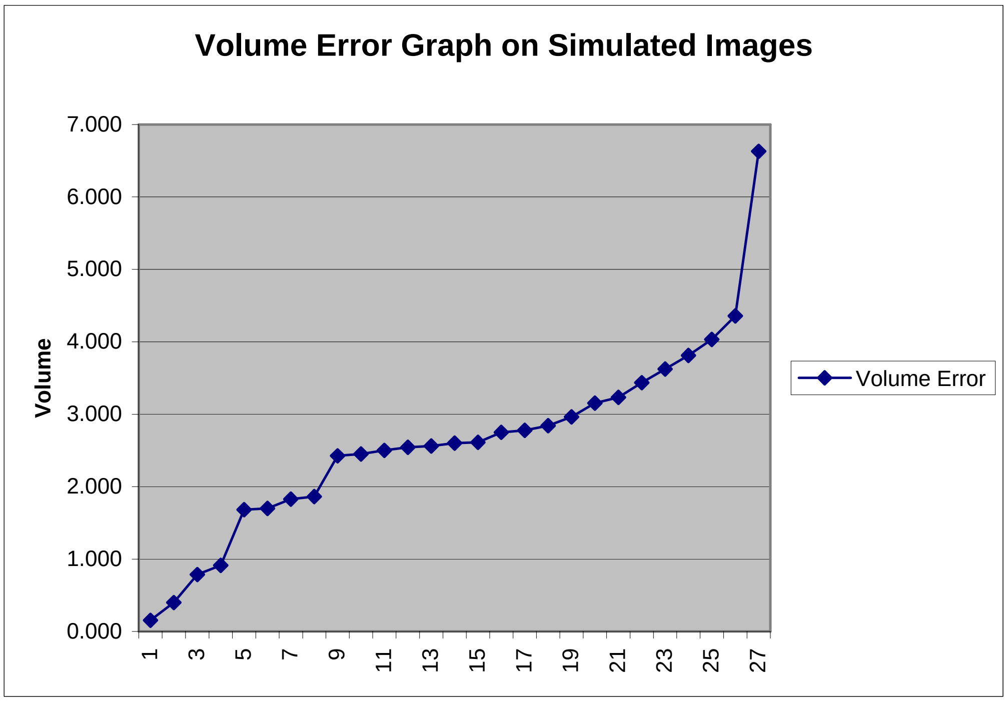
It is evident from Table 1 and Table 2 that the results from the simulated images are more accurate than those of the real images. Several factors contribute to such difference, and can be summarized as follows: errors inherent to the geometry of the box versus the perfect geometry of the simulated parallelepiped, cleaner edges in the simulated box versus the ragged edges of the real box, and better control of the parallelism of the laser dots on the simulation versus the laser dot apparatus in the real images.

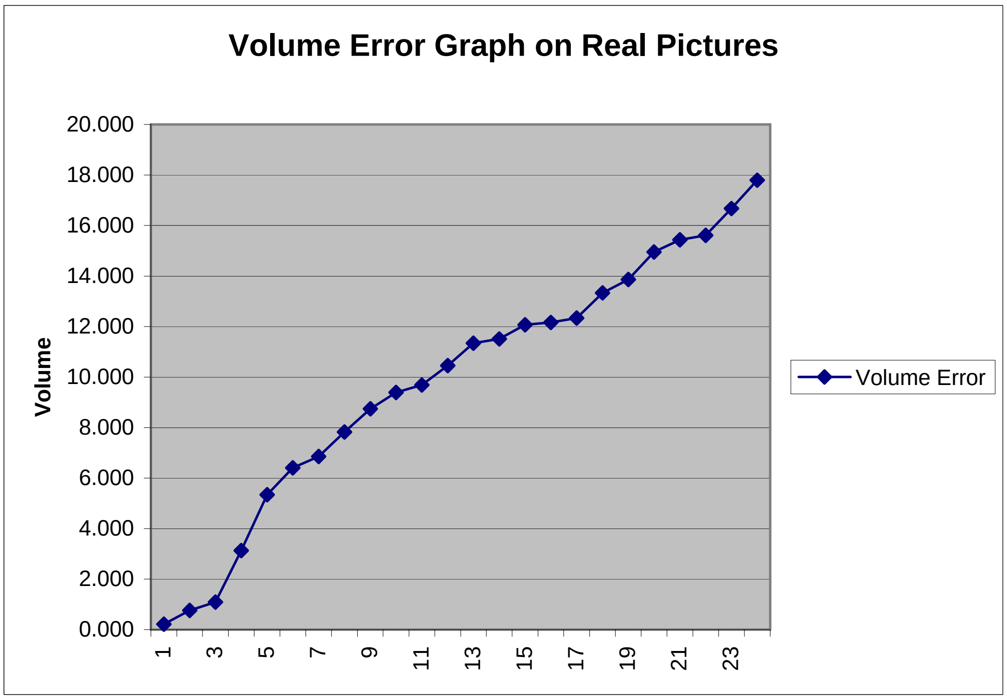
It is apparent that a real box contains some imperfection due to the methods used in the construction of the box. While it is true that the general shape of a box resembles that of a parallelepiped, it can be observe, for example, that two parallel edges on the same face may not have the same exact measurements. This inconsistency creates errors when calculating volumes from real images because the endpoints of these edges are used to compute its length. Thus if two edges have different measurements, and only one of this is used to determine one of the dimensions of the box, then we will obtain an erroneous result. This situation does not happen in the simulated box.

