A Fast and Accurate Approach for Computing the Dimensions of Boxes from Single Perspective Images

Leandro A. F. Fernandes[1], **Manuel M. Oliveira**[1], **Roberto da Silva**[1] **& Gustavo J. Crespo**[2]

[1]Universidade Federal do Rio Grande do Sul
Instituto de Informática - PPGC - CP 15064
91501-970 - Porto Alegre - RS - BRAZIL
{*laffernandes, oliveira, rdasilva*}*@inf.ufrgs.br*

[2]Stony Brook University
Department of Computer Science - 11794
New York - NY - USA
{*gus_crespo*}*@yahoo.com*

Abstract

This paper describes an accurate method for computing the dimensions of boxes directly from perspective projection images acquired by conventional cameras. The approach is based on projective geometry and computes the box dimensions using data extracted from the box silhouette and from the projection of two parallel laser beams on one of the imaged faces of the box. In order to identify the box silhouette, we have developed a statistical model for homogeneous-background-color removal that works with a moving camera, and an efficient voting scheme for the Hough transform that allows the identification of almost collinear groups of pixels. We demonstrate the effectiveness of the proposed approach by automatically computing the dimensions of real boxes using a scanner prototype that implements the algorithms and methods described in the paper. We also present a discussion of the performed measurements, and an error propagation analysis that allows the method to estimate, from each single video frame, the uncertainty associated to all measurements made over that frame, in real-time.

Keywords: Computing dimensions of boxes, image-based metrology, extraction of geometric information from scenes, uncertainty analysis, real time.

Figure 1. Scanner prototype: (left) Its operation. (right) Camera's view from another position showing the recovered dimensions and uncertainty computed in real time.

1. INTRODUCTION

The ability to measure the dimensions of three-dimensional objects directly from images has many practical applications including quality control, surveillance, analysis of forensic records, storage management and cost estimation. Unfortunately, unless some information relating distances measured in image space to distances measured in 3D is available, the problem of making measurements directly on images is not well defined. This results from the inherent ambiguity of perspective projection caused by the loss of depth information.

This paper presents a method for computing box dimensions from single perspective projection images in a completely automatic way. The approach uses information extracted from the silhouette of the target boxes and can be applied when at least two of their faces are visible,

even when the target box is partially occluded by other objects in the scene (Figure 1). We eliminate the inherent ambiguity associated with perspective images by projecting two parallel laser beams, apart from each other by a known distance, onto one of the visible faces of the box. We demonstrate this technique by building a scanner prototype for computing box dimensions and using it to compute the dimensions of boxes in real time (Figure 1). This can be an invaluable tool for companies that manipulate boxes on their day-by-day operations, such as couriers, airlines and warehouses.

The paper presents a revised and significantly extended version of the work originally described in [9]. The new materials in this extended version include: (i) the use of variable-size elliptical Gaussian kernels in the Hough transform voting procedure (Section 5). The use of such kernels makes the transform more robust to discretization errors and allows the proper detection of support silhouette lines of boxes with bent edges; (ii) the determination of the plane spanned by the laser beams using a calibration procedure (Subsection 3.2.1). The use of calibrated data removed the only assumption in the original derivation [9] of the equations shown in Section 3 and improved the accuracy of the method; (iii) the statistical analysis of the results (Section 7.1) was significantly enhanced considering a larger number of real boxes and new graphs that improve the interpretation of these results. Such an analysis shows that the proposed approach is both accurate and precise; and (iv) a modeling of the error propagation along all steps of the algorithm that allows our system to estimate the uncertainty in the computed measurements in real time (Section 7.2).

The main contributions of this paper include:

- An algorithm for computing the dimensions of boxes in a completely automatic way in real-time (Section 3);
- An algorithm for extracting box silhouettes in the presence of partial occlusion of the box edges (Section 3.1);
- A statistical model of homogeneous background color for use with a moving camera under different lighting conditions (Section 4);
- An efficient voting scheme for identifying nearly collinear line segments with a Hough transform (Section 5);
- A derivation of how to estimate the error associated with the computed dimensions from a single image in real time (Section 7.2).

2. RELATED WORK

Many optical devices have been created for making measurements in the real world. Those based on active techniques project some kind of energy onto the surfaces of the target objects and analyze the reflected energy. Examples of active techniques include optical triangulation [1] and laser range finding [22] to capture the shapes of objects at proper scale [17], and ultrasound to measure distances [10]. In contrast, passive techniques rely only on the use of cameras for extracting the three-dimensional structure of a scene and are primarily based on the use of stereo [18]. In order to achieve metric reconstruction [13], both optical triangulation and stereo-based systems require careful calibration. For optical triangulation, several images of the target object with a superimposed moving pattern are usually required for accurate reconstruction.

Labeling schemes for trihedral junctions [4, 15] have been used to estimate the spatial orientation of polyhedral objects from images. These techniques tend to be computationally expensive when too many junctions are identified. Additional information from the shading of the objects can be used to improve the process. Silhouettes have been used in computer vision and computer graphics for object shape extraction [16, 21]. These techniques require precise camera calibration and use silhouettes obtained from multiple images to define a set of cones whose intersections approximate the shapes of the objects.

Criminisi *et al.* [5] presented a technique for making 3D affine measurements from a single perspective image. They show how to compute distances between planes parallel to a reference one. In case of some distance from a scene element to the reference plane is known, it is possible to compute the distances between scene points and the reference plane. If such a distance is not known, the computed dimensions are correct up to a scaling factor. The technique requires user interaction and cannot be used for computing dimensions automatically. Photogrammetrists have also made measurements based on single images [23], but these techniques also require user intervention.

In a work closely related to ours, Lu [20] described a method for finding the dimensions of boxes from single gray-scale images. In order to simplify the task, Lu assumes that the images are acquired using parallel orthographic projection and that three faces of the box are visible simultaneously. The computed dimensions are approximately correct up to a scaling factor. Also, special care is required to distinguish the actual box edges from lines in the box texture, causing the method not to perform in real time.

