Video Surveillance Analysis as a Context for Embedded Systems and Artificial Intelligence Education

M. Ryan Bales[1] and Steve E. Watkins[2]
[1]Georgia Tech Research Institute and [2]Missouri University of Science and Technology

Abstract

Video surveillance analysis is an exciting, active research area and an important industry application. It is a multidisciplinary field that draws on signal processing, embedded systems, and artificial intelligence topics, and is well suited to motivate student engagement in all of these areas. This paper describes the benefits of the convergence of these topics, presents a versatile video surveillance analysis process that can be used as the basis for many investigations, and presents two template exercises in tracking detected targets and in evaluating runtime efficiency. The processing chain consists of detecting changes in a scene and locating and characterizing the resulting targets. The analysis is illustrated for targets in outdoor scenes using a variety of classification features. Also, sample code for processing is included.

Introduction

The proliferation of low-cost cameras and high-performance embedded platforms has enabled the application of computer vision systems to a wide range of surveillance problems, making embedded video surveillance analysis an exciting and rapidly growing area. We propose that surveillance video processing is well suited as a context for discussing concepts within many fields including signal processing, embedded systems, and artificial intelligence. Surveillance video analysis entails several interesting, multidisciplinary, real-world problems that can be tackled in depth from the perspectives of these topics. Several educational benefits can result from discussing these topics within the context of the accessible problems of video analysis.

- Students can study algorithms and concepts with a specific type of data rather than having to imagine abstract data without a specific goal in mind.
- Everyday familiarity with vision and video eases debugging, as students can self-check results with what makes sense visually and intuitively.
- Artificial intelligence and machine learning techniques generally require large amounts of data. The accessibility of existing video datasets and the ease with which new data can be collected using inexpensive webcameras are much greater than for other data types, such as radar or financial data.
- The large amount of data that must be processed in video encourages efficient programming techniques.

As humans, we are adept at seeing our environment, recognizing and tracking objects, and extracting information. The challenge to students then becomes determining how to design machine vision systems that can imitate these intuitive tasks. Object classification is an important component of many video surveillance systems; by classifying targets, a system can

discriminately choose which targets should be observed closely, and tracking and other analyses can be improved. Traffic management, perimeter security, infrastructure monitoring, public safety, driver assistance, and wildlife observation are examples of scenarios in which such abilities are useful. Such systems have the greatest potential if sensors are distributed throughout an environment—a constraint that prohibits the use of large computers, and encourages the deployment of small, dedicated, embedded processors at each sensing node.

A key challenge within classification and tracking is how to determine a set of object features that best distinguishes the object types of interest. This application can introduce many machine learning techniques. A video processing pipeline that utilizes background subtraction, blob silhouette formation, and feature extraction is suitable for student experimentation. The approach has the flexibility to address a wide range of topics and it can be adjusted for many levels of difficulty. We focus on the problem of extracting and selecting features for object classification and tracking which an active area of research and development. This paper describes a multidisciplinary implementation of sample exercises, presents the video surveillance analysis process that can be used as the basis for such investigations, and presents two template exercises in classifying objects observed in surveillance video and in evaluating runtime performance on hardware. The analysis is illustrated for targets in outdoor scenes using a variety of classification features. Finally, a real-time demonstration system is made available to engage student interest across these related topics.

Educational Overview

A. Multidisciplinary Topics

Environments for video surveillance can vary from well-defined backgrounds such as a hallway to complex changing backgrounds such as a highway. Targets for detection may be vehicles, pedestrians, etc. Applications can relate to security, safety, monitoring, management, or simple observation. Knowledge of the capabilities and techniques for such systems can be useful to both specialists in computer vision and users in other technical fields. Consequently, educational exposure to video hardware and analysis can have a place in many fields of study.

Exercises involving video systems and analysis are especially suited as resources for courses in image processing, embedded systems, and artificial intelligence. The objectives of the exercises can involve algorithmic techniques, hardware customization, or intelligent interpretation, respectively. Video sequences are readily available or can be obtained with inexpensive cameras. Processing can be accomplished with available computers or dedicated boards. The surveillance analysis processes described here can be tailored to different complexities and difficulties, and the basic components can be used in a variety of contexts.

