Robust Real-Time Tracking of Non-rigid Objects

Richard Y. D. Xu, John G. Allen, Jesse S. Jin
School of Information Technologies
University of Sydney, NSW 2006, Australia
{richardx,jallen,jesse}@it.usyd.edu.au

Abstract

Several approaches to real-time video object tracking are reviewed. A new alternative approach for fast real-time object tracking based on colour thresholding is presented. Tracking is performed on non-rigid objects in a sequence of video frames based on a user-selected region of the initial frame. Details of the tracking algorithm, including colour cluster representation and pixels region grouping using run-length and noise filtering algorithm are discussed in detail. The exact contours of the tracked objects are then extracted by minimizing snake energy of thresholding results. We also experimented alpha blending the thresholding results with Canny filter edge maps to achieve more robust tracking. Foreground object colour cluster extraction at the initial frame using K-means algorithm and filtering via Foreground Extraction Mask is also discussed.

Keywords: colour thresholding, non-rigid object tracking, run-length, foreground extraction, Snake energy

1 Introduction

Video Object Tracking plays a very important role in many vision applications. Apart from applications traditionally for video surveillance, object recognition, and video segmentation and indexing, real-time object tracking is now extensively used in audio-visual speech recognition (Liu 2002), human gesture recognition, and object based video compressions, such as MPEG-4.

When priori knowledge about shape or motion of the object is known, a parameterised model of the object can be used. This type of tracking is also known as model-based tracking. It is frequently used to track balls in sports, cars in traffic (Koller 1992) and hands in human body (Ouhaddi 1999).

This paper, on the other hand, discusses object tracking using information from a user-selected region of the initial frame, which contains objects of interest without priori information. A strong focus of this paper is performance. Robust real-time tracking of non-rigid objects is required in many applications.

Some tracking algorithms use point and edge information. It involves firstly detecting edges using Canny (Canny 1986) or Deriche filters. Tissainayagam et al. (2003) tracks key points of the contour after segmenting each frame using Canny filter and merging them by distance testing. The key points used in tracking are the local maxima or the turning points of the contours.

A few other tracking algorithms are based on minimizing statistical difference between target and candidate model probably density distribution function, such as employing Mean-shift algorithm (Comaniciu et al. 1999).

This paper discusses tracking objects using fast colour thresholding as colour information provides an efficient feature. They are robust to partial occlusion and geometry invariant, and computationally efficient.

2 Tracking Using Colour Clustering

2.1 Colour Space

RGB colour space is most native to many video capture devices. However, RGB colour space is very sensitive to change in brightness and intensity. HSI and YUV colour spaces are often used, as the intensity component can be treated separately to the chrominance components. The transformation is done by software conversion, which can be quite expensive. Price et al. (2000) have proposed a modified HSI by extending 360 degrees of hue for faster processing. For this project, we use YUV for less expensive conversion from RGB. We use the OpenCV implementation (OpenCV 2003) which takes advantage of MMX instruction to perform faster RGB to YUV conversion.

2.2 Colour Representation

Comparing each colour pixel value with the tracked colour cluster information can be computationally expensive if it is performed by exhaustive search of all colour clusters for the object of interest. We have adopted a modified approach to Bruce et al (2000), where each colour cluster is represented by ranges of indices to every colour component bin. The ranges of indices are stored in an array per colour component and we refer each array as Colour Space Component Array (CSCA).

In our project, we use YUV colour space, hence three CSCAs are used. To make search even more efficient, each CSCA element can be represented by an integer, and each bit indicates one of the 32 tracked colour clusters.

We can efficiently determine if a particular pixel belongs to any of the 32 colour clusters by using only two AND operations on three integers. We specify a Multiple Colour Thresholding function (MCT) to determine if a pixel belongs to the tracked object. The results are then to be used in the subsequent sections.