Our approach computes the dimensions of boxes from single perspective projection images, producing metric reconstructions in real time and in a completely automatic way. The method can be applied to boxes with arbitrary

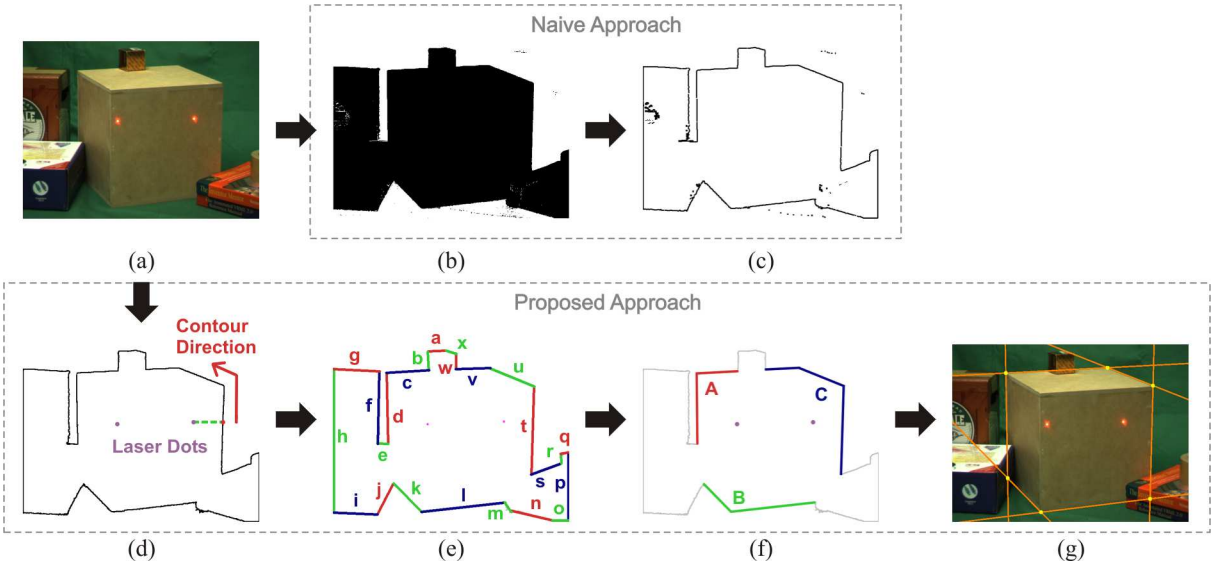

Figure 2. Identifying the target box silhouette. Naive approach: (b) Background segmentation, followed by (c) High-pass filter (note the spurious "edge" pixels). Proposed approach: (d) Contouring of the foreground region, (e) Contour segmentation, (f) Grouping candidate segments for the target silhouette, and (g) Recovery of supporting lines for silhouette edges and vertices.

textures, can be used when only two faces of the box are visible, even when the edges of the target box are partially occluded by other objects in the scene.

3. Computing Box Dimensions

We model boxes as parallelepipeds although real boxes can present many imperfections (*e.g.*, bent edges and corners, asymmetries, etc.). The dimensions of a parallelepiped can be computed from the 3D coordinates of four of its non-coplanar vertices. Conceptually, the 3D coordinates of the vertices of a box can be obtained by intersecting rays, defined by the camera's center and the projections of the box vertices on the camera's image plane, with the planes containing the actual faces of the box in 3D. Thus, before one can compute the dimensions of a given box (Section 3.3), its necessary to find the projections of the vertices on the image (Section 3.1), and then find the equations of the planes containing the box faces in 3D (Section 3.2).

In the following derivations, we assume that the origin of the image coordinate system is at the center of the image, with the X-axis growing to the right and the Y-axis growing down, and assume that the imaged boxes have three visible faces. The case involving only two visible faces is similar. Also, we assume that the images used for computing the dimensions were obtained through linear projection (*i.e.*, using a pinhole camera). Although images obtained with real cameras contain some amount of radial and tangential distortions, we compensate such distortions in real time with the use of simple warping procedures [13].

The number of visible faces of the box can be obtained by checking if the projection of some of the edges that share the same direction in 3D is (almost) parallel in 2D. In this case, two faces of the box are visible; otherwise, three faces are visible simultaneously. Although this approach has proven to produce good results for well-constructed boxes, most real boxes present some distorted edges, which breaks the parallelism assumption. Thus, in practice, it is more effective to assume that three box faces are visible. Since the system is capable of computing the dimensions of boxes at 30 Hz, we can afford to discard frames if the silhouettes recovered from the acquired images do not satisfy the imposed requirements. In this case, the perception of the user is similar to that of a barcode scanner user: if no answer is coming out, just slightly change the scanner's orientation relatively to the target object in order to get it.

3.1. Finding the Projections of the Vertices

The projection of the vertices can be obtained as the corners of the box silhouette. Although edge detection techniques [3] could be used to find the box silhouette, these algorithms tend to be very sensitive to the presence of other high-frequency contents in the image. In order to minimize the occurrence of spurious edges and support the use of boxes with arbitrary textures, we perform silhouette detection using a model for the background pixels. Since the images are acquired using a handheld camera, proper modeling of the background pixels is requi-

red and will be discussed in detail in Section 4.

However, as shown in Figure 2 (a, b and c), a naive approach that just models the background and applies simple image processing operations, like background removal and high-pass filtering, does not properly identify the silhouette pixels of the target box (selected by the user by pointing the laser beams onto one of its faces). This is because the scene may contain other objects, whose silhouettes possibly overlap with the one of the target box. Also, the occurrence of some misclassified pixels (see Figure 2, c) may lead to the detection of spurious edges. Thus, a suitable method was developed to deal with these problems. The steps of our algorithm are shown in Figures 2 (a, d, e, f and g).

In our approach, the target box silhouette is obtained starting from one of the laser dots, finding a silhouette pixel and using a contour-following procedure [12]. The seed silhouette pixel for the contour-following is found stepping from the laser dot within the target foreground region and checking whether the current pixel matches the background model. In order to be a valid silhouette, both laser dots need to fall inside of the contouring region. Notice this procedure produces a much cleaner set of border pixels (Figure 2, d) compared to results shown in Figure 2 (c). But the resulting silhouette may include overlapping objects, and one still needs to identify which border pixels belong to the target box. To facilitate the handling of the border pixels, the contour is subdivided into its most perceptually significant straight line segments [19] (Figure 2, e). Then, the segments resulting from the clipping of a foreground object against the limits of the frame (*e.g.*, segments h, o and p in Figure 2, e) are discarded. Since a box silhouette defines a convex polygon, the remaining segments whose two endpoints are not *visible* by both laser dots can also be discarded. This test is performed using a 2D BSP-tree [11]. In the example of Figure 2, only segments c, d, k, l, t, u and v pass this test.