B. Related Work in Classification and Tracking

Objects of interest must be detected before they can be classified. While humans are adept at quickly picking out salient regions of scenes at many scales, computer vision algorithms tend to approach such problems from the bottom up by detecting changes at the pixel level, then

aggregating changing pixels into higher-level features. Adaptive background subtraction is employed as a well-established method of detecting salient changes.[1, 2, 3] The literature demonstrates the usefulness and flexibility of this approach.

Two general approaches to object detection have been explored. Template-based matching involves searching an image for expected patterns (often extracted offline from training data) and relies on photometric features. Template detection requires relatively specific a priori knowledge about the expected targets. Silhouette-based processing uses background subtraction to find generic object blobs, and focuses on geometric features of the blobs. This framework is favored for its versatility and efficiency[4].

Prior work has explored several target classification architectures—including neural network (NN) based classifiers[5, 6, 7], support-vector machines (SVM) [8, 9, 10, 11], Bayesian networks and decision trees[9]—in conjunction with a wide range of input features. Height, width, aspect ratio, geometric moments, photometric moments, color histograms, and fitted ellipses are a few of the features that have served as inputs to object classifiers and trackers. Feature selection is often motivated by intuition or relevance to the underlying algorithm framework, and while some investigations present the effectiveness of their approaches using various combinations of their chosen features, few explicitly explore the suitability of those features for the associated computer vision task. Here, we consider the consistency of some of the most common object features used in object classification and tracking.

Ali et al.[12] propose a grayscale running average BGM for object detection. Objects are divided into hypothesized human body components, and human classification is performed by a NN using area, perimeter, centroid, and principle axis of inertia of each component. Serratosa et al.[7] use the average and standard deviation of color components, area, and Hu moments[13] to classify instances of user-chosen segmented regions. Gepperth et al.[14] focus on car classification, and relies on vertical and horizontal gradients, energy of gradients, local orientation and mean energies of line segments as classifier inputs. Li et al.[15] use aspect ratio, compactness, and horizontal and vertical centroid offsets to classify pedestrians and bicycles. Kong and Wang[8] find that Hu moments are less effective for classification of non-rigid objects whose topologies can change (such as people), but work well for rigid objects. They incorporate Euler Number to improve classification of non-rigid objects. Gurwicz et al.[9] propose a broad feature set including luminance asymmetry, DCT coefficients, 2D moments, compactness, solidity, and aspect ratio. Several classifier architectures are considered for the purpose of classifying humans, body objects, groups of people, bags, and clutter. This work also introduces a procedure for evaluating the relevance of each feature to each object class, which internally rates the usefulness of each feature based on entropy contribution and preserves only the features that improve classification accuracy. In the work by Watkins et al.[16], a neural network is trained and tested to monitor bridge traffic, and to determine if detected targets are pedestrians. Objects are detected by subtracting a static background frame from new frames, and by subdividing the scene into predefined, human-sized strips. Fourteen features are computed from each object's silhouette and appearance, and are used as inputs to the network.

Techniques for object tracking also often use geometric and photometric object features. A general survey of object tracking techniques is provided by Yilmaz et al.[17] The described approaches are motivated by a variety of applications, and are suited for tracking many different object types depending on the problem domain. Trackers have been presented that focus on specific types of traffic, such as vehicles or pedestrians.

Several point detection mechanisms have been pro-posed for producing trackable feature sets, such as SIFT points[18], SURF points[19], and multiresolution critical points[20, 21]. These points are often fed into particle or other statistical filters. Trackers have been proposed that first identify blobs by background modeling and change detection, and then distinguish each blob with a small, simple set of figures such as object-strip color[22], purely kinematic principles[23], or spectral distribution[24]. These approaches are attractive because their features are efficient to compute, and they have inherently manageable search spaces.

