Non–traditional image segmentation and filtering

Ján Cíger, Department of Computer Graphics and Image Processing *
Jaroslav Plaček, International Laser Center, Bratislava †

Faculty of Mathematics and Physics, Comenius University, Bratislava

Abstract

The first task of every computer vision system is separating interesting objects from the background. Contemporary systems have to take into account, that the background could be very complex, consisting of various objects and artefacts. This poster describes a fast color–based algorithm for the segmentation of human hand from the image acquired by the CCD camera with further refinement of object contour.

Keywords: Virtual reality, motion capture, CCD cameras, image segmentation, contour processing

1 Introduction

Often the gray–scale images could by used for object vision, especially in cases, when the background is of uniform/contrasting color (e.g. goods on the transporter). But when an attempt to track and/or recognize human motions or gestures is made, this approach is very difficult. Overview of some projects is at [1].

If color is available, it may be good to use it for segmenting human limbs from the environment, see e.g. [5]. The brute–force approach (thresholding on RGB or HSV components) produces too much artifacts, thus it would be very difficult to use such result for further processing (e.g. object contour finding or pattern recognition). It is necessary to use something smarter. We used a combination of several methods to achieve a very clean and proper segmentation.

2 Some terms

- Normalized color
  $\bar{r} = \frac{r}{(r+g+b+1)}, \bar{g} = \frac{g}{(r+g+b+1)}, \bar{b} = \frac{b}{(r+g+b+1)}$
  Where r, g, b are coordinates of a point in a RGB color space, $\bar{r}$, $\bar{g}$, $\bar{b}$ are coordinates in normalized color space. This normalization helps to avoid effects of different lighting. This was published in [2].

- Hashmap – Data structure, combination of the hash table and a map, holding information about colors required to mark points as part of an object. It is a square matrix of counters.

*janoc@woc.sk
†jarop@ilc.sk

3 Algorithm description

The presented algorithm works by comparing colors present in a set of calibration objects (”training set”) with colors in a captured bitmap. Colors present in objects from the ”training set” are stored into a hashmap. Segmentation is done by looking up the color of each pixel from the image in the hashmap. Pixels, with color present there, are treated as belonging to the object. Similar algorithm but using color histograms, was published in [2].

The normalized color vectors are transformed to 2D space due to reduction of the memory space required for storage of the training data set. When determining the presence of the color in the hashmap, its coordinates are also transformed to mentioned 2D space. Then the comparison of the counter representing the occurrence of similar colors in training set with the threshold value (required number of pixels with similar colors in the training set) gives us the decision whether the pixel should be considered as the object point.

3.1 Calibration

Calibration is done only once. It’s main purpose is to ”teach” the system, which colors are in the objects. It is done by selecting portions of the image and storing the colors found into the hashmap. This hashmap is subsequently used in the segmentation algorithm for all captured images.

Calibration algorithm is very simple, it just fills in a hashmap (see algorithm 1).

3.2 Segmentation

Segmentation works as described in the algorithm 2. This algorithm is optimal, because it runs in $O(n)$ time, where n is number of pixels in the picture. Color lookup for each pixel is done in $O(1)$ time.

4 Contour refinement

Segmented image is suitable for contour tracing by standard algorithms, but when the image is noisy (almost al-

ways), it is desirable to smooth the result. Traditional algorithms described in e.g. [3],[4], or curve fitting are slow and very complex. Another alternative is filtering and sub–sampling.

Input is the object contour created by standard tracing algorithm. It is the vector of the adjacent points from the border. The output vector must fulfill two requirements:

1. We want to have approximately equal distances between individual contour points.
2. The contour should be smooth - removing small and unwanted discrepancies (noise) from original data.

This can be achieved by constructing the new contour with algorithm 3.

5 Conclusions

No recognition algorithm would work properly when the input would be corrupted somehow. Therefore the proper segmentation is crucial for successful processing as well as the creation of the correct contours. Segmentation algorithm described there is very robust, it's only known limitation in its current implementation (with ignored green color component) is sensitivity to yellow color, which can interfere with segmentation. This can be fixed by using three-dimensional hashmap instead of only two-dimensional.

The results, we obtained, are promising for using this algorithms in human hand tracking and gesture recognition later.

Figure 1: Original scene

Figure 2: Segmented scene

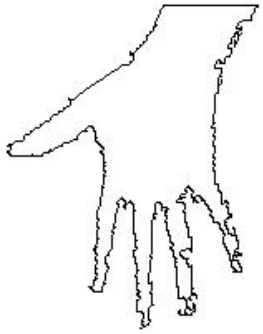

Figure 3: Raw contour

Figure 4: Refined contour

References

[1] http://ls7-www.cs.uni-dortmund.de/research/gesture/vbgr-table.html. Vision Based Hand Gesture Recognition Systems.

[2] http://www.subutai.com/asilomar94.ps.gz. A Usable Real-Time 3D Hand Tracker.

[3] Ferko A. and Ružický E. *Počítačová Grafika a Spracovanie Obrazu*. Sapientia, Bratislava, 1995.

[4] Hlaváč V. and Šonka M. *Počítačové Vidění*. Grada, Praha, 1992.

[5] Gonzales R.C. and Woods R.E. *Digital Image Processing*. Addison-Wesley, 1992.

Algorithm 1 Calibration

- NC – Normalized color, triplet of red, green and blue component
- H – Hashmap
- D – dimension of the Hashmap

```
interactively select area
from the image

FOR each pixel in selected area DO

  calculate normalized color NC
  calculate hash keys u,v :
     u = NC.red * D
     v = NC.blue * D

  H[u][v] ++;

END FOR
```

Algorithm 2 Segmentation

- C – pixel color (RGB triple)
- H – hashmap from calibration
- L – parameter, defines required frequency of the color to consider it as an object color
- NC – normalized color
- D – dimension of the hashmap

```
FOR every pixel DO

  calculate normalized color NC

  calculate hash keys u,v :
     u = NC.red * D
     v = NC.blue * D

  IF H[u][v] > L
  THEN
    mark point as object point
  ELSE
    mark point as background point

END FOR
```

Algorithm 3 Contour refining

- D – distance between points in the output contour
- C – output contour

```
store 1st point into output contour C

FOR each point DO
  calculate moving average of n points
  (center of gravity G of this segment)

  IF | C.last() - G | > D THEN
    store point into C
END FOR
```