International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:1, No:10, 2007

Using Mean-Shift Tracking Algorithms for Real-Time Tracking of Moving Images on an Autonomous Vehicle Testbed Platform

Benjamin Gorry, Zezhi Chen, Kevin Hammond, Andy Wallace, and Greg Michaelson

***Abstract*—**This paper describes new computer vision algorithms that have been developed to track moving objects as part of a long-term study into the design of (semi-)autonomous vehicles. We present the results of a study to exploit variable kernels for tracking in video sequences. The basis of our work is the mean shift object-tracking algorithm; for a moving target, it is usual to define a rectangular target window in an initial frame, and then process the data within that window to separate the tracked object from the background by the mean shift segmentation algorithm. Rather than use the standard, Epanechnikov kernel, we have used a kernel weighted by the Chamfer distance transform to improve the accuracy of target representation and localization, minimising the distance between the two distributions in RGB color space using the Bhattacharyya coefficient. Experimental results show the improved tracking capability and versatility of the algorithm in comparison with results using the standard kernel. These algorithms are incorporated as part of a robot test-bed architecture which has been used to demonstrate their effectiveness.

***Keywords*—**Hume, functional programming, autonomous vehicle, pioneer robot, vision.

I. Introduction

THIS paper compares and contrasts three computer vision algorithms for tracking moving objects in video sequences. Detecting and following moving objects against complex backgrounds is critical in a number of autonomous vehicle (AV) applications. For example, in a full-scale AV it may allow us to track and avoid collisions with pedestrians or moving vehicles, and in a robotic context, it may allow improved navigation and enhance safety. Good isolation of individual moving objects will allow us develop applications that involve following targets of interest. All these applications require us to process full-color video sequences in real time.

Our work is undertaken in the context of a robotic AV testbed platform, based on the Pioneer P3-AT all-terrain robot, as part of a broader UK project that is developing new sensor technology for autonomous vehicles, and funded by the Systems Engineering for Autonomous Systems (SEAS) Defense Technology Centre (DTC). The SEAS DTC is operated by a UK industrial consortium and aims to research innovative technologies relevant to autonomous systems, at both whole-system and sub-system level and, through the adoption of Systems Engineering approaches, to facilitate pull-through of the technology into military capabilities.

This paper makes a number of novel contributions. Firstly, we describe new algorithms for tracking moving images against complex, cluttered backgrounds, based on previously studied *mean-shift* algorithms. Secondly, we show that our algorithms are capable of tracking moving targets for full-size video images in real time. Thirdly, we show how the algorithm can be deployed in a simple AV testbed, based on a Pioneer P3-AT all-terrain robot. Finally, our implementation is unusual in being written using the novel programming language Hume [1, 2], a language that combines functional programming concepts with finite-state automata for programming real-time reactive systems.

B. Gorry, Z. Chen and G. Michaelson are with Department of Computer Science, Heriot-Watt University, Riccarton, Scotland.

K. Hammond is with School of Computer Science, University of St Andrews, St Andrews, Scotland.

A. Wallace is with Department of Electrical & Computer Engineering, Heriot-Watt University, Riccarton, Scotland.

II. The Mean-Shift Vision Algorithm

A. Mean Shift Segmentation

We present the results of a study to exploit various kernels for real-time tracking of moving objects in video sequences. The kernels are defined by the underlying, simple, robust, and diverse *mean shift*, clustering algorithm [3], first applied to image segmentation by Comaniciu and Meer [4, 5, 6]. For the autonomous vehicle application, the task is to first define an object of interest, by segmentation and/or by interactive selection, then by tracking the object as it moves within the camera field of view. The mean shift algorithm is designed to find modes (or the centers of the regions of high concentration) of data represented as arbitrary-dimensional vectors. The algorithm proceeds as follows [7].