Since the simulated images contain clean edges, it is easy for the Hough transform to locate the right lines corresponding to these edges. However, this is not the case with the real boxes. Because the edges of the box are not perfect line segments, several points that lie outside of the real edge of the box vote for lines in the Hough transform. This created ambiguity, and erroneous edges can be detected because of this type of error (See figure 5.3).

The last type of error corresponds to the apparatus used for the projection of the laser dots on the box. Because the simulation can position the camera and the laser dots in a perpendicular position (angle of 90 degrees) we are assure that very little error is created. However, in the case of the real images this is not the case. Even though a certain degree of accuracy can be claimed with regard to the parallelism between the laser pointers, the degree of perpendicularity of the laser apparatus and the camera itself is not that of a perfect 90 degrees. This creates some error in the calculation since we are assuming that this is the case. This can be improved by creating a casing for the lasers and the camera so that both are bolted together.

All these sources of errors contribute to the difference in value between the simulation and the real images. Some of these problems, as is the case with the laser pointers, can be fixed. By observing the graphs below (see graphs 5.1) one get another insight on the behavior of errors in the synthetic, as well as real images. In the synthetic case, we can see how the error seems to behave in a pattern once the outliers are taken out of the equations. For example, for measurements between 9 and 25 the error falls in the range from 2 to 4 cubic centimeters. The real images behave in a similar manner, but the slope is a little steeper. This is an indication of the wider distribution of error in the real images.





Graphs 5.1. Error graphs

In this chapter, the results of the implementation of the theory discussed in Chapter 4 are presented. As the data suggest, the method works almost perfectly for the simulated images, and it gives good results for the case of real boxes. Possible ways for improving these results will be discussed in the next chapter.

Chapter 6 Conclusion

A new method for calculating volumes of arbitrary boxes based on ideas presented in [Oliveira 01] has been implemented. All the ideas related to this method have been thoroughly covered in this report. We also implemented a simulation environment to ease in the creation of synthetic boxes for testing and development. These programs can be combined in a single environment to facilitate further development; however, as of this writing these tools have not been combined yet. The main volume calculation environment has all the functionality needed to calculate the volume of boxes given the proper image. All the theory covered in this report plays a role in the underlying code of the volume finding program.

In conclusion, the implementation presented here is a faithful reproduction of the theory discussed in Chapter 3 and 4. This implementation has been thoroughly tested with hundreds of images, and the results have been satisfactory as discussed in Chapter 5. It is left for further development the combination of the simulator and the volume calculation software. Other improvement would include the change of image format from JPEG to TIFF to improve on errors caused by the JPEG compression algorithm.

Bibliography:

J. Canny "A Computational Approach to Edge Detection," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 8, No. 6, Nov. 1986.

P. V. C. Hough, "Machine Analysis of Bubble Camber Pictures, International Conference on High Energy Accelerators and Instrumentation," CERN, 1959.

Camillo J. Taylor, "Reconstruction of Linearly Parameterized Models from Single Images with a Camera of Unknown Focal Length," CVPR 1999.

E. C. Freuder, "A Computer System for Visual Recognition Using Active Knowledge." AI-TR-345, Artificial Intelligence Laborarоty, Massachusetts Institute of Technology, 1976.

Takeo Kanade, "Recovery of the Three-Dimensional Shape of an Object from a Single View," Artificial Intelligence, 17 (1981), 409-460.

Yoshiaki Shirai, "Recognition of Real-World Objects Using Edge Cue," Compute Vision Systems, (1978), 353-362.

D. A. Huffman, "Realizable configurations of lines in picrutes of polyhedra," Machine Intelligence 6, 1971.

David G. Lowe, "Thee-Dimensional Object Recognition from Single Two-Dimensional Images," Artificial Intelligence, 31, 3 (March 1987) pp 355-395.

Manuel M Oliveira. "Computing Box Volume from Single Images," State University of New York at Stony Brook Technical Report TR01.10.31, Octorber 2001.