Estimating the Object Size from Static 2D Image

Ondrej Kainz
Dept. of Computers and Informatics
FEI TU of Kosice
Kosice, Slovak Republic
ondrej.kainz@cnl.sk

František Jakab
Dept. of Computers and Informatics
FEI TU of Kosice
Kosice, Slovak Republic
frantisek.jakab@cnl.sk

Matúš W. Horečný
Dept. of Computers and Informatics
FEI TU of Kosice
Kosice, Slovak Republic
horecny.matus@gmail.com

Dávid Cymbalák
Dept. of Computers and Informatics
FEI TU of Kosice
Kosice, Slovak Republic
david.cymbalak@cnl.sk

***Abstract*— In this paper the real size of the object from a static digital image is estimated. The analysis of selected algorithms used in the area of computer vision is carried out, this includes introduction of the software tools efficient enough to measure the object size. Proposed solution has to be in compliance with the conditions for correct processing of obtained information, such as a suitable scene for measuring and correct adjustment of the camera. The final software solution calculates the estimated size of measured object while utilizing substitution variables that were created for this purpose. Such variables are based on the acquired images and their subsequent processing.**

Keywords— camera; computer vision; object size; size estimation; static image

I. INTRODUCTION

A great number of computer disciplines is on the certain level based on computer vision, e.g. utilization of various optical sensors, way of converting the collected data to be stored in the most efficient way in computer memory or data to be presented accurately and as easy as possible. Every object can be represented as a model in the computer memory. These models have the advantage of containing only the essential information when compared to the real objects. A man naturally sees the world as a set of separate objects forming the overall surroundings. The goal for a computerized device is to recognize individual objects in a picture by its own. As a result, the new approaches and methods emerge to make this process as efficient and quick as possible.

The object detection in an image is not the whole process, equally important task is to use appropriate methods to obtain relevant information about the given object and ignore irrelevant data in the image. One of the basic properties of an object is its size; and the size is the very thing that this paper focuses on while using the image processing algorithms to calculate the dimensions of objects in the image. According to Kopparapu and Desai [1], the process of computer image analysis consists of three different sub-processes. At first, we look for features in the image. Further, these features are analyzed to obtain the information about image. And finally, the only information about individual objects in the image is retained. After these three phases, we have a representation of the image which can be used to recreate the original scene.

Methods used in the image processing differ in way of how they approach the data in the image. According to Sonka et al. [2], such methods may be divided into three categories. Primary focus is on the category of the low-level image processing, which requires only a little information about the content of an image, having as a base unit a pixel. Image compression, image pre-processing sharpening, smoothing, thresholding and so on, fall also into this category.

II. SELECTED TECHNIQUES AND ALGORITHMS IN COMPUTER VISION

Noise is a local feature in image recognizable as an area with high frequency in the signal spectrum of the image. Wojnar [3] and Zhao et al. [4] suggest utilization of the *Fourier transform* as an aid to reduce the noise of a periodic nature. However, this is associated with certain restrictions. In order to remove noise, we can filter out high frequencies and retain only the low frequencies, thus the noise removing filters are often referred to as *low-pass filters*.

The most common technique to distinguish objects from the background in the image is *thresholding*, Slezak et al. [5]. It is the fundamental process of digitization and method of obtaining the binary image with isolated objects, i.e. object segmentation. According to Cheriet et al. [6], thresholding can also be used to distinguish text from the background, e.g. in a greyscale image, this is done by filtering out the areas that are either above or below a threshold with a certain grey value. Thresholding can be used globally or locally. Thresholding itself may introduce the noise. This can be in the form of individual pixels distributed randomly in the image, created during the acquisition of the image, or in the form of groups of pixels, caused by an incorrect threshold value.

978-1-4799-6908-1/15/$31.00

An edge can be real or imaginary and is represented by a line that indicates the outer limits of a surface, Wojnar [3]. Human perception is sensitive to the recognition of object edges. Similarly, in a computer image analysis, the recognition of edges is of one of the basic methods for retrieving the features from the original image. *Corners* are points where at least two edges meet. Corners in the original image have distinct features that clearly distinguish them from the surrounding pixels. There are algorithms for corner detection and tracking that are reliable even after the image is geometrically distorted, Jeong and Moon [7]. Algorithms for object detection frequently utilize the location of corners in the image, prevalent in this category is *Harris corner detector*.