Still, there is no guarantee that all the remaining segments belong to the target box silhouette. In order to restrict the amount of possible combinations, the remaining chains of segments defining convex fragments are grouped (*e.g.*, groups A, B and C in Figure 2, f). We then try to find the largest combination of groups into valid portions of the silhouette. In order to be considered a valid combination, the groups must satisfy the following validation rules: (i) they must characterize a convex polygon; (ii) the silhouette must have six edges (the silhouette of a parallelepiped with at least two visible faces); (iii) the laser dots must be on the same box face; and (iv) the computed lengths for pairs of parallel edges in 3D must be approximately the same. In the case of more than one combination of groups pass the validation tests, the system discards this ambiguous data and starts processing a new frame (our system is capable of processing frames at the rate of about 29 fps).

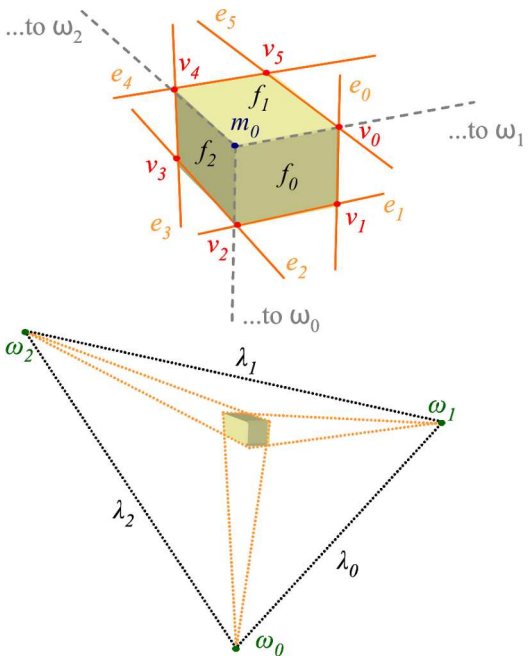

Figure 3. Vanishing points (ω_i) and vanishing lines (λ_i). e_j, v_j, m_0, and f_i are the supporting lines for silhouette edges, the silhouette vertices, the inner vertex and the faces of the box, respectively.

Once the box silhouette is known, the projections of the six vertices are obtained intersecting pairs of adjacent supporting lines for the silhouette edges (Figure 2, g). Section 5 discusses how to obtain those supporting lines.

3.2. Computing the Plane Equations

The set of all parallel lines in 3D sharing the same direction intersect at a point at infinite whose image under perspective projection is called a vanishing point ω. The line defined by all vanishing points from all sets of parallel lines on a plane Π is called the vanishing line λ of Π (Figure 3). The normal vector to Π in a given camera's coordinate system can be obtained multiplying the transpose of the camera's intrinsic-parameter matrix by the coefficients of λ [13]. Since the resulting vector is not necessarily a unit vector, it needs to be normalized. Equations (1) and (2) and Figure 3 show the relationship among the vanishing points ω_i, vanishing lines λ_i and the supporting lines e_j for the edges that coincide with the imaged silhouette of a parallelepiped with three visible faces. The supporting lines are ordered clockwise.

$$\omega_i = e_i \times e_{i+3} \tag{1}$$

$$\lambda_i = \omega_i \times \omega_{(i+1) mod 3} \tag{2}$$

where $0 \leq i \leq 2$, $0 \leq j \leq 5$, $\lambda_i = (a_{\lambda_i}, b_{\lambda_i}, c_{\lambda_i})^T$ and $\times$ is the cross product operator. The normal N_{Π_i} to plane

Π_i is then given by

$$N_{\Pi_i} = \frac{RK^T\lambda_i}{\|RK^T\lambda_i\|} \tag{3}$$

where $N_{\Pi_i} = (A_{\Pi_i}, B_{\Pi_i}, C_{\Pi_i})$. K is the matrix that models the intrinsic camera parameters [13] and R is a reflection matrix (Equation 4) used to make the Y-axis of the image coordinate system grows in the up direction.

$$K = \begin{pmatrix} \alpha_x & \gamma & o_x \\ 0 & \alpha_y & o_y \\ 0 & 0 & 1 \end{pmatrix}, \quad R = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{4}$$

In Equation (4), $\alpha_x = f/s_x$ and $\alpha_y = f/s_y$, where f is the focal length, and s_x and s_y are the dimensions of the pixel in centimeters. γ, o_x and o_y represent the skew and the coordinates of the principal point, respectively.

Once we have N_{Π_i}, finding D_{Π_i}, the fourth coefficient of the plane equation, is equivalent to solving the projective ambiguity and will require the introduction of one more constraint. Thus, consider the situation depicted in 2D in Figure 4 (left), where two laser beams, parallel to each other, are projected onto one of the faces of the box. Let the 3D coordinates of the laser dots defined with respect to the camera coordinate system be $P_0 = (X_{P_0}, Y_{P_0}, Z_{P_0})^T$ and $P_1 = (X_{P_1}, Y_{P_1}, Z_{P_1})^T$, respectively (Figure 4, right). Since P_0 and P_1 are on the same plane Π, one can write

$$A_\Pi X_{P_0} + B_\Pi Y_{P_0} + C_\Pi Z_{P_0} = A_\Pi X_{P_1} + B_\Pi Y_{P_1} + C_\Pi Z_{P_1} \tag{5}$$

Using the linear projection model and given $p_i = (x_{p_i}, y_{p_i}, 1)^T$, the homogeneous coordinates of the pixel associated with the projection of point P_i, one can reproject p_i on the plane $Z = 1$ (in 3D) using

$$p'_i = (x_{p'_i}, y_{p'_i}, 1)^T = RK^{-1}p_i \tag{6}$$

and express the 3D coordinates of the laser dots on the face of the box as

$$X_{P_i} = x_{p'_i} Z_{P_i} \;\;,\;\; Y_{P_i} = y_{p'_i} Z_{P_i} \;\; and \;\; Z_{P_i} \tag{7}$$

