Usage of the webcam as 3D input device

Pavel Vlašánek
Supervised by: Alexej Kolcun

Faculty of Science
University of Ostrava
Ostrava / Czech Republic

Abstract

The paper is focused on using the webcam as a 3D input device. Generally it means the user may use any object and shadow of this object to obtain the coordinates x, y and z (position in the space). This idea allows to use a navigation in three dimensions without an expensive or atypical hardware. The system needs one ordinary webcam, an ordinary computer and a source of suitable light.

The goal of this paper is to describe an implementation of a software usable for handling an input from webcam with conditions described previously.

Keywords: navigation, shadow detection, foreground separation, webcam

1 Introduction

We know many devices that can be used for communication with a computer. A keyboard and a mouse are common but nowadays they may not be sufficient and comfortable. The reason is simple, each of these devices is a piece of hardware not allowing modifications. You can neither add the additional buttons into the mouse nor the new keys into the keyboard. The modern way for communication with a computer is the input from webcam. The user does not need a special controller very often and that is a big advantage. Good examples are Microsoft project Natal [1] and project Cam Space [2]. In the case that we do not need to use a depth we can use a system for a hand gestures recognition [3].

Instead of webcam and computer vision the special hardware for 3D navigation may be used. The way to control the computer by using a data glove and gestures [4] or multimodal input [5] is very natural. We can also use a special hardware for virtual finger tap. The company NOVIA AG [6] offers touchscreens with this functionality.

Basically, role of commonly used devices is allowing to work with the computer in the most natural way. The mouse is a good example because of using it as a hand. The operating system obtains position of the mouse and state of the buttons. We can say the mouse is 2D input device but via click we can provide something like a touch. The touch may be interpreted as a level of depth.

The main idea of this paper is shown in Figure 1. A finger works as a mouse (has a position) but we can also obtain better information about depth instead of a simple click.

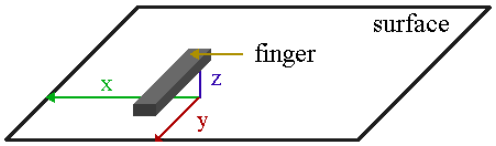


Figure 1: Image of the real scene

The parameters of a finger (or any another object) are a position (x and y) and the distance from the surface. The distance will be proxy for z coordinate allowing to work with the depth.

2 Previous works

The system needs to capture an image, detects an object and the shadow and then computes the coordinates dependently on mutual positions of those. I have studied lots of materials regarding object/shadow detection and chose a version via color (instead of e.g. contours). The first idea was to use HSV colour space because this space is more suitable for shadow detection [7]. At least RGB colour space was selected because we do not need to have a whole compact image of the object/shadow. The second reason was that surface colour(s) should be a very different comparing to object/shadow. The third reason was faster computing.

Detection based on the contours was explored too. I have experimented with Canny edge detector [8] but results were indicating another way of thinking. The solution based on contour detection instead of colour detection will be deeply investigated lately and is it a part of the future work.

The next problem was an foreground image. Colour values were very unstable and very often there was an edge between the object and the shadow practically invisible. The reason was that thresholds levels were very hardly setup. The pixels from object had a same value as the pix-

els of the shadow. This situation may causes some trouble, because shadow/object image may be very unclear. The first idea for improvement was to use an erode and dilation algorithms [9]. The results were not good enough so I have tried another idea. The meanshift image segmentation seems to be a better solution and it will be presented in chapter 4.2.

3 Hardware setup

There is the input image in Figure 2. The system needs an object, a surface, a source of the light and a webcam. From algorithmic point of view the system needs a mechanism for the foreground separation with some kind of smoothing, and for the shadow and the object recognition.

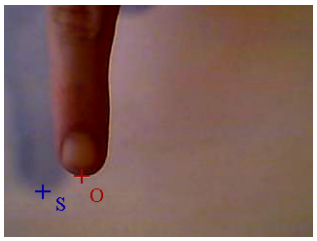


Figure 2: o point is a the end of finger, s point of the shadow

While the finger is falling down/raising up s point is coming closer/further to the o point. From these facts we can assume formula

$$h_l = |O_x - S_x|$$

where O_x is the x position of the finger and S_x is the x position of the thrown shadow. The formula above is valid (and we can assume that h_l is proportional to the z coordinate) for the scene with one shadow (or the strongest one among more shadows) coming closer/further to the object from the side. Practically the horizontal distance between points must be dependent on the real distance between object and surface. The relationship between the result h_l and z coordinate depends on the lighting. The comparison is shown in Figure 3.