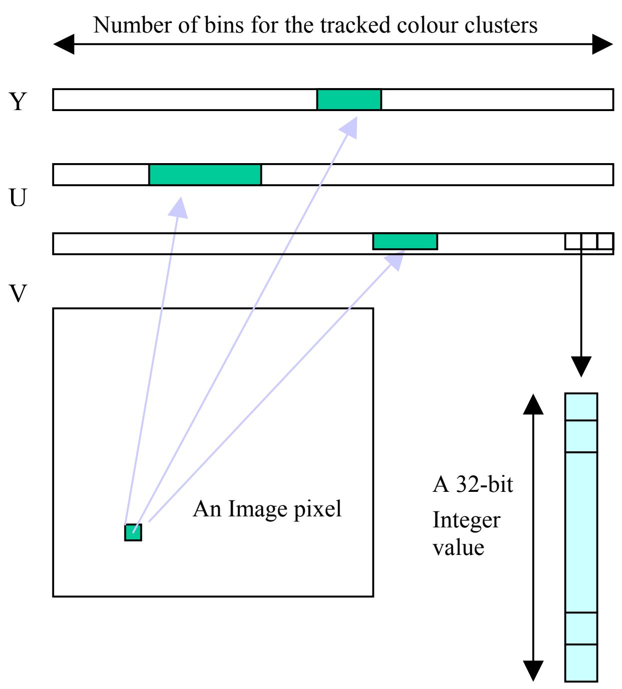


Figure 1: The representation of Colour Space Component Array[1]

2.3 Region Grouping and Noise Filtering

Colour thresholding is very sensitive to noise, the larger the threshold values the higher the noise. Several steps are used for region grouping and noise filtering;

1) The selection region is down-sampled using Gaussian pyramid sub sampling (Gonzalez 2002) to smooth the image. We then apply Multiple Colour Thresholding function (MCT) on each of the pixels in the selected region.
2) A Colour Cluster Window Filter (CCWF) is applied to each pixel before the next step. There are advantages to use CCWF instead of a smoothing spatial filter for noise reduction. Smoothing spatial filter attracts high computational cost as it needs to calculate the weight average using every pixel in the window. As stated in Section 2.1, calculate MCT only involves two Integer AND operations per colour pixel. We specify *CCWF(i)* as:

$$CCWF(i) = \begin{cases} 1 & \dfrac{Gsum}{\|CCFW\|} > 1/2 \\ 0 & otherwise \end{cases}$$

where $Gsum = \sum_{i \in CCFW} i$, if *MCT(i)* = 1.

3) Run length coding is used for region grouping. It also servers the purpose of integrating smaller regions inside the object of interest to merge together. We apply run length coding instead of region growing for efficient computation.
4) We then apply a vertical filter to the above result, which removes the horizontal strips created as result of run length coding.

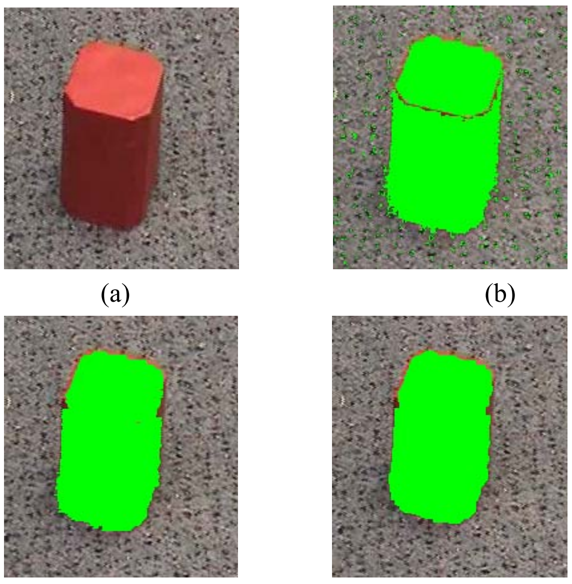


Figure 2: Objects of interest at each step in 2.3

a) Original video frame b) After applying MCT (In this case, 2 distinctive colour clusters concentrates at the top and around the side of the paper cup) c) Run-length coding reduces noise and eliminates small regions. d) Vertical filter removes horizontal strips

2.4 Foreground Object Extraction

2.4.1 Colour Clusters Determination

We apply well-known K-means clustering to determine all the colour clusters for the selected region. At each iteration a pixel value is assigned to the cluster with the closet cluster centre, after which the cluster centre is updated to be the centre of mass of all pixel values belonging to that cluster. By experiment, we notice the results favour YUV colour spaces compare with RGB, we are not able to group some of the similar colours with minor intensity contrast in RGB colour spaces, hence we obtain a lot more colour clusters in RGB colour space than those performed using YUV. As stated in Section 2.1, this is another reason we choose to perform computation in YUV colour space even it involves colour space conversions.