Substituting the expression for X_{P_0}, Y_{P_0}, X_{P_1} and Y_{P_1} (Equation 7) in Equation (5) and solving for Z_{P_0}, we obtain

$$Z_{P_0} = kZ_{P_1} \tag{8}$$

where

$$k = \frac{A_\Pi x_{p'_1} + B_\Pi y_{p'_1} + C_\Pi}{A_\Pi x_{p'_0} + B_\Pi y_{p'_0} + C_\Pi} \tag{9}$$

Now, let d_{lb} and d_{ld} be the distances, in 3D, between the two parallel laser beams and between the two laser dots projected onto one of the faces of the box, respectively (Figure 4). Section 6 discusses how to find the laser dots on the image. d_{ld} can be directly computed from N_Π,

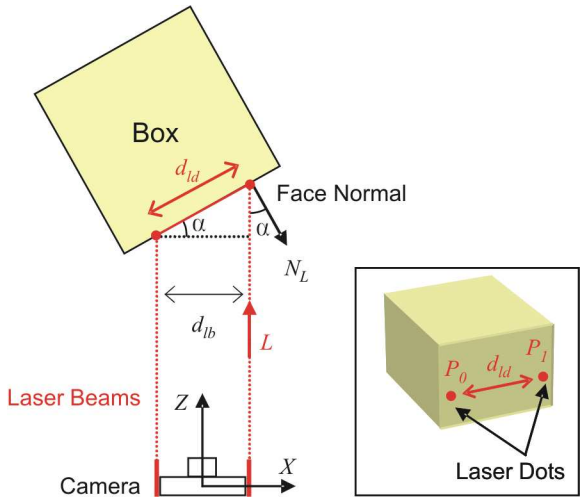

Figure 4. Top view of a scene. Two laser beams apart in 3D by d_{lb} project onto one box face at points P_0 and P_1, whose distance in 3D is d_{ld}. α is the angle between $-L$ and N_L.

the normal vector of the face onto which the dots project, and the known distance d_{lb}:

$$d_{ld} = \frac{d_{lb}}{\cos(\alpha)} = \frac{d_{lb}}{-(N_L \cdot L)} \tag{10}$$

where α is the angle between N_L, the normalized projection of N_Π onto the plane defined by the two laser beams, and L is the vector representing the laser beam direction. For now, we will assume that the laser plane is parallel to the camera XZ plane and $L = (0, 0, 1)^T$. Therefore, N_L is obtained by dropping the Y coordinate of N_Π and normalizing the resulting vector. d_{ld} can also be expressed as the Euclidean distance between the two laser dots in 3D:

$$d_{ld}^2 = (X_{P_1} - X_{P_0})^2 + (Y_{P_1} - Y_{P_0})^2 + (Z_{P_1} - Z_{P_0})^2 \tag{11}$$

Substituting Equations (7), (8) and (10) into (11) and solving for Z_{P_1}, one gets

$$Z_{P_1} = \sqrt{\frac{d_{ld}^2}{ak^2 - 2bk + c}} \tag{12}$$

where $a = (x_{p'_0})^2 + (y_{p'_0})^2 + 1$, $b = x_{p'_0}x_{p'_1} + y_{p'_0}y_{p'_1} + 1$ and $c = (x_{p'_1})^2 + (y_{p'_1})^2 + 1$. Given Z_{P_1}, the 3D coordinates of P_1 can be computed as

$$P_1 = (X_{P_1}, Y_{P_1}, Z_{P_1})^T = (x_{p'_1}Z_{P_1}, y_{p'_1}Z_{P_1}, Z_{P_1})^T \tag{13}$$

The projective ambiguity can be finally removed by computing the D_Π coefficient for the plane equation of the face containing the two dots:

$$D_\Pi = -(A_\Pi X_{P_1} + B_\Pi Y_{P_1} + C_\Pi Z_{P_1}) \tag{14}$$

3.2.1. Estimating the Laser Plane: In practice, it is difficult to guarantee that the plane defined by the laser beams is parallel to the camera's XZ plane, and that the L vector is aligned with the camera Z-axis. In our scanner prototype, we noticed that although the laser beams are parallel to each other, the plane they define (Π_{lb}) is not parallel to the camera's XZ plane. Therefore, it is necessary to take into account the angle between these two planes before computing N_L and then d_{ld} (Equation 10).

The orientation of Π_{lb} and the direction of the L vector were estimated projecting the laser beams on a planar checkerboard calibration pattern, placed at varying distances from the scanner. By collecting the coordinates of a set of 3D points (corresponding to these projections) along the laser lines, we estimated both Π_{lb}'s orientation and L's direction with respect to the camera's coordinate system.

3.3. Computing the Box Dimensions

Having computed the plane equation of a face of the box, one can recover the 3D coordinates of vertices of that face. For each such vertex v on the image, we compute v' using Equation (6). We then compute its corresponding Z_V coordinate by substituting Equation (7) into the plane equation for the face. Given Z_V, both X_V and Y_V coordinates are computed using Equation (7). Since all visible faces of the box share some vertices with each other, the D coefficients for the other faces of the box can also be obtained, allowing the recovery of the 3D coordinates of all vertices on the box silhouette, from which the dimensions are computed.

Although not required for computing the dimensions of the box, the 3D coordinates of the inner vertex m_0 (Figure 3, top) can also be computed. Its 2D coordinates are obtained as the intersection among three lines. Each such line is defined by a vanishing point and the silhouette vertex falling in between the two box edges used to compute that vanishing point. This situation is illustrated in Figure 3. Since it is unlikely that these three lines will intersect exactly at one point, we approximate this intersection using least-squares. Given the inner vertex 2D coordinates, its corresponding 3D coordinates are computed using the same algorithm used to compute the 3D coordinates of the other vertices.

4. A Model for Background Pixels

In order to obtain the box silhouette, we need to classify the pixels as either background or foreground pixels. One of the most popular techniques for object segmentation is chroma keying [24]. Unfortunately, standard chroma keying techniques do not produce satisfactory results for our application. Shading variations in the background and shadows cast by the boxes usually lend to misclassification of background pixels. Horprasert *et al.* [14] describe a statistical method that computes a per-pixel model of the background from a set of static background images. While this technique is fast and produces very good segmentation results for scenes acquired from a static camera, it is not appropriate for use with moving cameras. Also it requires a complete new calibration when the lighting conditions change too much. To avoid problems from lighting changes, a threshold solution based on hue component of the HSV color space might seem to be a good solution. However, such an approach tends to misclassify foreground pixels whose colors are close to the background color.

