Objects recognition on Block World Environment scene: Algorithm of objects detection and classification

Tomasz Grzejszczak, *Silesian University of Technology*
(**07.07.2010**, MSc. Tomasz Grzejszczak, *Silesian University of Technology*)

Abstract

Artykuł opisuje problem rozpoznania sceny z elementarnego świata klocków za pomocą system wizyjnego. Skupia się on na algorytmie wykrywania klocków. Algorytm ten rozróżnia kolorowe obiekty od tła i klasyfikuje je. Ten algorytm i inne opisane tutaj funkcje są częścią działającego systemu wizyjnego używanego do sterowania manipulatorami.

W pierwszej części tego artykułu przedstawione jest stanowisko laboratoryjne do problemu planowania w świecie klocków. Opisane są cele, konstrukcja i ograniczenia. Następnie dogłębnie wyjaśniony jest algorytm wykrywania klocków i klasyfikacji kolorowych obiektów. Ostatnia część przedstawia inne zastosowania tego algorytmu wykorzystane w systemie wizyjnym.

1. Introduction

Elementary Block World (EBW) is an artificial intelligence planning domain. In typical elementary block world problem there is a finite number of blocks, each on the working space. Each block can be either on the table or on another block. The whole scene contains a towers of blocks placed on the table.[1]

An laboratory stand has been created. It contains colour blocks, two cameras, two manipulators and a computer with software for block detecting vision system and manipulator controlling planning program.

This article will focus on presenting the ways of detection, recognition and classification of the blocks. The semantic representation of classification and algorithms of solving the EBW planning problem will not be described in this article, however it can be found in [2][3].

2. Scene construction and equipment

The described laboratory stand is presented on figure 1. The scene consists of a table on witch there are several color blocks. Each block can be put on a table on another block formulating columns. Those columns are placed in a row creating a matrix of all possible positions of blocks. In case of fig.1. there are four blocks, which can produce a 4x4 matrix. The manipulators are placed in such way that they can reach for every block placed in any position of blocks matrix. In order to provide a good recognition of the scene, the cameras are placed from top and from side. This solution provides an opportunity of adding the white background for better blocks detection.

ARM1 is a robot-manipulator with five degrees of freedom and a grabber. In order to grab a block from any direction, all blocks were assumed to be a cylinder shape. The used cameras are Live! Cam Video IM Pro web cameras. Web cameras are cheap, easy to use and provide the sufficient image quality for their purpose. Blocks differ from each another by its unique color.

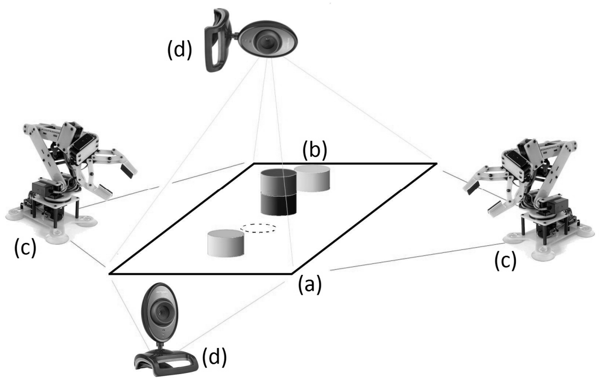

Fig.1. Scene construction. (a) table, (b) blocks, (c) manipulators, (d) cameras.

The vision system computer program was created in C++ with use of OpenCV. The OpenCV is a free open source computer vision library with set of functions and tools for image interpretation. It has wide range of application. Some of most frequently used features are filtering, binarization, image statistics, calibration techniques, detection, tracking, shape analysis, segmentation and recognition.

All the functions and algorithms in OpenCV are optimized, which guarantees high performance and quality. The functions are written using dynamic data structures, ensuring flexibility of application.[4]

3. Block detection algorithm

The aim of the vision system is to provide the feedback information about the state of the scene for control program. The blocks position matrix containing a state of a scene is concluded using information from top and side camera. Those data is called a vector of blocks seen from top or side. The top and side vector can formulate a blocks position matrix using reasoning algorithm. An example of scene, position matrix and position vectors is shown in figure 2.

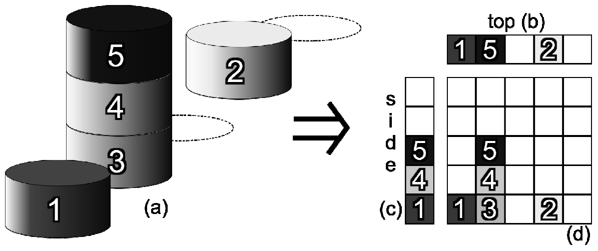

Fig.2. Example of (a) scene with 5 blocks and its interpretation with use of (b) top position vector, (c) side position vector and concluded (d) position matrix.

In order to get a position vector an recognition algorithm needs to be performed. The algorithm can be presented in following points:

1. Prepare image. Grab a frame from camera, convert it to HSV color space.
2. Threshold image in order to cut the background and obtain a mask of objects (blocks).
3. Calculate color histogram in order to know the color distribution on the image.
4. Find most popular color and get its position on image.
5. Subtract detected area from mask, and find another color from new mask (perform step 3-5 until the most popular color area is below certain level)
6. Sort all gathered data and create a position vector from all detected blocks.