C. Suggested Course Exercises

The intents of these exercises are to reinforce specific material in the host course and to provide guided procedures for other aspects. (More comprehensive experience could be assigned for course projects, honors research, capstone design work, etc.) Consider exercises for the three suggested topics, i.e. image processing, embedded systems, and artificial intelligence, as shown in Table 1. A basic processing pipeline is assumed here that uses background subtraction to find object blobs and that calculates geometric features of the blobs for detection (this methodology is described in detail in the next section). Note that variation in the dataset environments or targets can add an element of realism, can adjust the level of difficulty, and can change the solution for different students or different semesters.

Table 1: Potential Exercises

Main Topic	Instructor Provides	Student Controls	Exercise Results
Image Processing	Software (Image I/O and Feature Ranking) and Hardware	Filtering Algorithm, etc. (Intermediate Software Stages)	Object Detection and Classification Accuracies
Embedded Systems	Software (Image I/O and Feature Ranking) and Hardware	Images/Camera, Algorithm Complexity and Hardware Parameters	Classification Accuracy and Runtime Tradeoffs
Artificial Intelligence	Software (Image I/O and Object Detection) and Hardware	Feature Extraction and Post Processing of Feature Data	Feature Impact and Tracking and Classification Accuracy

The specific parameters that the students control will depend on the material an instructor desires to reinforce. Possibilities include the following:

- Image Processing. A) Students may add specific types of noise and observe the effects on object detection and classification. B) Students may incorporate various types of image preprocessing, noise filtering, and background modeling and observe the effects. C) Students may vary parameters associated with blob formation and observe the effects on detection accuracy.

- Embedded Systems. A) Students may collect images related to a specific surveillance application or environment, and compare the effects to those of standard image sets. B) Students may optimize algorithms for specific platforms and evaluate runtime and memory usage.
- Artificial Intelligence. A) Students may investigate different combinations of object features and observe the effects. B) Students may test the effectiveness of various feature-ranking algorithms. C) Students may implement different types of neural network architectures and observe the effectiveness of the post processing.

Hardware and Images

The resources needed to support such exercises are processing hardware, such as a PC or single board computer, and a dataset from an archived video or created from a camera. The dataset must be chosen carefully to provide an appropriate number of distinct images and to provide appropriately constrained targets. A single uncompressed, 3 color channel, 640 x 480 image represents 900 kB of data. The volume of data contained in video encourages attention to video algorithm complexity and efficient programming techniques. Video data is sufficiently complex and readily available that it need not be contrived, as occasionally must be done to make projects seem realistic. The ability to visualize algorithm output as a video stream imposes a natural real-time processing constraint. The resources for the results in this work are described as follows.

A. Hardware

The algorithms for object detection and feature extraction are tested on two platforms for comparison: (1) a PC running Ubuntu 10.04 and equipped with a 3.4 GHz Pentium D processor and 1 GB of RAM, and (2) a Beagleboard–xm, a single-board computer featuring a 1 GHz Texas Instruments Cortex A8 ARM processor and 512 MB of RAM. The Beagleboard is small (3.25" per side), low-power (4.5 Watts peak), and uses the Angstrom OS, a lightweight Linux distribution for embedded platforms.

B. Images

A combination of new[25] and previously published datasets[26] is used for feature evaluation. Samples of these sequences are shown in Figure 1, and include several background types, camera angles, urban and suburban environments, and distances to targets. The test set contains nearly 11,000 images and represents a total elapsed time of over two hours. Four target classes are depicted: pedestrians, cars, SUVs, and trucks. Over 3000 objects are extracted and analyzed by the pipeline.

Processing Software

We analyze a set of 24 geometric and photometric features for consistency within object classes (pedestrian and vehicular), and for discrimination among object classes. Foreground pixels are detected using adaptive background models and background subtraction, and objects are localized using a two-dimensional foreground density scan. The feature set is computed for each

object, and each feature is evaluated for its utility in object tracking. As we are interested in embedded performance, we consider the execution costs of the processing pipeline and each feature on two platforms.