Role of edges play a significant part in recognizing and understanding the scene in the terms of human vision. Images are composed of areas of different size with the same luminance values and color component distribution. Edges mostly occur between two objects, or an object and a background, or two different surfaces. The areas look seamless and can be intuitively identified as separate parts of the image. A curve, ideally a straight line, with a certain orientation that divides two areas is called an *edge*. Precise location of edges may be extracted using derivative function, where the local extrema in this derivation represent locations where values of neighboring pixels vary significantly. From the function can be detected not only the actual extent of neighboring pixels that form edges but also the very direction of the edges, Burger and Burge [8]. Every result of edge detection contains a lot of points that do not represent edges. Bearing this in mind, the other processes are deployed in order to eliminate the points with goal of obtaining the real edges captured in the image.

The most popular edge detector is the *Canny edge detector*, which targets the three basic criteria, i.e. low error rate, good localization of edge points and the elimination of multiple detector responses from the same edge, Tang and Shen [9]. The final phase of the algorithm consists of selection of edge points using a double threshold. According to Rong et al. [10] and Li et al. [11] usage of the double threshold is a way to reduce the impact of noise in the final phase of the identification of edges.

III. Estimating the Size: The Requirements

The principal goal is to propose a solution, which is to utilize computer vision algorithms in the estimation of size of the object from images. Premise to achieve this is to have two types of images – one with a scene without the object and the other one with the object present in the scene. Another premise is to have to information of the camera height.

Following the prior analysis of several frameworks and libraries, specifically Accord.Net, BoofCV, CCV, CImg, OpenCV, SimpleCV, it was determined to utilize OpenCV library. Selected tool is an open-source software, available for Linux operating systems, its principal advantage is an extensive documentation, which makes it easier to understand and properly use its functions and data structures. OpenCV is widely used, resulting in the algorithms verification by a large group of users, further it can be used in several programming languages. Moreover, in OpenCV, the algorithms for real-time object detection, conversion of images to grayscale from different color models, Harris corner detector, and the algorithm for bounding box and more is already to be found. Bearing in mind that the final software solution is be to run on a personal computer device, the number of useful algorithms is more important than the requirements on the resources of a computer.

In this project the 3D object to be measured is represented by a 2D area where the background is not visible because it is being covered by the object. The object being hull or its 2D representation being concave should not affects the obtained dimensions.

IV. Estimation the Size: The Solution

In the previous part was already indicated that the solution is to be consisted of two parts. At first, the user is required to take images and subsequently computer vision techniques are being utilized.

A. Input images

The images of the scene without and with the object are taken. The substantial is the alignment of the object and the camera, since it affects the size the object that we want to measure.

B. Computer vision techniques in the Measurement

The height at which the object appears in the display area of the image depends on the distance of the object from the camera. The real distance of the object from the camera consists of the distance between the object area and the lower border of the display area, and the distance not captured by the camera.

Presumption is to have two images, one with and one without the object. Subtraction using RGB color model is used as the first part of the process. Every pixel of second image is subtracted from a pixel with same position on the first image. This subtraction removes the background (see Fig. 1), as it was identical in both images.

Thresholding, technique frequently used tool to separate the object area and the background, is used to binarize the image. The result is a white area of the object and a black area of the background. In order to retrieve more accurate results following the fuzzy logic, it is suitable to use a smoothing algorithm. A median filter is used as it gives the best performance in processing pictures with black and white areas, Marchand-Maillet and Sharaiha [12]. Canny edge detector (see Fig. 1) is used to find edges after smoothing the picture.

The following step requires to acquire the displayed distance between edges of the object area. These values can be determined if the object is marked by a bounding box and a circle, to obtain a minimal rectangular and circular area which the object occupies. The size of the object will be represented by two numerical values – the width and height of the object area (or one if a circular diameter is chosen). These values will characterize the entire object, thus they represent the maximal horizontal and vertical distances between the edges of the object.