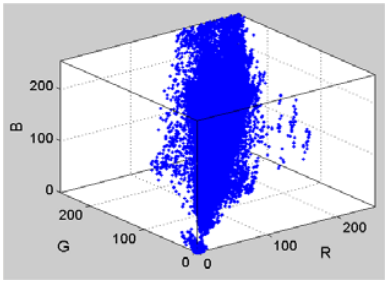
- Choose the radius of the search window.
- Choose the initial location (center) of the window.
- Repeat

  Compute the mean (average) of the data points over the window and translate the centre of the window into this point.
- Until the translation distance of the center becomes

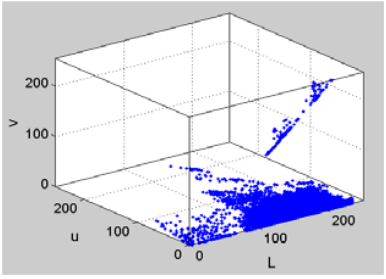
less than a preset threshold.

High-density regions in the feature space correspond to sufficiently large numbers of pixels in a narrow range of intensities/colors in the image domain. Therefore, provided the pixels form connected regions (as is often the case for relatively smooth images); the algorithm essentially finds relatively large connected regions that have sufficiently small variations in intensity/color (and are thus perceived as well defined regions by humans). In practice, the algorithm proceeds by placing randomly one search window at a time, finding the corresponding mode, and removing all the feature vectors in the final window from the feature space. Thus one would expect to find larger regions first.

(a) A 320×240 colour image.



(b) The corresponding RGB image.

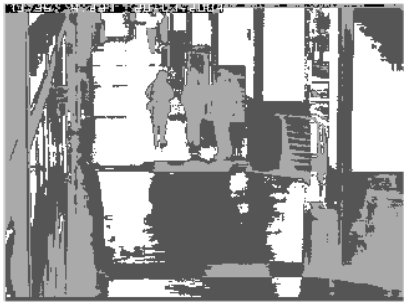


(c) The corresponding Luv colour space.

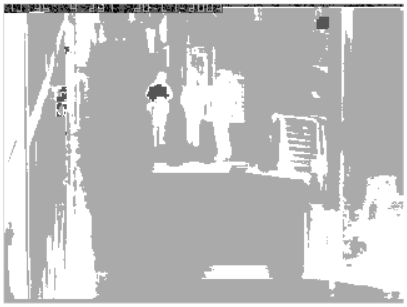
Fig. 1 Relationship between image, RGB and Luv

In our implementations, before segmenting color images, pixels, usually represented in the ***RGB*** color space, are mapped into the ***Luv*** color space which has a "brightness" component represented by ***L*** and two "chromatic" components represented by ***u*** and ***v.*** It has been argued that the latter color space is more isotropic and thus is better suitable for the mode finding algorithm. Finally, when defining a variable kernel, this is constrained within a rectangular window that we also used when displaying the tracking result.

Fig. 1 illustrates an example of the relationship between image and feature space. Fig. 2 shows the segmentation results in ***RGB*** and ***Luv*** space, respectively. Subjectively at least, this shows an improvement in using the ***Luv*** parameterization for this particular example. The flowchart of implementation of ***RGB*** to ***Luv*** by Hume is shown in Fig. 3. The flowchart of mean shift segmentation is shown in Fig. 4.



(a) Segmentation results in RGB colour space



(b) Segmentation results in Luv colour space

Fig. 2 Segmentation results

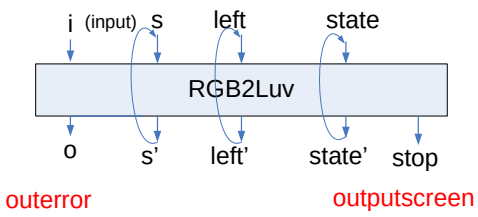


Fig. 3 The flowchart of RGB to Luv (Hume)

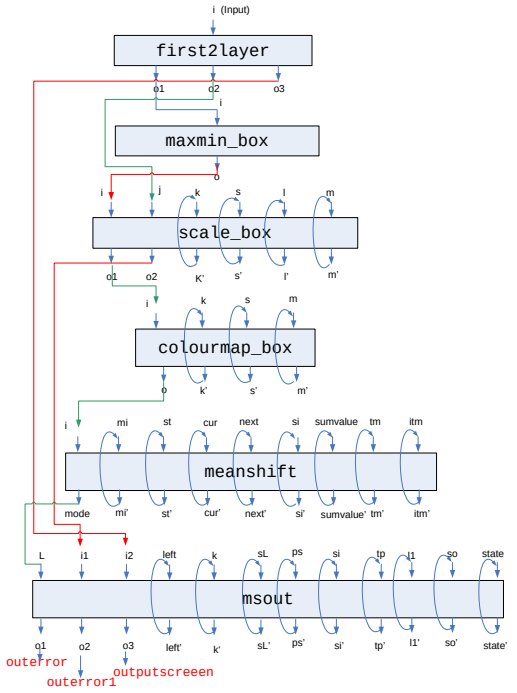


Fig. 4 The flowchart of mean shift segmentation algorithm (Hume)