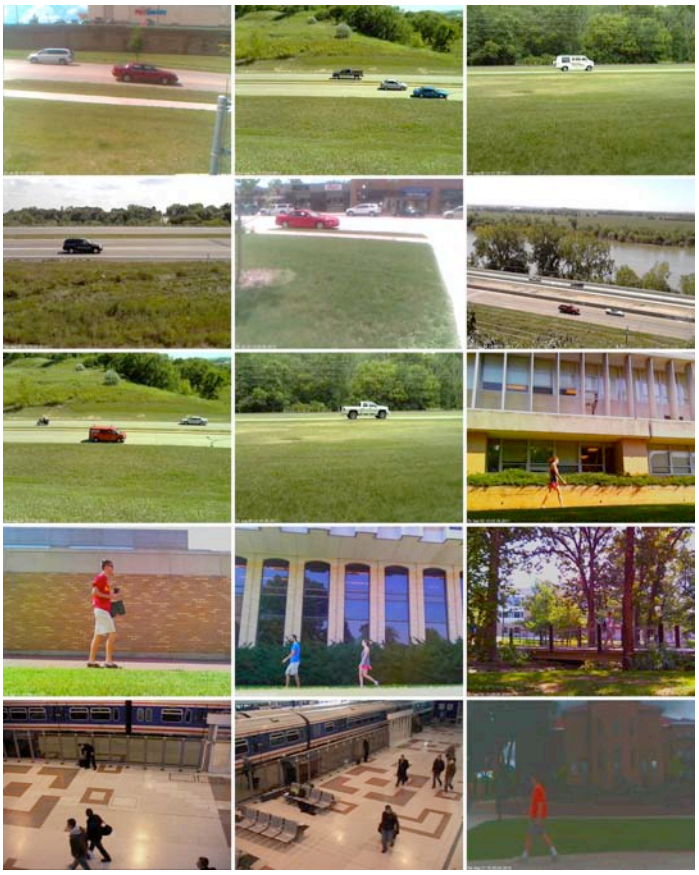

Figure 1. Sample images from the Dataset.

All video processing algorithms here are implemented in C. The OpenCV 2.2 library is used for file I/O and for computing the convex hull, ellipse features, and Hu moments. The remaining features are extracted using custom code. To improve time measurement accuracy, the average feature runtimes were found by computing each feature 100 times.

A. Processing Pipeline

Our processing pipeline consists of a multimodal background model, background subtraction and foreground detection, morphological filtering, and blob detection. The morphological filtering stage produces the silhouette image from which blobs are detected and most object features are extracted. This process is depicted in Figure 2.

A background model is a statistical representation of the uninteresting regions of a scene. Background models are used to detect changes in the scene that correspond to objects or events

of interest. A large number of background models have been proposed[1, 2, 3]; the extensive literature available and the wide range of variations possible make this an excellent topic for students to explore and experiment. Tradeoffs among model complexity, change detection accuracy, rate of adaptation to scene changes, sensitivity to parameter selection, processing time, and memory usage can be considered.