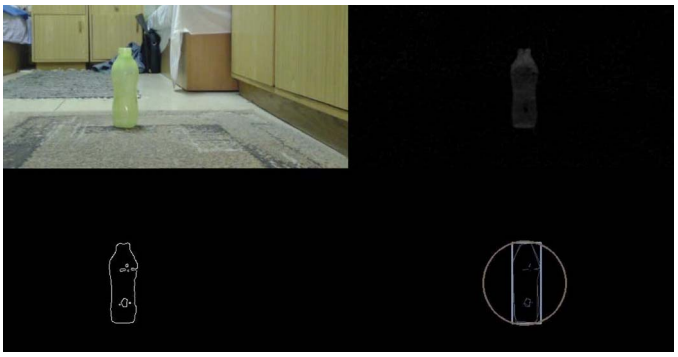

Fig. 1. Image processing: original image (the upper-left corner), removed background (the upper-right corner), Canny filter (the bottom-left corner), boundingbox of the object (the bottom-right corner)

Suitability of the shape of the bounding area is determined by a comparison of the amount of the background it covers. The bounding shape containing less background area is to be chosen. In Fig. 1 is depicted the example of both shapes of bounding areas. The size of the bounding area and a position of the area in the picture are used to calculate the physical dimensions of the measured object. The horizontal location of the object area in display, herein referred to as *yPos* (see Fig. 2), represents the amount of pixels that separate the lower borders of the bounding area from the bottom edge of the display window.

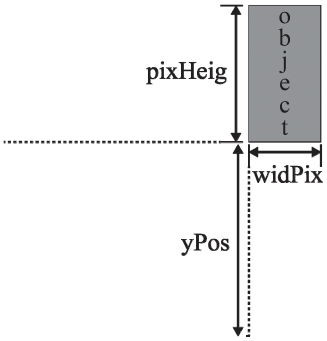

Fig. 2. Position of the object in the displaywindow

Several sets of tests were performed to compile equations. Each set varies in the size of the object or the height from which the camera was shooting the scene. At least four tests with different distances from the camera were taken in each set. The size of the displayed object depends on the ratio of the height from which the camera takes photos and the distance of the object from the camera. It is possible to add advanced settings for more precise definition of the object, e.g. enter more accurate setting of the physical height of a camera that takes photos.

1) Implementing the Software Solution

The user who wants to use the application should have prepared objects to be measured, or may use images of these objects stored in the computer on which the application is running. It is necessary to have a camera plugged in a USB port and the height from which the camera shoots images must be known to create new images. In the case of processing of already stored images, they have to meet the requirements stated in the previous parts. Once the calculations are finished, the estimated size of the measured object is shown and an option to take another measurement is provided.

The inner process of the application consists of three phases. The first phase provides the object detection and a filtering out of the background from an image. The second phase is the edge detection and obtaining pixel dimensions of an object in the image. The last phase is the conversion of the pixel size to the estimated real values, which are output of the application.

C. Estimating the Object's Width

A height from which the camera shoots the scene is referred to as *cam* (see Fig. 3) and the displayed width of the object in pixels is referred to as *widPix*. The *widPix* value represents a linear function expressed in the equation (1). The analysis of the data shows that the gradient - α values and the *Y-intercept* – *β* values match the values of linear functions dependent on the ratio of the value of the cam to the physical width of the object, hereinafter *widReal*. *α* can be obtained from an equation of a linear function (2), *β* can be obtained from a similar equation but with different values of variables (3).

$$widPix = \alpha * yPos + \beta \quad (1)$$

$$\alpha = m_1 * \frac{cam}{wid\,\mathrm{Re}al} - b_1 \quad (2)$$

$$\beta = m_2 * \frac{cam}{wid\,\mathrm{Re}al} - b_2 \quad (3)$$

An equation for estimated width - *estWidth* (4) is obtained after adjustment of the three previous equations. The estimated width should ideally be equal to *widReal*.

$$estWidth = cam * \frac{widPix + b_1 * yPos + b_2}{m_1 * yPos + m_2} \quad (4)$$

D. Estimating the Object's Height

Following the analysis of the tests, if the height of the object measured is the same as *cam* then graphs of the displayed height versus *yPos* have the same gradient as a linear function.