2.4.2 Foreground Extraction Mask

Note that because the user-selected region contains colour clusters belonging to both the object of interest and the background, we need to extract the clusters of foreground (object of interest) from above result. Instead of extracting the foreground by using traditional background subtraction, which obtains moving objects and newly appeared objects by subtracting the background from the current frame (Wu 2001)**,** we use a simple Foreground

[1] Mapping of an image pixel value to the CSCAs:

Each element of the array contains an integer value which has 32 bits. Effectively we can determine if a pixel belongs to one of 32 colour clusters using two integer AND operations

Extraction Mask (FEM) on the initial frame. We assign weights to the user selected regions with high positive values towards the centre of the selection and low negative values towards the edges of the selection. Our method does not involve any inter-frame differentiation.

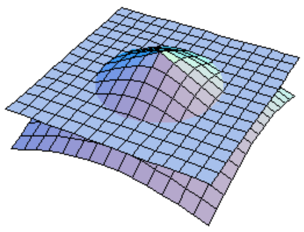


Figure 3: FEM applies high positive values towards the centre and low negative values towards the edges of the selected region.

This only needs to apply to the user-selected window in the initial frame. Once the colour clusters have been determined and CSCA (see Section 2.2) is filled, the consequent frames are examined by CSCA information only and executes steps in Section 2.3. The current search window is slightly larger than the previous frame and the centre of the search window is updated progressively as centre of all tracked colour pixels move.

The formulation of FEM can be simplified as using $FEM(x,y) = h_i - \sqrt{x^2 + y^2}$ on points near the centre (generates high positive value) and the edges (low negative value) of the user selected region and assigns zero in other pixels of the region. User can interactively alter values of h_i to filter out background colour clusters. However, experiments show that user adjustment is not required when the object region has a moderately distinct colour distribution from the background.

2.5 Contour Extraction

At each frame, using steps 1-4 in Section 2.3, we obtain a binary image that contains the object of interest. We then apply the OpenCV implementation (OpenCV 2003) of contour extraction algorithm based on (Suzuki 85) to the binary image. This generates a collection of external contours. We obtain the refined contours by minimizing the snake energy of each of the returned contours with respect to the binary image. Experiment shows that most of the time a single contour is obtained.

2.6 Alpha Blending with Edge Map

We have also tested results by performing alpha-blending of binary image generated from steps 1-4 in Section 2.3 with Canny filter binary mask image implemented by OpenCV. By varying alpha, we find for alpha > 0.5 towards the Canny filtered binary mask images, the resultant contours are more robust. We implemented our alpha blending using MMX instructions to achieve higher performance.

3 Empirical Results

We have tested our approach using various video downloaded from the Internet. Most of the video with relative distinct colour features between the object and the neighbouring regions are able to track non-rigid object of interest and extract its contour successfully.

We have performed the experiments on a standard Pentium4 machine and are able to achieve real-time tracking.

Figure 4 shows user selection contains the object of interest. The colour clusters are calculated and CSCA information is updated.

Figure 5 shows binary image as a result of section 2.3-2.5 and alpha-blended with Canny filter mask (section 2.6). For the purpose of illustration, we superimpose the result with Canny edge map for the entire image. The Bold White line showing the contour of the object, and red line is the smallest enclosing square of that contour.

Figure 6 shows experiments to track objects in 2 video sequences are successful.

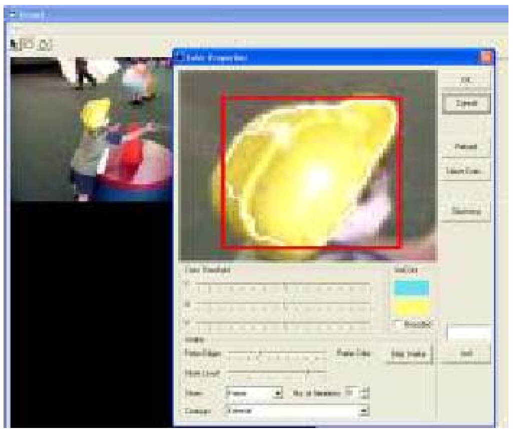


Figure 4: Enlarged user selection region showing colour cluster feature used for tracking.

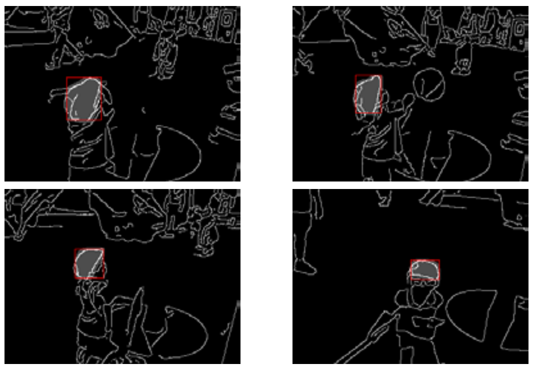
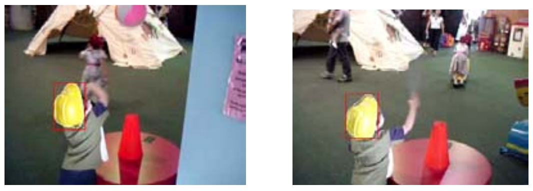


Figure 5: shows binary image as a result of section 2.3-2.5 and alpha-blended with Canny filter mask (section 2.6).