All of the above algorithm points will be described in the next subchapters.

3.1 Image preparation

The first point of any vision system is to grab a frame from a vision device. In this case it were two USB web cameras. With use of C++ and OpenCV, created program was able to constantly grab frames from two devices and process them. Each frame is a RGB image, which is described by three monochromatic maps presenting intercity of red, green and blue color. However the scene consist of color blocks on white background, thus another color model needs to be introduced.

HSV is a model based on human vision. Its three components describe the hue, saturation and light intensity called value. This color space has been chosen because it allows an easy background cutting by thresholding low saturated regions of image and easy block recognition using hue value.[5]

In conclusion, after this point, a computer program has two images from two cameras, each converted to HSV and ready to process.

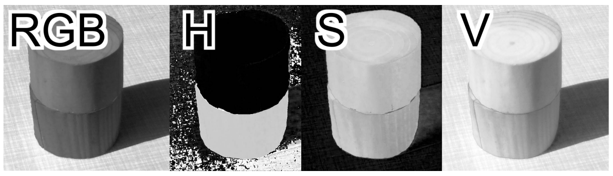

Fig.3. Example of frame and its transformation into HSV color space.

3.2 Thresholding

In order to know where is a useless background and where are the blocks that we would like to detect, a background cutting needs to be performed. It is assumed that the background is white. Even if there is a shadow that would change the light intensity value, its saturation should not change. Thus a threshold can be assigned in order to determine the saturation value under which a pixel is considered as background and under which its considered as block. The result of thresholding is a binary mask in form of map of pixels with the same dimensions as the original image. After multiplying the mask and image, the result is a new image with cut background and with blocks only.

The mask do not uniformly distinguish regions. There are some lonely pixels that was assigned improperly. In order to get rid of them, it is convenient to use morphological functions.

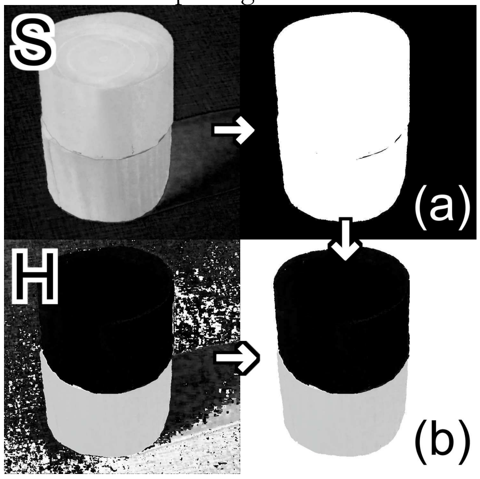

Fig.4. Thresholding image using saturation. (a) The obtained mask after threshold, (b) region of interest after applying mask on hue map.

3.3 Calculating color histogram

With mask telling the region of block, the next step is to obtain the mask of only one block. The first step is to calculate a color histogram. The hue value from region pointed by first mask is gathered and presented in form of histogram. The highest pitch of histogram means that certain color is on the highest number of pixels, thus it is assumed that a block with this color is the biggest.

Again, a shadow from another block, or from cylindrical shape of block should only affect light value, instead of hue. However, some bounds of error are assumed, thus after detecting a color of exemplary value 44 from range 0-180, the second mask of this block should be thresholded from a range. For example from value 41-47.

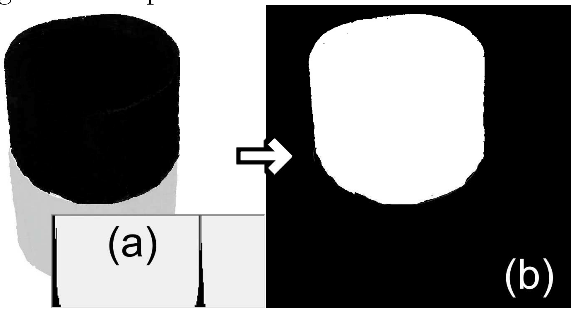

Fig.5. (a) Color histogram of threshold hue image with two colors and dominant red (value near 0). (b) Calculated mask for red color.

3.4 Obtaining position of detected color

The second mask describes the region of one block. In this point the block would be identified and it will be assigned to a position in detection vector. In order to perform the assignment, a series of OpenCV functions are performed. The most crucial one is called camshift.

Intel describes camshift as an algorithm that for each frame captured from video device converts it into probability distribution image using hue. It is obtained from color histogram. For declared color, CamShift finds regions of this color, and enclose it in rectangle. The output is this rectangle with calculated size, position and orientation. Also this calculated data is passed again to CamShift as an initial position for finding objects on next frame. CamShift is an extension of Mean Shift algorithm.[4]

Presented vision system works in similar way. Here a frame consist of few blocks, each with different color, and all of those blocks are needed to be found. However it is not necessary to constantly track the objects, thus previously calculated positions are not helpful to find blocks on the next frame. The block detecting vision system uses this function only to calculate the center of object.