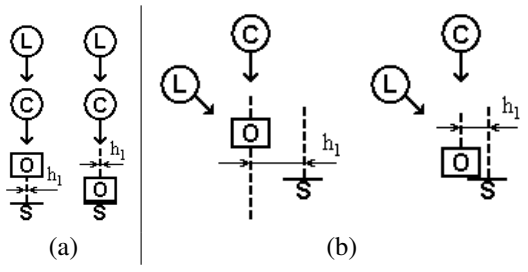


Figure 3: light L, camera C, object O and shadow S

If the shadow lies exactly under the object distance h_l is not affected by the distance between the object and surface. In Figure 3 (a) the distance h_l is roughly constant but in the second image (b) it depends on the height of the object over the surface. You can see that important thing is the mutual position of the camera and the light. There is shown a whole scene in Figure 4. Our goal is to express dependency between the height h and the distance between the object and the shadow as

$$\xi = \xi_1 + \xi_2$$

We can assume from triangle similarity that

$$\xi_1 = \frac{hx}{H-h}$$

$$\xi_2 = \frac{h(\Delta - x)}{H-h}$$

From these formulas we can assume the final formula

$$h = \frac{H\xi}{\Delta + \xi}$$

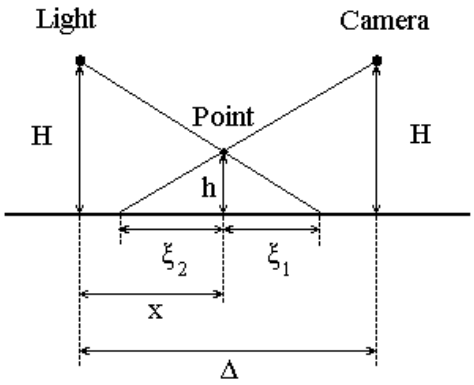


Figure 4: The exact view of the scene from up

There is dependency graph between h and Δ in Figure 5. There is shown an effects H to Δ. The outermost curve is for $\Delta = H/4$ and the lowest for $\Delta = 2H$. The increased Δ approaching a linear relationship thus linear formula above may be used instead of linear fractional. The linear formula is much simpler and faster and for more correct behavior variable h_l should be used as

$$h = h_l c$$

where c is the calibration constant[1].

4 Image processing

We briefly present the techniques used in the paper in this section. The output of these techniques will be more suitable image for processing than the original input from the webcam.

The system works as it is shown in the pseudo code below:

[1] A value depends on difference between the real height h and the measured height h_l.

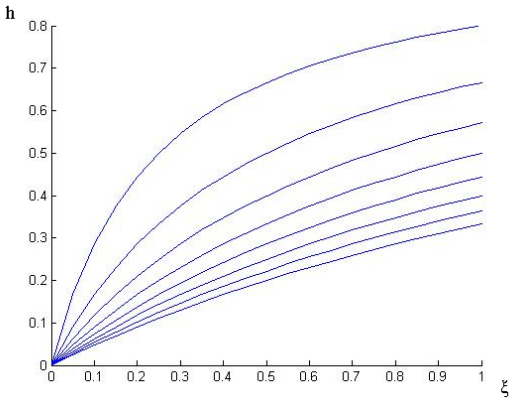


Figure 5: Dependency graph

```
determine_background();

while (!end())
{
    determine_foreground();
    smooth_foreground();
    create_images_IoIs();
    determine_the_points_OS();
    compute_coordinates();
    show_results();
}
```

The while cycle is repeated till the user stops the program.

4.1 The foreground separation

We are using an averaging background method which is suitable for static background and for scenes with the fairly constant lighting [10]. The average and averaged difference of each pixel are used in this method. As the first step the user has to provide a couple of images of the background. Then we can compute an average as

$$I_{avg} = \frac{\sum_{i=0}^{n} I_i}{n}$$

and the averaged difference as

$$I_{diff} = \frac{\sum_{i=1}^{n} |I_i - I_{i-1}|}{n}$$

where n is a number of the background images and I is the background image. Now we can create a model which will be used for the foreground detection. We will use the two thresholds defined as

$$I_{low} = I_{avg} - I_{diff} s_l$$

$$I_{high} = I_{avg} + I_{diff} s_h$$

where I_{low} is the lower threshold, I_{high} the higher one and s is for determine a range[2].