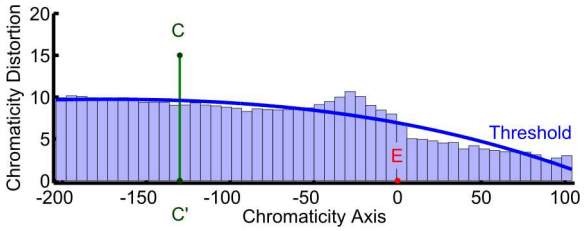

Figure 5. Chromaticity axis rotated to align to a horizontal axis. The curve is a polynomial fit to the chromaticity distortion threshold for each slice (the small rectangles).

In order to support a moving camera, we have developed an approach that proved to be robust, lending to very satisfactory results. It works under different lighting conditions by computing a statistical model of the background, which contains a single hue. Such a model is defined by a *chromaticity axis* that represents the mean expected shade of the background under various lighting conditions and a *polynomial curve* describing a variable threshold along the chromaticity axis.

The algorithm takes as input a set of n images I_i of the background acquired under different lighting conditions. In the first step, we compute E, the average color of all pixels in all images I_i, and the eigenvalues and eigenvectors associated with the colors of those pixels. E and the eigenvector associated with the highest eigenvalue define an axis in the RGB color space (the chromaticity axis). The chromaticity distortion d of a given color C is computed as the distance from C to the chromaticity axis.

After discarding the pixels whose projections on the chromaticity axis have at least one saturated channel (they lend to misclassification of bright foreground pixels), we subdivide the chromaticity axis into m slices (Figure 5). For each slice, we compute $\bar{d}_j$ and $\sigma_{\bar{d}_j}$, the mean and the standard deviation, respectively, for the chromaticity distortion of the pixels in the slice. Then, we compute a threshold d_{Tj} for the maximum acceptable slice-chromaticity distortion considering a confidence level of

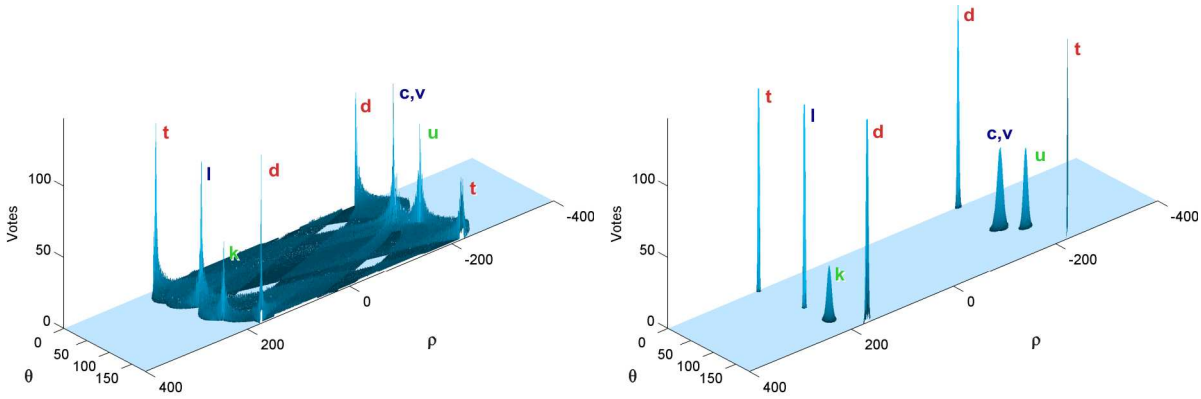

Figure 6. Hough Transform parameter space obtained with the conventional (left) and the new voting scheme (right) applied to the segments shown in Figure 2 (f). The peaks represent the supporting lines for silhouette edges.

99% as $d_{Tj} = \bar{d}_j + 2.33\sigma_{\bar{d}_j}$.

Finally, the coefficients of a polynomial that models the chromaticity-distortion thresholds are computed by fitting a curve through the d_T values at the centers of the slices (Figure 5). Intuitively, such a polynomial describes a variable threshold for the different shades of the background color. Once the coefficients have been computed, the d_T values are discarded and the tests are performed against the model. Figure 5 illustrates the case of a color C being tested against the background color model. C' is the projection of C on the chromaticity axis. In this example, as the distance between C and the chromaticity axis is bigger than the threshold defined by the polynomial, C will be classified as foreground.

Changing the background color only requires obtaining samples of the new background and computing the new values for the chromaticity axis and the coefficients of the polynomial. It is possible that the box texture may contain some pixels whose chromaticity distortions are smaller than the threshold defined by the polynomial for a given background color shade. In this case, the classification process would incorrectly indicate the presence of background pixels inside the box region. However, the silhouette detection approach described in Section 3.1 can handle groups of misclassified foreground pixels and, in practice, no problems have been noticed as a result of possible such misclassification. According to our experiments, 100 slices and a polynomial of degree 3 produce very satisfactory results.

5. Identifying Almost Collinear Segments

To compute the image coordinates of the box vertices, first we need to obtain the supporting lines for the silhouette edges. We do this using a Hough transform procedure [7]. However, the conventional voting process and the detection of the most significant lines is computationally intensive and turned out to be a bottleneck to our system. To reduce the amount of computation, an alternative to the conventional voting process was developed.

As seen in Section 3.1, silhouette pixels are organized into perceptually most significant straight-line segments. The new voting scheme consists in casting votes directly for these segments, instead of for individual pixels as it is traditionally done [7]. Thus, for each segment, the (ρ, θ) parameters of its supporting line are computed from the average position of the set of pixels defining the segment and from the 2D eigenvectors of that pixel distribution. The eigenvector with the smaller eigenvalue is the normal to the line, so ρ can be computed as the dot product between this eigenvector and the average pixel. θ is the angle between the X-axis of the image and the secondary eigenvector. The use of eigenvectors makes the process robust, allowing it to handle lines with arbitrary orientations in a consistent way.