Let us define a reference model as a theoretical model of an object having the height same as the height from which the camera detects the object. The reference model is characterized by its displayed height – *referH* (5).

$$referH = m_1 * yPos + m_2 * cam + b_2 \quad (5)$$

Few relations can be clarified for the graph of displayed height of the reference model versus *yPos*. The slope coefficient is the same as the slope of graphs of objects that meets the conditions of the reference model. The *Y-intercept* depends on the height of real object. This dependence can be

expressed in the equation for the height of the reference model *referH*, where *yPos* is the variable for values of the pixel distance of the displayed object and lower border of display area, *cam* is variable for the height from which the camera captures the scene and m_1 = -1.028, m_2 = 0.06892, b_2 = 203.7 are constants obtained from the analyzed data from the tests. This equation defines the displayed height of the reference model dependent on the height of the camera and *yPos*.

The scheme of the scene with important angles and distances is displayed in the Fig. 3. The real height of the measured object is shown as *heigReal*. The aim is to obtain an estimation of the height as close to that value of *heigReal* as possible. A physical height of a reference model is represented as the distance between points $real_0$ and $real_2$ The distance is the value of *cam*, thus the same as the height of the camera. *heigPix* is the value of the displayed height in pixels of the object and *refer* indicates the displayed height of the reference model, hereinafter referred to as *referHeig*. *Gamma* is a half of vertical angle of view which is in the case of a camera *Logitech HD Pro Webcam C920 44.4*.

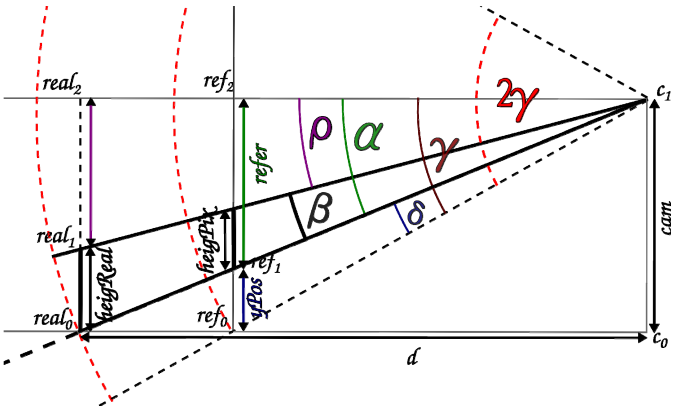

Fig. 3. Important angles and distances

Our goal is to obtain estimated height from the equation (6), which would ideally be equal to *heigReal*. *cam* is the height of the camera when shooting photos, the coefficient *K* with possible values 1 or -1. If the reference height, *refer*, is greater than the displayed height of the object *pixHeig, K* then equals to -1, else *K* is equal to 1. According to the basic trigonometry rules, we obtain the distance between points $real_1$ and $real_2$ from a part of the equation for estimated height. The value of ρ is possible to get from the equation (7). α is a difference between angles γ and δ, (8). δ can be obtained from an equation (9).

$$estHeig = cam + k * \tan(\rho) * d \qquad (6)$$

$$\rho = \frac{|pixHeig - referH|}{referH} * \alpha \qquad (7)$$

$$\alpha = \gamma - \delta \qquad (8)$$

$$\delta = \frac{yPos}{yPos + referHeig} * \gamma \qquad (9)$$

The physical horizontal distance of the measured object and the camera is referred to as *d*, which is obtained from the equation (10). The equation uses the same α values as previous equations.

$$d = \cot(\alpha) * cam \qquad (10)$$

E. Filtering out the background and shadows

The system of object detection offers two separate options for filtering out the background from the image. The first one is a measuring of an object using the RGB model; this results in the object being measured together with its shadow.

The second option is to filter out the shadows thrown by the measured object. After several tests of various color models the CIELab was chosen for this project. It was found that the CIELab color model reduces the effect of shadows of objects in the picture. However note that if the measured object is black, white or in a shades of grey, the CIELab color model appears to be at disadvantage as together with shadows the object itself is being filtered out.

Providing that the photos of an object are taken in an environment where the shadow does not affect the identification of edges of the object, it is appropriate to use the method of measurement without filtering out the shadows. The tests of objects detection were mostly successful. The detection is troublesome when the object consists of a number of areas of significantly different colors. In that case, only one of the color areas of the object is identified as the object.

V. Estimating the Size: Testing

The software solution is set to use the camera with a vertical angle of view 44.4°. The final output was tested using a camera Logitech HD Pro Webcam C920, which meets the given condition for the viewing angle. The output of the applications was recorded in several separate tests with objects of different sizes, at different distances from the camera and camera set to different heights. Randomly selected data are shown in a Tab.1.