After this point an information about block color and position are stored and are waiting for further calculations.

3.5 Subtracting color from mask

When the block position is calculated, it is time to detect another block. This lead to repeating the algorithm from point 3. However without changing the first mask, the block just detected would be detected again. It is necessary to modify the first mask by subtracting from it the second mask. This operation result in considering just detected block as a part of background. After performing step 3 again, the most popular color from image would not be considered, leaving an opportunity for another color to be detected.

When a color is detected, and a second mask with block is formed, all its pixels are summed, and an area of block is obtained. If this area is too small, it means that all blocks has been detected, and no more blocks are on the scene. The object with too small area that is already detected is not considered to be a block and this leads to a final step of an algorithm.

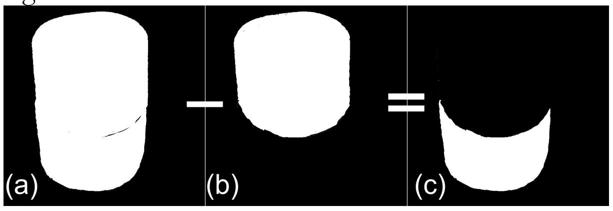

Fig.6. (a) First mask after threshold, (b) second mask of block, (c) resulting mask with undetected blocks.

3.6 Creating position vector

After all blocks has been detected, program analyzes all data gathered about blocks colors and positions and calculates the side position vector.

For side vector, all positions are sorted. For example if a red block has been detected with lower vertical coordinate value then blue block, it means that red block lies on blue. The block colors are compared to defined colors obtained from calibration. If an exemplary block with hue value 44 would be detected, and in a database there are two blocks defined: 1) 40, 2) 58, then our block would be recognized as block 1.

Side vector is detected similarly, however there are possibilities of empty places (see fig. 2., top vector, position 3, 5) In this cases detected blocks needs to be assigned into defined from calibration positions.

When the side and top vector are obtained, the final step is to conclude the blocks position matrix. This step is performed using reasoning algorithm described in [3].

4. Other functions of vision system

Obtaining the blocks position matrix was the main task of the vision system, however there were two other useful functions, which were implemented using the similar algorithm. Those functions were camera distinguish function and a calibration function.

4.1 Distinguish function

After connecting the USB cameras, operating system automatically gives a number to a connected device. However it is unknown which camera (side or top) has been connected first. This is the reason for implementing a camera distinguish function. The aim of this function is to determine which image stream from a camera suits the expectation of top and side camera.

First two steps of main recognition algorithm are used in the same way, however a block mask is then used to calculate the contours of objects. The contours of objects is used to calculate area and perimeter, and then to calculate the circularity shape factor. It is assumed that top camera would see some cylindrical blocks from top, that are projected as circles. On the other hand, side camera would see the piles of blocks projected as rectangles. The shape factors are calculated and the camera with image containing more round objects is assumed to be a top camera.

4.2 Calibrate function

The main recognition algorithm is searching for blocks, however without a previous declarations, it would not know what to search for. It is necessary to declare the initial conditions and searched blocks properties. The function is divided into three steps.

First, the user is asked to properly adjusted the cameras in order to see the whole scene clearly. The region of interest is selected from the camera field of view.

Next, user is asked to put all blocks on the scene, that all of them would be seen from top camera. In other words, the position matrix should contain all blocks in the last, bottom row. User is able to adjust the saturation level for thresholding.

Final step is performed automatically. All blocks from top camera are detected with the same algorithm as presented in point 3, however the last point of algorithm do not uses gathered data to conclude the top position vector, but to get information about the blocks colors and their positions.

5. Conclusion

The presented algorithm has been implemented in vision system. Despite its limitations it is working as expected and it is providing good feedback for manipulator control program. It has been tested in various cases. The algorithm can be changed in order to detect other types of object that are distinguishable by color. Its output data can be used as a base for more complicated algorithms. In the presented application the algorithm is used as a base for blocks reasoning using incomplete information

Bibliography (Style Chapter)

[1] Slaney J., Thiébaux S., *Blocks World Revisited*, Canberra, Australia, 14 June 2000

[2] Grzejszczak T., *Semantic representation of simple Block World environment in application to ARM1 robot arm control*, Gliwice: master's thesis, Dept. Automatic Control, Electronics and Computer Science, Silesian University of Technology, 2010.

[3] Grzejszczak T. *Semantic representation of Block World Environment: algorithm of scene reasoning from incomplete information,* Przegląd Elektrotechniczny, Polska 2011.

[4] Intel Corporation, *Open Source Computer Vision Library Reference Manual*, USA 2001

[5] Bovic A., *Handbook of Image and Video Processing Second Edition*, USA 2005

Authors:

MSc. Tomasz Grzejszczak
Silesian University of Technology
44-100 Gliwice
tel. (+48 32) 237 21 76
fax. (+48 32) 237 11 65

email: *tomasz.grzejszczak@polsl.pl*