B. *Mean Shift Object Tracking*

A rectangular window is defined about the region of interest in an initial frame. Then the mean shift algorithm is applied to separate the tracked object from the background in Luv color space. As the object moves, an unusual kernel weighted by the Chamfer distance transform improves the accuracy of target representation and localization, minimising the distance between two color distributions using the Bhattacharyya coefficient. In tracking an object through a color image sequence, we assume that we can represent it by a discrete distribution of samples from a region in color space, localised by a kernel whose centre defines the current position. Hence, we want to find the maximum in the distribution of a function, ρ, that measures the similarity between the weighted color distributions as a function of position (shift) in the *candidate* image with respect to a previous *model* image. If we have two sets of parameters for the respective densities $p(x)$ and $q(x)$, the Bhattacharyya coefficient [8] is an approximate measurement of the amount of overlap, defined by:

$$\rho = \int \sqrt{p(x)q(x)}dx \tag{1}$$

Since we are dealing with discretely sampled data from color images, we use discrete densities stored as *m*-bin histograms in both the *model* and *candidate* image. The discrete density of the model is defined as:

$$\mathbf{q} = \{q_u\}, u = 1,2,\cdots,m \quad \sum_{u=1}^{m} q_u = 1 \tag{2}$$

Similarly, the estimated histogram of a candidate at a given location **y** in a subsequent frame is:

$$\mathbf{p}(\mathbf{y}) = \{p_u(\mathbf{y})\}, u = 1,2,\cdots,m \sum_{u=1}^{m} p_u = 1 \tag{3}$$

According to the definition of Equation (1), the sample estimate of the Bhattacharyya coefficient is given by:

$$\rho(\mathbf{y}) = \rho[\mathbf{p}(\mathbf{y}), \mathbf{q}] = \sum_{u=1}^{m} \sqrt{p_u(\mathbf{y})q_u} \tag{4}$$

Let $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\}$ be an independent random sample drawn from $f(\mathbf{x})$, the color density function. If K is the normalized kernel function, then the kernel density estimate is given by:

$$q_u = \sum_{i=1}^{n} K(\mathbf{x}_i)\delta[b(\mathbf{x}_i) - u] \tag{5}$$

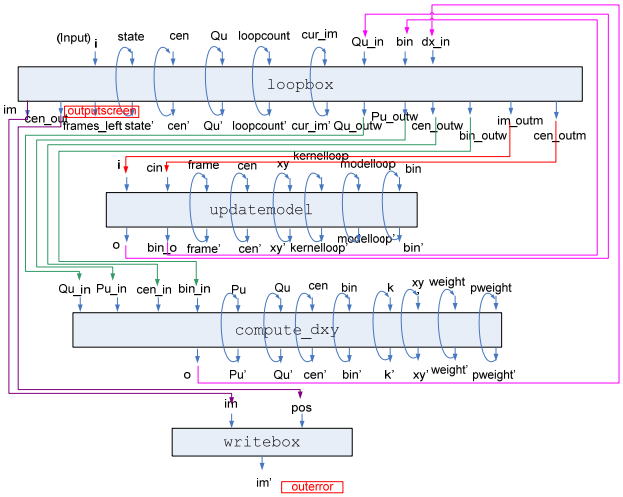


Fig. 5 The flowchart of the mean shift object tracking algorithm (Hume)