For each segment, we distribute its votes in the parameter space using a Gaussian elliptical kernel (Figure 6, right), whose central position is defined by the (ρ, θ) parameters of the line fit to its set of pixels. The Gaussian kernel spread the votes over a region of the parameters space around (ρ, θ), according to the quality of the fit. Notice however, that different segments have different numbers of pixels, as well as have different degrees of dispersion around their corresponding best-fitting lines. The smaller the dispersion, the more concentrated these votes should be in the parameter space. We estimate the quality of the line fit by computing the variances and covariance of the (ρ, θ) parameters. The variances give the dimensions of the axes of the elliptical kernel, while the covariance gives the orientation of the ellipse. One can compute the variances and covariance of ρ and θ using a linear regression procedure [6]. However, standard linear regression procedures use the slope-intercept line notation, so one can use

a first order uncertainty propagation analysis [25] to compute the variances and covariance of ρ and θ from the variances and covariances of the slope-intercept parameters. Once one has computed the variances and covariance associated with ρ and θ, the votes are cast using a bi-variated Gaussian distribution [26]. The use of a Gaussian kernel distributes the votes around a neighborhood, allowing the identification of approximately collinear segments. This is a very important and unique feature of our approach that allows our system to better handle discretization errors and boxes with slightly bent edges. A detailed explanation about the proposed voting process can be found in [8].

Using the new approach, the voting process and the peak detection are significantly improved because the amount of cells that receive votes is substantially reduced. Figure 6 shows the parameter space after the traditional (left) and the new (right) voting processes have been applied to the segments shown in Figure 2 (f). Using the conventional (i.e., per-pixel) voting scheme [7], $376,884$ votes are distributed over $228,255$ cells. In contrast, the new approach only casts $6,382$ votes distributed over $5,020$ cells, which represents 1.7% of the number of votes and 2.2% of the number of cells used in the conventional approach. As a result, the produced voting map is very clean (Figure 6, right), reducing ambiguities and improving the identification of the most important lines. The extra cost involved in computing the covariance matrices associated with a few segments and by the use of Gaussian elliptical kernels to cast votes is more than compensated by the huge saving achieved.

Special care must be taken when the θ parameter is close to 0° or to 180°. In this situation, the voting process continues in the diagonally opposing quadrant, at the $-\rho$ position (see Figure 6, peaks d and t). For the examples shown in the paper, the parameter space was discretized using 360 angular steps in the range $\theta = [0^\circ, 180^\circ)$ and $1,600$ ρ values in the range $[-400, 400]$.

6. FINDING THE LASER DOTS

The ability to find the proper positions of the laser dots in the image can be affected by several factors such as the camera's shutter speed, the box materials and textures, and ambient illumination. Although we are using a red laser (650 nm class II), we cannot rely simply on the red channel of the image to identify the positions of the dots. Such a procedure would not distinguish between the laser dot and red texture elements on the box. Since the pixels corresponding to the laser dots present very high luminance, we identify them by thresholding the luminance image. However, just simple thresholding may not work for boxes containing white regions, which tend to have large areas with saturated pixels. We solved this problem by setting the camera's shutter speed so that the laser dots are the only elements in the image with high luminance.

Since the image of a laser spot is composed by several pixels, we approximate the actual position of the dot by the centroid of its pixels. According to our experiments, a variation of one pixel in the estimated center of the laser spot produces a variation of a few millimeters in the computed dimensions. These numbers were obtained assuming a camera standing about two meters from the box. After computing the position of the inner vertex (Section 3.3), the face that contains the laser dots is identified.

The system may fail to properly detect the laser dots if they project on some black region or if the surface exhibits specular peaks. This, however, can be avoided by aiming the beams on other portions of the box. Due to the construction of the scanner prototype and to some epipolar constraints [13], one only needs to search for the laser dots inside a small window in the image. Although a single laser beam could be used to break the projective ambiguity, the use of two beams introduces additional constraints that make silhouette identification more robust.

7. RESULTS

We have built a prototype of a scanner for computing box dimensions and implemented the techniques described in the paper using C++. The system was tested on several real boxes. For a typical scene, such as the one shown in Figure 2, it can process video and compute box dimensions at about 29 fps. For comparison, the frame rate drops to 10 fps if the traditional pixel-based Hough-transform voting scheme (Figure 6, left) is used. Such numbers illustrate the effectiveness of the proposed voting solution. These measurements were made on a 2.8 GHz PC with 1.0 GB of memory. A video sequence illustrating the use of our scanner can be found at http://www.inf.ufrgs.br/~laffernandes/boxdimensions.

Figure 1 (left) shows the scanner prototype whose hardware is comprised of a firewire color camera (Point Grey Research DragonFly with 640×480 pixels), a 16 mm lens (Computar M1614, with manual focus, no iris and 30.9 degrees horizontal field of view) and two laser pointers. The camera is mounted on a plastic box and the laser pointers were aligned and glued to the sides of this box. In such an assembly, the laser beams are 15.8 cm apart. For our experiments, we acquired pictures of boxes from distances varying from 1.7 to 3.0 meters to the camera. The background was created using a piece of green cloth and its statistical model was computed from a set of 23 images. Figure 7 shows some examples of boxes used to test our system. Some of these boxes are particularly

Figure 7. Examples of real boxes used for testing.

challenging: (e), (f) and (g) are very bright; (f) and (g) have a reflective plastic finishing; and box (b) is mostly covered with red texture. The dimensions of these boxes vary from 13.8 to 48.2 cm. The intrinsic parameters of the camera (Equation 4) were estimated using a calibrations procedure [2].

The geometry of the box is somewhat different from a parallelepiped due to imperfections introduced during construction and handling. For instance, bent edges, different sizes for two parallel edges of the same face, lack of parallelism between opposing faces, and warped corners are not unlikely to be found in practice. Such inconsistencies lend to errors in the orientation of the silhouette edges, which are cascaded into the computation of the box dimensions.

In order to estimate the inherent inaccuracies of the proposed algorithm, we performed measurements on a wooden box (Figure 7, h) that was carefully constructed to avoid these imperfections. We have also implemented a simulator that performs the same computations on images of synthetic boxes (exact parallelepipeds) generated using computer graphics techniques. Using images generated by the simulator, our system can recover the dimensions of the box with an average relative error of 0.58%. Next, we analyze some of the results obtained on real boxes.