[2] $s_l = 6$, $s_h = 7$ according to [10].

After this analysis we can already process a new image from the webcam containing a foreground. Each of the pixels of I_{new} covered by

$$I_{low}[x,y] < I_{new}[x,y] < I_{high}[x,y]$$

is recognized as a foreground.

4.2 The meanshift image segmentation

The technique was used to simplify a foreground image. The pixels of the shadow have very similar values and we may suppose the same for the pixels of the object[3]. Situation may be different in the captured image because of the noise. Therefore the usage of the meanshift image segmentation is very suitable because it smooths colour variations [11].

The algorithm is based on meanshift clustering over the colour [10]. A meanshift window passes through the space and finds the groups with the highest density of data. Each point which is converging at a peak becomes connected by the peak. This ownership forms the segmentation of the image [10]. The Figure 6 shows a comparison of an intensity of the row of the image without this segmentation and with the segmentation.

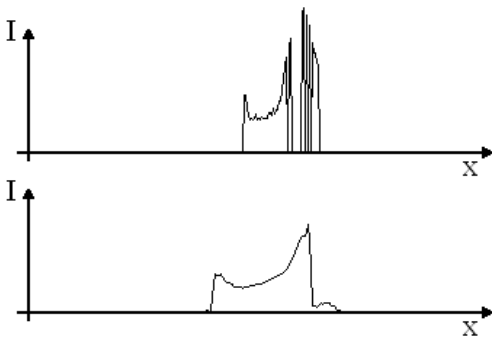


Figure 6: The intensity of the same line without meanshift segmentation (up) and with the segmentation (down)

Clusters with similar colour are replaced by the one colour. The shadow and the object are more recognizable and some noise is also eliminated.

4.3 The object detection and the shadow detection

Colours of the object (finger) and of the shadow have to be different than the background colour. The prerequisite is very important because of RGB colour space[4].

The finger has a value of the red channel higher than value of the green and the blue channel. All three values of the shadow are very similar. It seems that the blue channel

[3] Object with small colour variations is highly recommended.

[4] Segmentation is based on colours so shadow and object must be visible.

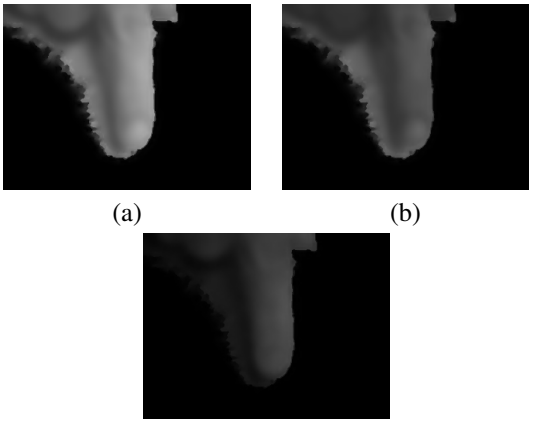
is not important so only red and green channels will be used.

$$I = \{I_r, I_g, I_b\}$$

The new image I_p is defined as follows

$$I_p = I_r - I_g$$

where I_r and I_g are channels from the image I mentioned above. The images related to this formula are in Figure 7[5] and you can see that each pixel in the shadow has a very similar colour and each pixel in the finger too.



(a) (b)

(c)

Figure 7: (a) I_r (the red channel of the foreground); (b) I_g (the green channel of the foreground); (c) I_p (difference between red and green channel)

In the chapter 4.4 there is explained how we can obtain an images of the object I_o and of the shadow I_s. The image I_s contains the white pixels represent a shadow and black pixels represent the rest. Similarly the white pixels in an image I_o represent an object and black pixels represent the rest. The points o and s from Figure 2 are the first white points in images I_s and I_o from right bottom.

In comparison with [3] our system does not need to know the contours. The main points are obtained from similar pixels recognized as shadow and from pixels recognized as object.

The user have to observed the rule about the object position. The system is depend on the pivot points thus the object have to be positioned as is shown in Figure 2. It means that the top of the object must come closer/further to the top part of the captured area.

4.4 Calibration

A suitable lighting source is very important prerequisite for the correct behaviour of the system. In other words there have to be a clearly visible shadow and an object (finger) in the image I_p as it is shown in Figure 7. In other words, after $I_r - I_g$ shadow and object must be visible. From observations we can say that floodlight is more appropriate than a sunlight.

