Annales UMCS Informatica AI X, 2 (2010) 63-68
DOI: 10.2478/v10065-010-0053-5

Annales UMCS
Informatica
Lublin-Polonia
Sectio AI
http://www.annales.umcs.lublin.pl/

Modelling 3D scene based on rapid face tracking and objects recognition

Krzysztof Dmitruk, Grzegorz M. Wójcik

Institute of Computer Science, Maria Curie-Sklodowska University,
pl. M. Curie-Sklodowskiej 5, 20-031 Lublin, Poland.

Abstract – Mixed reality techniques are presented. Implementation and application of CAMSHIFT algorithm is discussed to some extent. In the initial stage of the research the technology of real object's edge detection and geometrical figures' representation in the virtual scene has been worked out.

1 Introduction

Despite undisputed domination of traditional input devices like keyboard and mouse, there are many other ways for providing data by the computer user. Alternative controllers may be popular in the future, but even nowadays we perceive their extensive application. Medicine uses cameras and brain activity analysers to provide paralysed people opportunity to contact with others by respectively moving eyeball, blinking or thinking. Cameras and accelerometers are used for collecting data about human position, movement and other actions depending on situation. This, with the addition of realistic graphics presentation like head-up displays and binaural sound forms structure called mixed reality. This term is wide and can be referred to small control icons on digital camera viewer as well as sophisticated environment of flight simulator with multiple display panels and realistic force feedback. Nowadays mixed reality elements are going into mainstream of commercial market. The Nintendo Wii controller using the infra-red camera and accelerometers or binocular monitors are the examples.

The work presented herein concerns creation of simple mixed reality – virtual 3D scene with the models of really existing flat shapes, which rapidly adapts its virtual camera to the position of observer's eyes. The main objective of the research was to show how accurate and realistic results can be achieved without specialised sensors, but with low-end web camera used for face tracking.

2 Algorithms

We are using two independet algorithms. First for face tracking, which is operating in real time of program activity analysig every single frame. Second, for objects detection is used once for every loaded file with objects' photo.

2.1 Face tracking algorithm

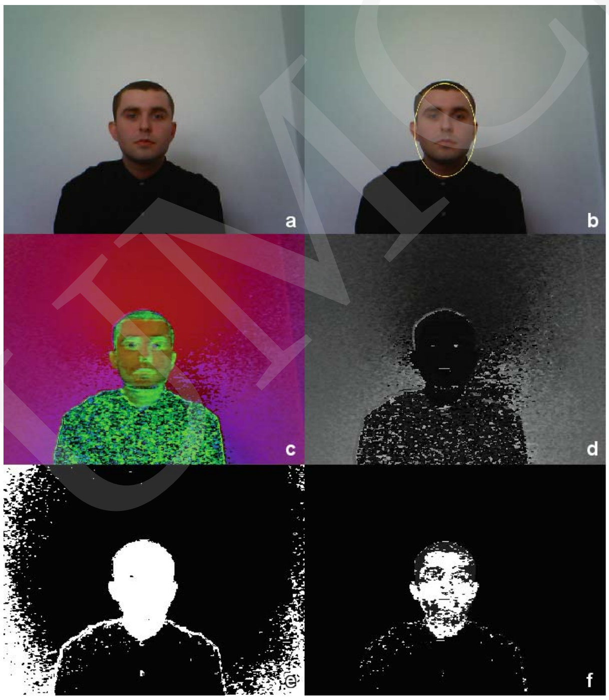

Fig. 1. Objects' recognition process images: (a) captured frame, (b) captured frame with the ellipse circumscribed on matching centroid, (c) HSV channels presented as RGB, (d) H (hue) channel, (e) mask of points with matching possibility different from zero (f) probability distribution.