TABLE I. Several Tests for Estimation of the Size

camHeight [mm]	Real size width x height [mm x mm]	yPos [pix]	estSize width x height [mm x mm]
154	97x154	61	94x153
154	52x80	2	54x88
129	47x140	73	54x152
80	154x97	22	149x102
350	73x129	102	81x130
275	47x140	3	43x128
140	140x	47	147x149

From the table, the *camHeight* indicates the height from which the camera took photos of an object, the *Real size* is a real dimension of the measured object in the form of entry [*width x height*], *yPos* indicates lower vertical coordinates of the object in the image and the *estSize* is the output of the

application for the test. This output is recorded in the format [*estimated width x estimated height*]. Tests were almost always successful and the dimensions of the object were specified within the error of 10%. In several cases, the deviation was noticeably larger, this was caused by the object being too far for a precise detection of the edges, or in cases when height of the camera was set below 5 cm. The designed software solution takes in account such settings as non-recommended settings and alerts the user of possible variations.

VI. CONSLUSION

In this paper, a solution for estimating the object size from static images using one camera was introduced and related software implementation is comprehensively described. The solution itself is simple, fast and yet easy to implement. The detection of an object in a picture is based on the processing of two images taken in the same environment with and without the object, following the given requirements. The detected object is characterized by the coordinates and the size of the displayed object area. The values obtained from the processing of these images are used as input parameters for the equations to calculate the estimated size. The equations were created according to the basic trigonometric rules and sets of tests are described in this paper. Based on these tests, while using various objects, different heights and different distances of objects from the camera, it was found that the deviation of the measurement is smaller than 10%.

A. Acknowledgment

We support research activities in Slovakia/This project is being co-financed by the European Union. Paper is the result of the Project implementation: University Science Park TECHNICOM for Innovation Applications Supported by Knowledge Technology, ITMS: 26220220182, supported by the Research & Development Operational Programme funded by the ERDF.

B. References

[1] S. K. Kopparapu, and U. B. Desai, Bayesian Approach to Image Interpretation. Norwell, MA: Kluwer Academic Publishers, 2000.

[2] M. Sonka, V. Hlavac, and R. Boyle, Image Processing, Analysis and Machine Vision. Stamford, CT: Thomson Learning, 2007.

[3] L. Wojnar, Image Analysis: Applications in Materials Engineering. Boca Raton, FL: CRC Press, 1999.

[4] L. Zhao, J. Chen, and J. Benesty, "A multichannel widely linearwiener filter for binaural noise reduction in the short-time-fourier-transform domain," IEEE China Summit & International Conference on Signal and Information Processing (ChinaSIP), pp. 227 - 231, July 2014.

[5] D. Slezak, S. K. Pal, B.-H. Kang, J. Gu, H. Kuroda, and T.-h. Kim, Signal Processing, Image Processing and Pattern Recognition. Germany: Springer Science & Business Media, 2009.

[6] M. Cheriet, N. Kharma, Ch.-L. Liu, and Ch. Suen, Character Recognition Systems: A Guide for Students and Practitioners. Hoboken, NJ: John Wiley & Sons, 2007.

[7] K. Jeong, and H. Moon, "Object Detection Using FAST Corner Detector Based on Smartphone Platforms," First ACIS/JNU International Conference on Computers, Networks, Systems and Industrial Engineering (CNSI), pp. 111 - 115, May 2011.

[8] W. Burger, and M. J. Burge, Principles of Digital Image Processing, London, UK: Springer, 2009.

[9] Z. Tang, and D. Shen, "Canny Edge Detection Codec Using VLib on Davinci Series DSP," International Conference on Computer Science & Service System (CSSS), pp. 221 - 224, August 2012.

[10] W. Rong, Z. Li, W. Zhang, and L. Sun, "An improved Canny edge detection algorithm," IEEE International Conference on Mechatronics and Automation (ICMA), pp. 557 - 582, August 2014.

[11] D. Li, P. Qiao, and G. Zhao, "A New Moving Object Detection Approach with Adaptive Double Thresholds," Fifth International Conference on Computational and Information Sciences (ICCIS), pp. 102 – 105, June 2013.

[12] S. Marchand-Maillet, and Y. M. Sharaiha, Binary Digital Image Processing, London, UK: Academic Press, 1999.