The requirements for the calibration are very simple. We have to set thresholds of the finger and of the shadow. It will be used for separation from the foreground. Only four variables below have to be set:

- shadow low threshold T_{sl}
- shadow high threshold T_{sh}
- object low threshold T_{ol}
- object high threshold T_{oh}

The low/high thresholds mean the highest/lowest values of the pixels in the shadow or the object part. The range between the T_{sl} and the T_{sh} must be different compared to the range between the T_{ol} and the T_{oh}. We can obtain the images of the object I_o and of the shadow I_s as follows

$$I_o[x,y] = \begin{cases} 255 & T_{lo} < I_p[x,y] < T_{ho} \\ 0 & otherwise \end{cases}$$

$$I_s[x,y] = \begin{cases} 255 & T_{ls} < I_p[x,y] < T_{hs} \\ 0 & otherwise \end{cases}$$

In Figure 8 there is shown one row in I_p image. You can see the edge between the shadow and the object as rapid change in intensity. The pixels with lower intensities will be recognized as shadow where exact bounds are chosen via T_{sl} and T_{sh}. Similarly, the object will be represented by higher intensities defined by T_{ol} and T_{oh} variables.

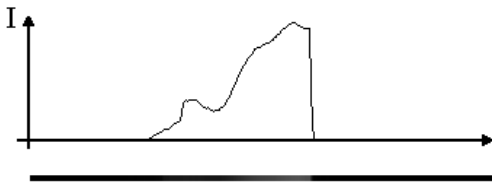


Figure 8: The intensity of the line of the I_p image.

5 Application

We are presenting the possible usage in this section. The application is written in C++ for graphic routines and obtaining images from a webcam an OpenCV library [12] was used. The user needs a computer, one ordinary webcam, a paper and a pencil.

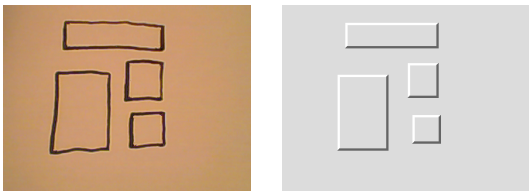
We have chosen a virtual keyboard for demonstration of the whole idea. The application provides handling of the key drawn by hand. The user may tap on the paper and virtually press the key and use it for comunication with the computer. The user may draw any number of the keys exactly how many really needs. For example the user will

[5] These images were captured with indoor lighting.

need a controller for any car game thus draw only key for acceleration, key for break and keys for turning. This solution has a big advantage because the user may choose count and position. The user may draw also a special controller as piano keyboard and on the touch react by playing a sound.

5.1 The keyboard detection

The user may draw any number of the keys as the arbitrary quadrangles. There are shown sketches and recognized keys in Figure 9.



(a) (b)

Figure 9: (a) Image of the four hand drawn keys; (b) Keys have been found and drawn as the rectangles

The recognition is based on the corners. We need only the strongest corners because there may occur a roughness on the line of the hand drawn key. At first we need to calculate minimal eigenvalue of gradient matrices using the Sobel operators (for more information [13], [14]). The results are stored to the I_{src}. Then non-maxima suppression is computed as follows

$$I_{dst}[x,y] = \begin{cases} I_{src}[x,y] & I_{src}[x,y] > mq \\ 0 & I_{src}[x,y] \leq mq \end{cases}$$

where I_{src} is a source image, I_{dst} is a destination image, m is the max value of the intensity obtained from I_{src} and q is quality level[6].

We have the corners as a result of the method above. The recognition starts with the left up point. Then we find the nearest point in the x-direction and the y-direction. The fourth point is the nearest point in the x-direction from the third point.

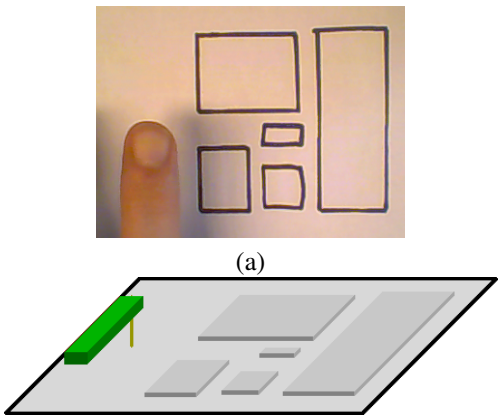
5.2 Graphic output

There was added an isometric view for better navigation in the 3D space. Figure 10 shows 2D input image captured from the webcam. It is drawn as an isometric image with some level of depth.