7.1. Statistical Analysis on Real Boxes

In order to evaluate the effectiveness of the proposed approach, we carried out a few statistical experiments. First, we selected several real boxes (Figure 7) and manually measured the dimensions of all their edges with a ruler. Each edge was measured twice, once per shared face of the edge. The eight measurements of each box dimensions were averaged to produce a single value per dimension. All measurements were made in centimeters. We then used our system to collect a total of 30 measurements of each dimension of the same box. For each collected sample, we projected the laser beams on different parts of the box. We used this data to compute the mean, standard deviation and confidence intervals for each of the computed dimensions. The confidence intervals were computed as $CI = \left[\bar{x} - t_\gamma \frac{\sigma}{\sqrt{n}}, \bar{x} + t_\gamma \frac{\sigma}{\sqrt{n}}\right]$, where $\bar{x}$ is the mean, σ is the standard deviation, n is the size of sample and t_γ is a $t-$Student variable with $n-1$ degrees of freedom, such that the probability of a measure x belongs to CI is γ. The tightest the CI, the more precise are the computed values.

Figure 8 shows the computed confidence intervals with $\gamma = 99.5\%$. Note that the values of the actual dimensions (the red line) fall inside most these confidence intervals, indicating accurate measurements. Boxes (f), (g) and especially the wooden box (h) are the ones with tightest confidence intervals. Those are well constructed boxes. Wider confidence intervals were obtained for boxes with bent faces and edges, like boxes (a) and (e). The only box whose actual dimensions do not fall inside the confidence interval is box (d). This box has a cardboard lid that changes the box silhouette, shifting the computed mean values away from the true ones.

Another estimate of the error can be expressed as the relative error $\epsilon = |x - x_v| / x_v$, where x is the computed dimension and x_v is the value of the actual dimension. Figure 9 shows a histogram of the relative errors in the measurements obtained with our scanner prototype for the boxes shown in Figure 7. The higher relative errors were computed for boxes (a), (d) and (e) (the ones exhibiting imperfections) and is in accordance with the experiment summarized in Figure 8. Considering all measurements, the mean relative error for all real boxes is 3.75%, indicating very good accuracy.

7.2. Error Propagation

The error associated to a variable w computed from a set of experimental data can be estimated using the fol-

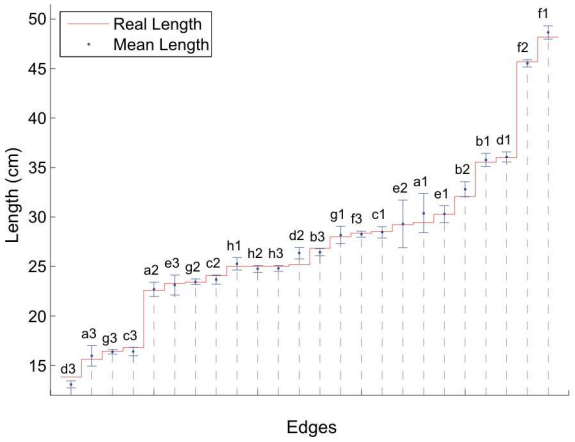

Figure 8. Confidence intervals computed from measurements of the edges of the boxes shown in Figure 7. The boxes edges are sorted by length.

lowing error propagation model [25]:

$$\Lambda_w = \nabla f \, \Lambda_\vartheta \, \nabla f^T \tag{15}$$

where Λ_w is the covariance matrix that models the errors in w; ∇f is the Jacobian matrix for the function $f(\vartheta)$ that computes each term of w from the n input variables; and Λ_ϑ is the covariance matrix that models the errors of the input variables. The model assumes a Gaussian distribution of the errors around the mean values estimated by $f(\vartheta)$ and allows the computation of confidence intervals for the length of each visible edge using a single input image.

To apply this error propagation model, one needs to estimate the error associated to each input variable. In the proposed method, the input variables are:

- $h_i = (\rho_i, \theta_i)^T$, $0 \leq i \leq 5$: the coefficients of the normal equation of the supporting lines for the silhouette edges (12 variables);
- $p_j = \left(x_{p_j}, y_{p_j}\right)^T$, $0 \leq j \leq 1$: the image coordinates of the laser dots (4 variables);
- d_{lb}: the distance between the laser beams (1 variable);
- K: the camera's intrinsic-parameters matrix (5 variables);
- $L = (X_L, Y_L, Z_L)^T$: the laser beam direction (3 variables).

So, Λ_ϑ is a 25×25 covariance matrix, comprised by the variances and covariances of all input variables:

$$\Lambda_\vartheta = diag\left(\Lambda_{h_0}, \ldots, \Lambda_{h_5}, \Lambda_{p_0}, \Lambda_{p_1}, \Lambda_{d_{db}}, \Lambda_K, \Lambda_L\right) \tag{16}$$

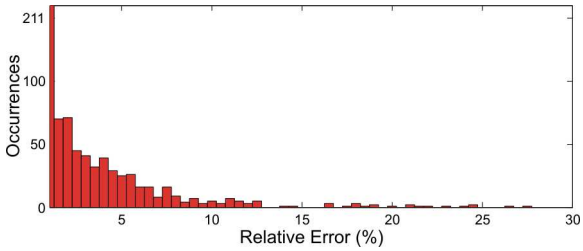

Figure 9. Histogram of the relative errors in the computed dimensions for the edges of the boxes in Figure 7.

Given these variables, the Jacobian of the function that computes the length of the target box edges can be obtained as shown in Equation 17. The partial derivatives in ∇f are calculated using the chain rule over the set of equations presented in Section 3.

$$\nabla f = \left(\frac{\partial w}{\partial \rho_0} \quad \frac{\partial w}{\partial \theta_0} \quad \cdots \quad \frac{\partial w}{\partial X_L} \quad \frac{\partial w}{\partial Y_L} \quad \frac{\partial w}{\partial Z_L} \right) \tag{17}$$

The error propagation model expressed by Equations 15, 16 and 17 allows our system to estimate the uncertainty associated with the measurement of each edge of the box in real time. This information can be used to discard unreliable measurements, which may result if the box is relatively far from the camera, or if one of the box edges approaches a direction almost perpendicular to the cameras image plane.

Special care must be taken when choosing the distance between the laser beams. The uncertainty in the computed dimensions increases as the lasers distance decreases, because the relative error tends to increase as the distance becomes smaller. However, the distance between the laser beams constrains the minimal accepted size for a box, since both laser dots must fall inside the same face. So, for a given application, one should consider a trade-off between the minimal box size and the accepted uncertainty in the measurements.