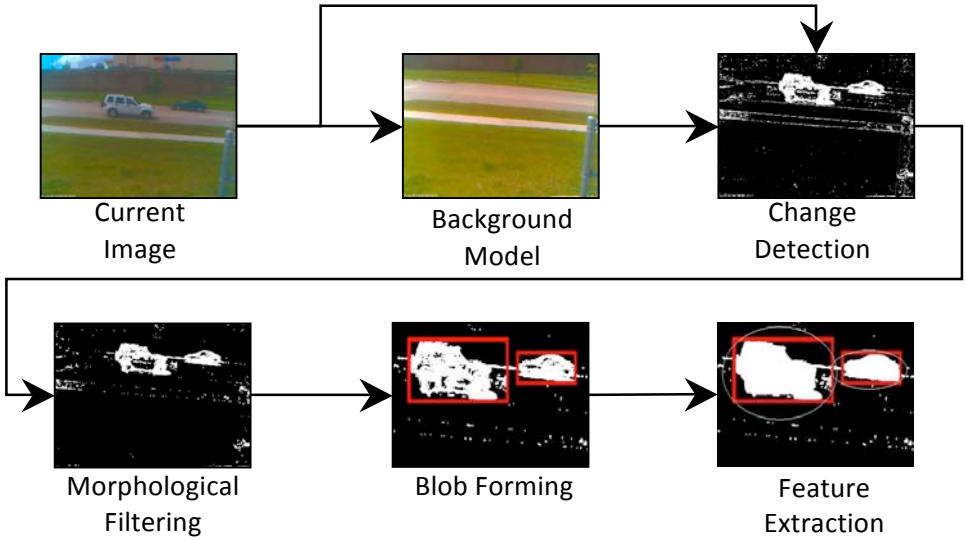

Figure 2. Block diagram of video processing pipeline.

Here, an adaptive, color, multimodal background model called Multimodal Mean[27] is used for change detection. This model represents each pixel as a set of N possible modes (we choose N=4). Each mode consists of a running sum for each color channel and a counter that indicates how many pixels have contributed to the sum, making the mean simple to calculate at any time. Mode data is periodically divided by two to prevent integer overflow and to decay outdated modes. A pixel in the current image is identified as foreground if it differs from all of its background modes by more than a maximum component difference threshold. Vertical and horizontal histograms of foreground pixels are computed to locate blobs.

B. Object Features

Keeping in mind runtime constraints for plausible real-time surveillance systems, we consider popular geometric and photometric features with low computational cost. These features and their symbols are listed in Table 2. Height, width, and aspect ratio pertain to an object's bounding box. Solidity is the ratio of an object's area to its convex hull area. Compactness is the ratio of an object's area to its squared perimeter. Centroid offsets measure the distance from the bounding box center to the centroid. Skewness, kurtosis, and the central moments measure aspects of the grayscale intensity distribution of an object. To compress their dynamic range, the Hu moment features are computed as the $\log_{10}$ of the Hu moments. Perimeter and Euler number are computed locally as discussed by Horn[28]. The major and minor axis lengths, eccentricity and orientation features are computed from an ellipse fitted to an object's silhouette. Figure 3 shows an example of how the bounding box, ellipse, and convex hull are fitted to an object silhouette.

TABLE 2: TESTED FEATURES AND ABBREVIATIONS

Symbol	Feature	Symbol	Feature
H	Box Height	KUR	Kurtosis
W	Box Width	M2	2nd Order Moment
AR	Aspect Ratio (H/W)	M3	3rd Order Moment
A	Object Area	M4	4th Order Moment
PER	Object Perimeter	MJL	Ellipse Major Axis Length
CA	Convex Hull Area	MNL	Ellipse Minor Axis Length
SLD	Solidity (A/CA)	ECC	Ellipse Eccentricity
CMP	Compactness (A/P*P)	OR	Ellipse Orientation
COX	Horiz Centroid Offset	HU1	1st Hu Moment
COY	Vert. Centroid Offset	HU2	2nd Hu Moment
EN	Euler Number	HU3	3rd Hu Moment
SKW	Skewness	HU4	4th Hu Moment

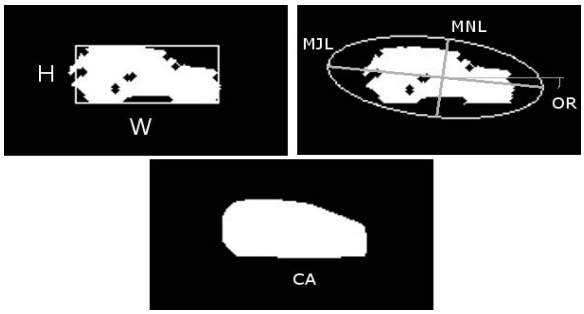

Figure 3. Examples of some geometric features: height and width, ellipse axes and orientation, and convex hull area.

Example Video Analysis Exercises

Consider the suggested exercises for an embedded systems context in which the influence of a specific target type and the influence of hardware are investigated. In particular, the first example gives an analysis of feature consistency for vehicles and pedestrians. The results show which features are preferable for vehicle verses pedestrian tracking. The second example gives an analysis of processing runtime for the two different hardware systems described earlier.