6 Results

We are able to provide the navigation in the 3D space using one webcam. The navigation may be controlled via

[6] $q = 0.2$ according to [10].



(a)

(b)

Figure 10: (a) The original image; (b) Isometric view drawn

object and the shadow of this object. The user has to show a background using a source of the light for obtaining a clearly visible image of the foreground composed from the object and the shadow. The places with very similar gray colours (according to Figure 7 (c)) are recognized as the object and the other one as the shadow using the thresholds. Middle top points of these places are chosen as the pivot points. The distance between the points is used as the level of depth and the position of the object point as x and y coordinates in Cartesian xyz-system. The coordinate z may be computed via linear or linear fractional depends on the position of the light and webcam. The computing of these coordinates works fast enough and it is usable for the usage in the real time.

The system has been tested in home conditions. It means that distances H and Δ from Figure 4 has been in tens of centimeters. Additionally the distance Δ was much longer than H thus linear formula was used. If the user set the suitable c then value of the h corresponds with real distance between the object and the surface.

7 Conclusions and future work

Functionality of the system was shown on virtual keyboard example where the level of the depth (z coordinate) is used for click detection. The virtual click is only one aspect of the possible usage and it has been chosen because of not very demanding dependencies on lightness and calibration. We can expand the idea of the virtual click using the projector showing an image on the wall. According to Figure 4 the projector is the *Light*. From the foreground image obtained as described above a hand and the main finger are separated. The user may tap on the wall and press the button shown by the projector. Then a computer obtains a press event and handles it by a particular way.

As mentioned above a virtual click is only one possible usage. In general the system may be used as a navigation in the 3D space using an object as the controller. The con-

troller may be used for 3D modeling and via gesture we can create and grab the virtual points and than make the models, move models, and so on.

We can also use a simple object as a virtual pencil and draw via touching the wall. The webcam obtains an image. On the places where the pencil is touching the wall the program draws the points. This idea may be expand by third dimension. The distance between object and the surface will be used for computing of the z coordinate as is shown in chapter 1. The user have to set an specific event to begin drawing and event for stop. We have the webcam and the system for the computer vision thus usage of gesture is very suitable [3]. For example the user may join index finger and thumb and than via the object (or the finger) draw in space. The system will be interpreted as path of the object as colour track and then when the user makes a stop gesture drawing will be stopped.

The work on this project is still ongoing and these ideas presented in this chapter will be the main targets in the future.

References

[1] "Project natal." `http://www.xbox.com/en-us/live/projectnatal/`. visited: 28.3.2010.

[2] "Camspace." `http://www.camspace.com/`. visited: 28.3.2010.

[3] J. Cíger and J. Plaček, "The hand as an ultimate tool," *SCCG*, 2000.

[4] E. Sánchez-Nielsen, L. Antón-Canalís, and M. Hernández-Tejera, "Hand gesture recognition for human-machine interaction," *WSCG*, vol. 12, 2004.

[5] J. García, J. P. Molina, D. Martínez, A. S. García, P. González, and J. Vanderdonckt, "Prototyping and evaluating glove-based multimodal interfaces," *Journal on Multimodal User Interfaces*, vol. 2, 2008.

[6] "Interactive devices." `http://www.novia.ch`. visited: 28.3.2010.

[7] "Shadow detection." `http://dali.mty.itesm.mx/~autonomos/Navdyn/node11.html`. visited: 28.3.2010.

[8] J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 8, no. 6, pp. 679–698, 1986.

[9] J. R. Parker, *Algorithms for image processing and computer vision*. John Wiley & Sons, 1997.

[10] G. Bradski and A. Kaehler, *Learning OpenCV*. Gravenstein Highway North: O'Reilly, 2008.

[11] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, pp. 603–619, 2002.

[12] "Open source computer vision." `http://opencv.willowgarage.com/wiki/`. visited: 28.3.2010.

[13] M. Nixon and A. Aguado, *Feature Extraction & Image Processing*. Linacre House: Elsevier, 2008.

[14] "Opencv feature detection." `http://opencv.willowgarage.com/documentation/cpp/feature_detection.html`. visited: 28.3.2010.