Continuously Adaptive Mean Shift (CAMSHIFT) algorithm proposed and elaborated by Gary Bradski [1] was used. Here will be described only the essentials, necessary to understand how application works. CAMSHIFT is derived from the mean shift algorithm designed to work with video sequences. It is able to register four degrees of freedom: movement in x, y, z vectors and rotation in x, y plane. Tracking mechanism is based on colour probability distribution. In order to separate colour from the other information we used the hue channel from the HSV model. Before start of tracking, the user should select fragment of image, that will be used as the reference. The algorithm gathers information about all pixels' hue (discrete 0 – 255) values and fills up the table which contains the number of pixels with this hue that has been found in the area. We will define the mean point as p_c. The first one can be chosen from any point inside the selected area. Next points are computed according to equation (1):

$$p'_c = \frac{\sum_{i=x_p}^{x_k} \sum_{j=y_p}^{y_k} I(p_{ij}) p_{ij}}{\sum_{i=x_p}^{x_k} \sum_{j=y_p}^{y_k} I(p_{ij})}, \tag{1}$$

where (x_p, y_p) – the search area upper left edge coordinates, (x_k, y_k) – the search area lower right edge coordinates,$I(p_{ij})$ – the probability value of window with (i, j) as the mean point, p_c – the current mean point.

The obtained p'_c coordinates should be compared with the last p_c point. If their distance is smaller than the constant threshold (in our implementation it is 1) we can accept it as a new mean point.

2.2 Objects recognition

Models of really existing objects can also be created. Although the program works properly with flat, convex and solid coloured figures (which is caused by displaying code imperfection) the recognition algorithm is able to handle sunken and multicoloured structures and in most cases, they will be properly or partially properly shown. As the input the set of two images was used. The same plane with and without the object used as background was present in the images. The first step is to create a boolean mask that shows differences between these two images. They are shown in Fig. 2 (a) and (b).

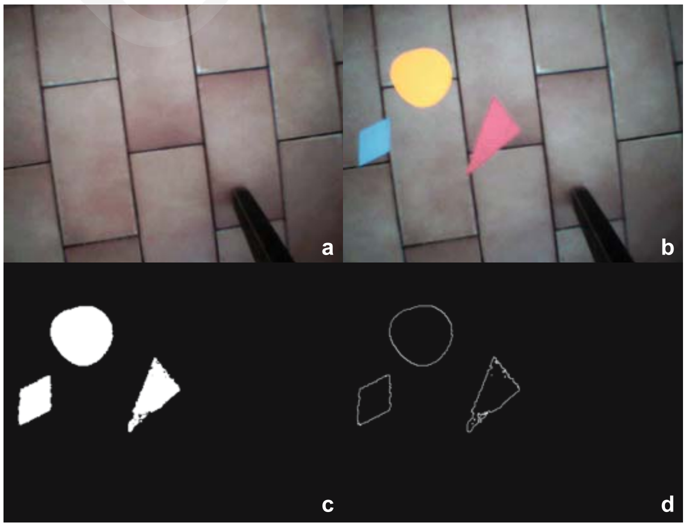

Fig. 2. Objects' recognition process images: (a) background, (b) objects, (c) difference mask, (d) border mask.

The program compares each pixel in both images matched by coordinates. If they have different colour, the coordinates are saved to the mask. Unfortunately, unlike computer generated images, shapes created with the camera often have some amount of noise. Another problem we encountered, was automatic histogram equalization present in all cameras that were used, causing minor differences between two images, where the same colour should be present. To cope with that, the algorithm has similarity threshold set to 40 for each of RGB values. Fig. 2 (c) shows the difference of mask created from the images obtained by the camera. While blue and orange figures are properly mapped, the red one has lots of artifacts in the left-bottom corner. It can not be repaired and will mess that figure in later processing. The next step is conversion of the difference mask into borders map. Each pixel of difference mask is analysed with the additional information of its two neighbours, one on the right, and one at the bottom. If the checked pixel is white (belongs to the object), and exactly one from the mentioned neighbours is black we can toggle on pixel of these coordinates on the borders map. The results are shown in Fig. 3 (b) and Fig. 2 (d). Now the borders map can

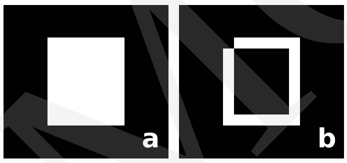

Fig. 3. Border mask creation: (a) difference mask, (b) border mask.