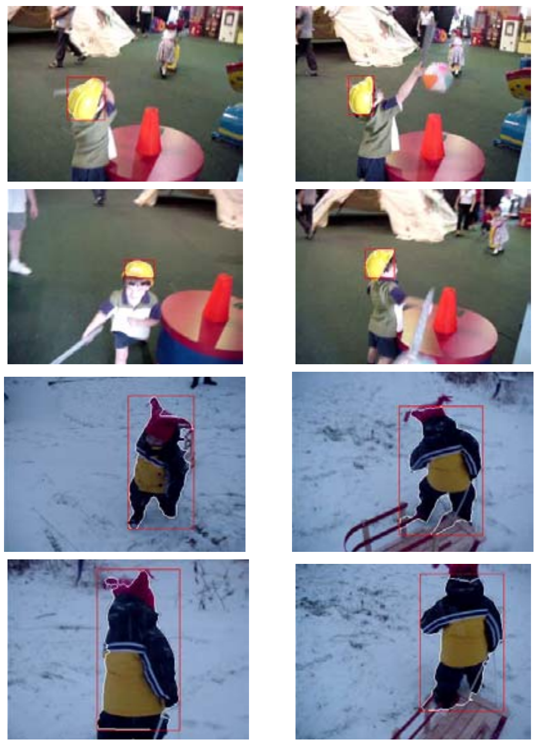


Figure 6. Two testing sequences of tracking.

4 Conclusion

This paper presents an approach for fast real-time non-rigid object tracking based on colour thresholding. The object is in a user-selected region in the initial frame. The experimental results show the algorithm is fast and requires minimum user interaction.

Future research focuses are concentrated on combining other features of the video such as edges and texture together with colour information to provide an even more robust tracking algorithm.

References

J. Bruce, T. Balch, & M. Veloso (2000). Fast and Cheap Color Image Segmentation for Interactive Robots *Proceedings of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '00)*, Vol. 3, October, 2000, pp. 2061 - 2066.

Zhengping Wu & Chun Chen (2001). A new foreground extraction scheme for video stream*s, ACM Multimedia 2001 Electronic Proceedings* http://www.acm.org/sigmm/mm2001/ep/wu/index.html

S.Suzuki, K.Abe. (1985) Topological structural analysis of digital binary image by border following. *CVGIP*.30(l): 32-46.

P. Tissainayagam & D. Suter (2003). Object tracking in image sequences using point features, *APRS Workshop on Digital Image Computing Online Proceedings* http://www.aprs.org.au/wdic2003/CDROM/69.pdf

D. Comaniciu and P. Meer. (1999) *Mean shift analysis and applications*. In IEEE International Conference on Computer Vision, pages 1197--1203, 1999.

Xiao Xing Liu, Yibao Zhao, Xiaobo Pi, Lu Hong Liang & Ara V Nefian, (2002). Audio-visual continuous speech recognition using a coupled hidden Markov model, *IEEE International Conference on Spoken Language Processing*, pp.213-216, September 2002.

J. Canny (1986) Computational approach to edge detection, *IEEE T-PAMI* 8(6):679-698.

H. Ouhaddi & P. Horain (1999) 3D hand gesture tracking by model registration*,, Proc. IWSNHC3DI'99*, 1999.

Koller, K. Daniilidis, T. Thorhallson, & H.-H. Nagel.(1992) Model based object tracking in traffic scenes,. *Proceedings of ECCV '92*. Springer-Verlag, 1992.

A. Price, G. Taylor & L. Kleeman (2000) Fast, robust colour vision for the monash humanoid, *Australian Conference on Robotics and Automation*, Melbourne, August 30 - September 1, 2000, pp 141-146.

Rafael C. Gonzalez, Richard E. Woods. (2002): *Digital Image processing,* Second Edition. Prentice Hall International.

OpenCV (2003): Intel Open Source Computer Vision Library http://www.intel.com/research/mrl/research/opencv/