8. Conclusions and Future Work

We have presented a completely automatic approach for computing the dimensions of boxes from single perspective projection images in real time. The approach uses information extracted from the silhouette of the target box and removes the projective ambiguity with the use of two parallel laser beams. We demonstrated the effectiveness of the proposed techniques by building a prototype of a scanner and using it to compute the dimensions of several real boxes even when the edges of the target box are partially occluded by other objects.

We have also introduced an algorithm for extracting box silhouettes in the presence of partially occluded edges, an efficient voting scheme for grouping approxima-

tely collinear segments using a Hough transform, and a statistical model for background removal that works with a moving camera and under different lighting conditions. We validated the proposed approach performing a statistical analysis over measurements obtained with our scanner prototype from real boxes. In addition, we presented an analytical derivation of uncertainty propagated along the entire computation chain that allows real-time estimation of the error in the computed measurements. The statistics and experimental validation have shown that the proposed approach is accurate and precise.

Our algorithm for computing box dimensions can also be used by applications requiring heterogeneous backgrounds. For that, background detection can be performed using a technique like the one described in [14]. In this case, the camera should remain static while the boxes are moved on some conveyor belt.

We believe that these ideas may lead to optimizations on several procedures that are currently based on manual measurements of box dimensions. We are currently exploring ways of using arbitrary backgrounds with a moving camera.

Acknowledgments

This work was partially sponsored by CNPq - Brasil (Processo No 477344/2003-8), Petrobras (Processo 502009/2003-9), and Microsoft Brazil.

References

[1] P. Besl. *Active Optical Range Imaging Sensors. Advances in Machine Vision*, pages 1–63. Springer-Verlag, New York, NY, USA, 1988.

[2] J. Y. Bouguet. Camera calibration toolbox for matlab. http://www.vision.caltech.edu/ bouguetj/ calib_doc, Jan. 2005.

[3] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, Nov. 1986.

[4] M. B. Clowes. On seeing things. *Artificial Intelligence*, 2:79–116, 1971.

[5] A. Criminisi, I. Reid, and A. Zisserman. Single view metrology. In *Proceedings of the 7th IEEE International Conference on Computer Vision (ICCV-99)*, volume 1, pages 434–441, Kerkyra, Greece, Sept. 1999. IEEE Computer Society.

[6] N. R. Draper and H. Smith. *Applied Regression Analysis*. John Wiley & Sons, New York, 1966.

[7] R. O. Duda and P. E. Hart. Use of the Hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1):11–15, Jan. 1972.

[8] L. A. F. Fernandes. Um método projetivo para cálculo de dimensões de caixas em tempo real. Master's thesis, Universidade Federal do Rio Grande do Sul, Porto Alegre, RS, Brazil, Jan. 2006. (in Portuguese).

[9] L. A. F. Fernandes, M. M. Oliveira, R. da Silva, and G. Crespo. Computing box dimensions from single perspective images in real time. In *Proceedings of XVIII Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI 2005)*, pages 155–162, Natal, RN, Brazil, Oct. 2005. IEEE Computer Society.

[10] F. Figueroa and A. Mahajan. A robust method to determine the coordinates of a wave source for 3-D position sensing. *ASME Journal of Dynamic Systems, Measurements and Control*, 116:505–511, Sept. 1994.

[11] H. Fuchs, Z. M. Kedem, and B. F. Naylor. On visible surface generation by a priori tree structures. In *Proceedings of the 7th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH-80)*, pages 124–133, New Orleans, Louisiana, 1980. ACM Press.

[12] J. Gauch. KUIM, image processing system. http://www.ittc.ku.edu/˜jgauch/research, Jan. 2003.

[13] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, Cambridge, UK, 2000.

[14] T. Horprasert, D. Harwood, and L. S. Davis. A statistical approach for real-time robust background subtraction and shadow detection. In *Proceedings of the 7th IEEE ICCV-99, FRAME-RATE Workshop*, Kerkyra, Greece, Sept. 1999. IEEE Computer Society.

[15] D. A. Huffman. Impossible objects as nonsense sentences. In *Machine Intelligence*, volume 6, pages 295–324. Edinburg University Press, Edinburg, 1971.

[16] A. Laurentini. The visual hull concept for silhouette-based image understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(2):150–162, Feb. 1994.

[17] M. Levoy, K. Pulli, B. Curless, S. Rusinkiewicz, D. Koller, L. Pereira, M. Ginzton, S. Anderson, J. Davis, J. Ginsberg, J. Shade, and D. Fulk. The digital Michelangelo project: 3D scanning of large statues. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH-00)*, pages 131–144, New Orleans, Louisiana, Jul. 2000. ACM Press.

[18] H. C. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293:133–135, Sept. 1981.

[19] D. G. Lowe. Three-dimensional object recognition from single two-dimensional images. *Artificial Intelligence*, 31:355–395, Mar. 1987.

[20] K. Lu. Box dimension finding from a single gray-scale image. Master's thesis, SUNY Stony Brook, New York, 2000.

[21] W. Matusik, C. Buehler, R. Raskar, S. J. Gortler, and L. McMillan. Image-based visual hulls. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH-00)*, pages 369–374, New Orleans, Louisiana, Jul. 2000. ACM Press.

[22] L. Nyland, D. McAllister, V. Popescu, C. McCue, A. Lastra, P. Rademacher, M. Oliveira, G. Bishop, G. Meenakshisundaram, M. Cutts, and H. Fuchs. The impact of dense range data on computer graphics. In *Proceedings of Multi-View Modeling and Analysis Workshop (MVIEW99)*, pages 3–10, Fort Collins, CO, Jun. 1999. IEEE Computer Society.

[23] J. W. H. Tangelder, P. Ermes, G. Vosselman, and F. A. van den Heuvel. Cad-based photogrammetry for reverse engineering of industrial installations. *Computer-Aided Civil and Infrastructure Engineering*, 18:264–274, Jul. 2003.

[24] P. Vlahos. Composite color photography. U.S. Patent 3.158.477, 1964.

[25] J. H. Vuolo. *Fundamentos da Teoria de Erros*. Edgard Blcher, São Paulo, SP, Brazil, 1992. (in Portuguese).

[26] E. W. Weisstein. Bivariate normal distribution. http://mathworld.wolfram.com/BivariateNormalDistribution.html, Oct. 2005.