be easily divided into a set of figures by the observer looking at it. The same functionality has also been implemented in the program. In the end we want to obtain a vector of figures represented by the vector of consecutive coordinates which seems to be the most adequate for displaying purposes. The algorithm looks through the borders mask in search for white pixels. If the border is found a new figure is created and its coordinates are added to it as its first point. Next points are determined by checking all eight neighbours of found pixels (in our implementation it is done clockwise). If the white pixel is found, it is added as the next border point of current figure. It is also added to another vector of used pixels, which prevents the program from creating multiple figures based on the same border. If the pixel cannot be found in the neighbourhood we have to drop it and come back to the one we analysed before to check if another neighbour pixel is toggled. This operation is recursively repeated till we meet the starting point again or end up with the empty border coordinates vector. In the second case the figure is deleted.

3 Implementation details

The application that implements these two algorithms and creates 3D scene has been written. It is developed in C++ with Qt, OpenCV and OpenGL libraries. The application was tested under Linux OS with V4L support, but because all used libraries are fully portable there is a high probability that it can be compiled and run on other operation systems without major changes. The face tracking algorithm is implementation of CAMSHIFT algorithm published by G. Bradski [1]. In application we have used the code from the CAMSHIFT example file from OpenCV documentation. Application is released under GPL and can be downloaded from http://alan.umcs.lublin.pl/˜kadmi/rftandor.tar.gz.

4 Results

The final result of our efforts is virtual scene creation based on the gathered data. The example is shown in Fig. 4. The scene contains the figures created with the algorithm described in paragraph 2.2 placed at the bottom of wireframe corridor. Figure images should be created externally and loaded into the application. Virtual camera of this scene moves in three dimensions according to the tracked head position.

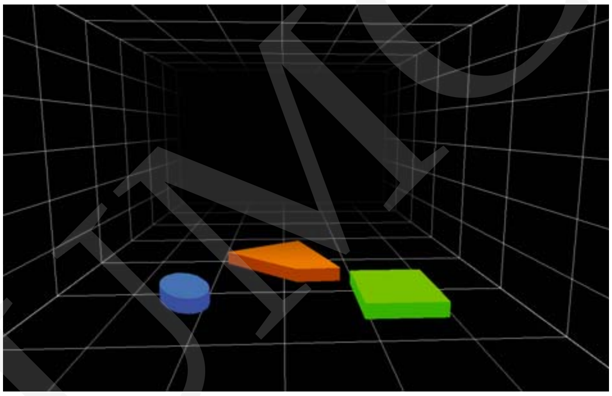

Fig. 4. 3D scene – result of our actions.

Both algorithms are colour based. To avoid problems with figures detection we should use colours contrasting with the background. Otherwise some figures could be omitted. Another problem already mentioned is automatic histogram equalization present in the tested cameras. It causes difference in colour of background when figures are placed from the scene without them. Contrast is also important for face tracking. Too little differences between the background and the tracked object (like i.e., skin and wood) lead to recognize too much area as the recognized object. Also moving too fast (causes blur) and moving out of camera view often results in losing focus.

To run the application properly a little set up of environment is required, but when basic requirements are passed, results can be satisfying.

References

[1] Bradski G. R., Computer vision face tracking for use in a perceptual user interface, Intel Technology Journal Q2 (1998):1–15.

[2] Comaniciu D., Meer P., Mean Shift: A robust approach toward feature space analysis, Pattern Analysis and Machine Intelligence 24(5) (2002): 603–619.

[3] Freeman W. T., Tanaka K., Ohta J., Kyuma K., Computer vison for computer games, Proc. FG '96 Proceedings of the 2nd International Conference on Automatic Face and Gesture Recognition, IEEE Computer Society Washington (1996): 100–105.

[4] Freeman R., Steed A., Zhou B., Rapid scene modelling, Registration and Specification for Mixed Reality Systems, Virtual World Content Creation & Management, Proc. of the ACM symposium on Virtual reality, ACM New York (2005): 147–150.