Estimating the color density in this way, the mean shift algorithm is used to iteratively shift location **y** in the target frame, to find a mode in the distribution of the Bhattacharyya coefficient (Equation 4). Using Taylor expansion around the values, $p_u(\mathbf{y}_0)$, the Bhattacharyya coefficient is approximated by [8]:

$$\rho[\mathbf{p}(\mathbf{y}), \mathbf{q}] \approx \frac{1}{2}\sum_{u=1}^{m} \sqrt{p_u(\mathbf{y}_0)q_u} + \frac{1}{2}\sum_{i=1}^{n} w_i K(\mathbf{x}_i) \tag{6}$$

where

$$w_i = \sum_{u=1}^{m} \delta[b(\mathbf{x}_i) - u] \sqrt{\frac{q_u}{p_u(\mathbf{y}_0)}} \tag{7}$$

To maximize Equation (4), the second term in Equation (6) is maximized as the first term is independent of **y.** In the mean shift algorithm, the kernel is recursively moved from the current location $\mathbf{y}_0$ to a new location $\mathbf{y}_1$ according to the relation:

$$\mathbf{y}_1 = \sum_{i=1}^{n} \mathbf{x}_i \, w_i G(y_0 - x_i) \Big/ \sum_{i=1}^{n} w_i G(y_0 - x_i) \qquad (8)$$

where G is the gradient function computed on K. This is equivalent to a steepest ascent over the gradient of the kernel-filtered similarity function based on the color histograms. The flowchart of mean shift object tracking is shown in Fig. 5.

III. Deploying The Tracking Algorithms on an Autonomous Vehicle Testbed Platform

Our hardware testbed consists of a Pioneer P3-AT all-terrain robot [9], SEBO (SEas roBOt, Fig. 6). We have configured SEBO with a front array of sonar discs, radio Ethernet, front and back safety bumpers, and a surface-mounted camera that is used to collect data for the mean-shift vision algorithms. Our Hume implementation interfaces to standard software supplied with the Pioneer robot: ARIA (Advanced Robotics Interface for applications), an open-source development environment which interfaces to the robot's microcontroller, and provides access to basic motor and camera functions; and VisLib, an open-source C-based vision-processing library that provides basic image processing capabilities.

Fig. 6 SEBO - the Heriot-Watt/St Andrews Pioneer P3-AT

A. Software Architecture

The software architecture for our testbed implementation is shown in Fig. 7. Solid arrows represent local socket communication on the robot while broken arrows represent wireless socket communication. All source code is located on the robot apart from the Java GUI, which runs on a laptop computer. Real-time images are captured from the robot camera, passed through an image-processing program located on the robot and written in Hume, and then sent wirelessly to a laptop computer which displays the images in real-time. For each image a Red, Blue, and Green component is captured. The image size which is captured is 240 x 320, thus requiring a storage structure of size 3 x 240 x 320.

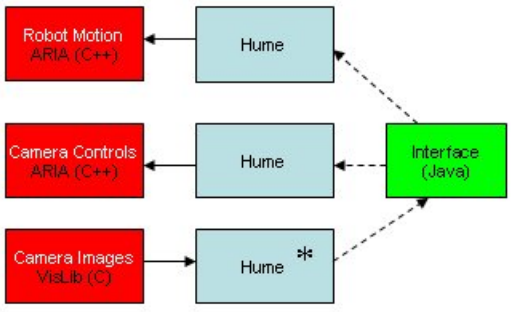


Fig. 7 Robot Test-bed Architecture

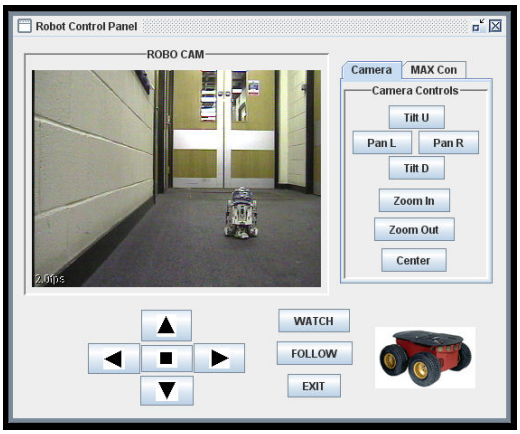


Fig. 8 Screenshot of Interface

We have implemented a simple command interface on the laptop. When the user decides to move the robot, a signal is sent wirelessly from the laptop computer to a Hume program located on the robot. The Hume program then communicates with a C++ ARIA program which sends the basic motor commands to the robot. The camera is controlled in a similar way to the movements of the robot. Fig. 7 shows that as the user selects to control the camera a signal is sent wirelessly from the laptop computer to a Hume program located on the robot. The Hume program then communicates with a C++ ARIA program which sends commands to the camera. The camera panel, displayed on the top left part of Fig. 8, is used to control the pan, tilt, and zoom features of the camera. Two sets of camera controls are provided. The first set allows minimal movement or focus values for the camera while the second set allows maximal movement or focus values.

B. Incorporating the Vision Algorithms

The Hume pass-through box marked with an asterisk in Fig. 7 has been successively replaced with:

1 the LUV conversion algorithm;

2 the mean-shift segmentation algorithm;

3 the mean-shift object-tracking algorithm.

These algorithms produce different image results. From the initial experiments, each of these algorithms can be incorporated as part of the test-bed architecture by simply replacing the Hume box which passes the images from the camera to the Java interface. Work has begun on identifying dependencies between each algorithm and establishing efficient link points where required.

For the LUV conversion algorithms, images are presented in the LUV color space. For the mean-shift segmentation algorithm, experiments have included using a variety of image types and sizes. Normally, images of size 240 x 320 are processed by the algorithm. Initial work with the mean-shift object-tracking algorithm shows encouraging results. For an object placed in central view of the camera, the robot or camera is moved at a steady rate, the object is tracked on the screen.

Current work includes introducing an option where the user can highlight an object on the interface screen by rubber-banding the object of interest. The co-ordinates of this object, relative to its position on the screen, are then passed to the mean-shift object tracking algorithm. From this, if the object moves, the robot moves; or the camera on the robot moves, then the object is tracked using the algorithm discussed in section 2.2. This is visible on the interface screen.

C. Implementation and Experimental Evaluation

Fig. 9 shows the first frame and the foreground image of the tracked object. In this case a simple regional homogeneity criterion has been applied as the target had relatively uniform intensity. Partial results of tracking a male pedestrian using the NCDT kernel are shown in Fig. 10.

Fig. 9 Rectangular window and segmentation

Fig. 10 Partial results of tracking a male pedestrian by NCDT kernel

D. The Robot Platform

Currently, the robot platform illustrated in Fig. 7 is being used to as a deployment architecture for the vision algorithms that have been developed in Hume. The use of these algorithms is proof of concept that the Hume implementations work, and also that Hume can be used in conjunction with other industry-standard languages such as C, C++, and Java. Each of the three algorithms discussed in Section 2 process images in real-time as they are captured from the camera mounted on the surface of the robot.

Possible extensions to the robot platform may involve de-localizing the sections of Hume code which link with the robot API. By doing this we could cost these individual programs to gain a measure of performance analysis. Alternatively, the three Hume programs indicated in Fig. 7 could be combined. This one program could then be analysed for its performance and we could also assess the reactivity rates when a robot or camera movement request is sent from the Java interface. For the mean-shift tracking algorithm a number of experiments are underway. These involve tracking objects of different sizes, colors, objects against similar and dissimilar background colors, and objects of different shapes.

IV. Related Work

Work on real-time tracking algorithms has taken place in a number of application areas. By capturing images in real-time and then performing image segmentation we can partition an image into several different regions. We can then use this information to track highlighted objects. In the area of processing these algorithms work has been carried out using FPGAs (Field Programmable Gate Arrays) instead of microprocessors [10]. This aims to take advantage of the low cost and parallelism associated with FPGAs. However, the algorithm discussed in [10] requires the use of clearly distinguishable fluourescent markers – the mean shift object

tracking algorithm discussed in this paper requires no such markers. Our initial experiments have shown that two similar objects, which differ in color in some small way, can be identified independently of each other.

In embedded real-time applications the importance of obtaining accurate bounds of time and space usage is extremely valuable [11, 12]. If we can predict how our system should behave we can place an upper bound on the expected execution time of one program cycle. Using Hume, this can be done. The key to the design of Hume is its ability to be costed. To provide structure to these costs, Hume has been developed as a series of language subsets which overlap [13]. Each superset of this overlapping series of subsets adds expressibility to the language. By choosing the appropriate level of language, the programmer can obtain the balance they require between language expressiveness and the required degree of costability. Therefore, we can identify the time and space bounds which are required – this allows us to identify exact quantities of hardware resources we require. What would be interesting is to deploy the Hume algorithms discussed in this paper on FPGAs. We could then use the Hume approach to cost the algorithms and compare the results against those obtained from costing the algorithms on a microprocessor.

V. Conclusion and Further Work

In this paper we have explored the use of variable kernels to enhance mean-shift segmentation. Experimental results show the improved tracking capability and versatility of our implementation of mean-shift object tracking algorithms when compared with results using the standard kernel. These algorithms are incorporated as part of a robot test-bed architecture which has been used to demonstrate their effectiveness. Each algorithm has been developed using Hume. By processing real-time images and communicating wirelessly with our robot, we can track moving images against complex, cluttered backgrounds.

Work is currently underway to extend out testbed platform by:

1. developing new image processing algorithms for AV deployment, and
2. complementing our motion-tracking algorithms by adding a line following algorithm.

This will involve extending the interface used to control the physical motion of the robot and camera. These extensions should further demonstrate:

1. how Hume can be used to develop algorithms which perform real-time processing,
2. the flexibility of our testbed platform, and
3. the accuracy of our tracking algorithms.

Acknowledgment

The work reported in this paper was funded by the Systems Engineering for Autonomous Systems (SEAS) Defence Technology Centre established by the UK Ministry of Defence. We would like to thank our collaborators in the EU FP6 EmBounded project, in particular Christian Ferdinand, Reinhold Heckmann, Hans-Wolfgang Loidl, Robert Pointon and Steffen Jost.

References

[1] K. Hammond and G. Michaelson, "The Hume Report, Version 0.3", 2006

[2] K. Hammond and G. Michaelson, "Hume: a Domain-Specific Language for Real-Time Embedded Systems", *Proc. of Int. Conf. on Generative Programming and Component Engineering*, Erfurt, Germany, Sept. 2003, Springer-Verlag Lecture Notes in Comp. Sci., pp. 37-56.

[3] Y. Z. Cheng, "Mean shift, model seeking, and clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8): 790 -799, 1995.

[4] D. Comaniciu, P. Meer, "Robust analysis of feature space: Color image segmentation," *In IEEE Conf. Computer vision and Pattern Recognition*, 750 – 755, 1997.

[5] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5), 603-619, 2002.

[6] D. Comaniciu, V. Ramesh, P. Meer, "Kernel-based object tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(5), pp564-575, 2003.

[7] Y. Keselman and E. Micheli-Tzanakou, "Extraction and characterization of regions of interest in biomedical images," *In Proceeding of IEEE International conference on Information Technology Application in Biomedicine (ITAB 98)*, 87-90, 1998.

[8] A. Bhattacharyya, "On a measure of divergence between two statistical populations defined by their probability distributions," *Bulletin of the Calcutta Mathematics Society,* 35, pp99-110, 1943.

[9] MobileRobots Inc., "Pioneer 3 Operations Manual with MobileRobots Exclusive Advanced Control & Operations Software", *MobileRobots Inc.*, January 2006.

[10] C. T. Johnston, K. T. Gribbon and D. G. Bailey. "FPGA based Remote Object Tracking for Real-time Control", *1st International Conference on Sensing Technology*, Palmerston North, New Zealand, 2005.

[11] G. Hager and J. Peterson, "FROB: A Transformational Approach to the Design of Robot Software", *Proc. of the ninth International Symposium of Robotics Research*, Utah, USA, 1999.

[12] D. Fijma and R. Udink, "A Case Sudy in Functional Real-Time Programming", *Technical Report, Dept. of Computer Science*, Univ. of Twente, The Netherlands, 1991

[13] K. Hammond, "Exploiting Purely Functional Programming to Obtain Bounded Resource Behaviour: the Hume Approach," *First Central European Summer School*, CEFP 2005, Budapest, Hungary, July 4-15, 2005, Lecture Notes in Computer Science 4164, Springer-Verlag, 2006, pp. 100-134.