A. Feature Consistency for Tracking

Geometric and photometric features are frequently used to establish object correspondence between frames, often in conjunction with kinematic constraints such as maximum changes in position or velocity. For features to be useful for tracking, they must (1) be stable during the object's traversal of the scene, and (2) vary sufficiently between objects to be distinctive. Fisher's criterion—the ratio of inter-instance variance to intra-instance variance—is useful for quantifying these traits.

To evaluate the tracking utility of the feature set, we measure the relative standard deviations (RSD, standard deviation divided by mean) of the features for each object as it traverses the scene (intra-object variation), and compare this with the RSD of the average feature values for each object observed in a sequence (inter-object variation). The ratios of inter-object variation to intra-object variation are shown in Table 3 for vehicular and pedestrian traffic. Ratios greater than 1 indicate that feature values vary more among different objects of the same class, than among multiple instances of the same object (i.e., the feature value of an object traversing the scene is relatively stable, and distinguishes the object from other objects of the same class). The four highest ratios (corresponding to the best tracking utility) are highlighted. The rigid nature of vehicular traffic makes many of these features potentially useful for tracking. Features describing vertical height and area are most distinguishing among vehicles. The deformable nature of pedestrians renders geometrical features unreliable. Area-related features are again most distinguishing, including compactness (ratio of area to perimeter), as these are least affected by changes in shape and appearance due to walking. Features are 2x to 6x more discriminatory for vehicles than for people. In the absence of additional constraints such as kinematics, the discussed features are generally unsuitable for tracking pedestrians.

Additional examples of surveillance-based artificial intelligence exercises include using feature-ranking algorithms to determine the most effective classification features, and implementing object classifiers or trackers based on those features. The exercise can be adjusted by varying the environments under surveillance, the feature set, and the object classes being considered (e.g. humans, vehicles, bicycles, animals, etc).

TABLE 3: UTILITY OF FEATURES FOR TRACKING

Feature	Vehicle Variance Ratio	Pedestrian Variance Ratio	Feature	Vehicle Variance Ratio	Pedestrian Variance Ratio
H	**7.23**	1.16	KUR	3.29	1.16
W	5.02	0.96	M2	4.06	0.94
AR	2.91	0.71	M3	4.43	0.06
A	**7.06**	**1.27**	M4	3.00	0.93
PER	2.82	0.93	MJL	5.38	1.14
CA	**6.86**	**1.19**	MNL	**8.14**	0.95
SLD	2.19	0.65	ECC	3.40	0.70
CMP	1.84	**1.85**	OR	4.18	0.83
COX	0.94	0.41	HU1	2.05	1.10
COY	0.93	0.48	HU2	1.92	1.06
EN	1.11	0.70	HU3	1.91	0.77
SKW	3.16	**1.56**	HU4	1.98	0.78

B. Runtime Comparison for Different Hardware Platforms

The runtimes for major surveillance processes and feature computations are shown in Table 4. Height, width, and aspect ratio are found as part of the blob finding process. Silhouette area and convex area are computed simultaneously. Photometric moments require a grayscale conversion, which is included in these measurements. Due to the expense of computing the intensity image, the photometric moments are the most expensive features to compute. The PC achieves approximately 15 frames per second, while the Beagleboard reaches 5 frames per second.

TABLE 4: PROCESS AND FEATURE RUNTIMES

Process	PC (ms)	BBxm (ms)
Background Model	31.19	80.87
Change Detection	25.37	61.63
Blob Detection	1.74	5.35
Areas	0.88	3.73
Perimeter	0.10	0.86
Solidity	0.88	3.73
Compactness	0.98	4.60
Euler Number	0.14	1.32
Centroid Offsets	0.05	0.36
Ellipse Features	0.46	10.28
Central Moments	2.78	14.69
Hu Moments	2.67	12.31
Total Time (ms)	67.2	199.7
Frame Rate (Hz)	14.9	5.0

The large amount of video data processed by background modeling and change and blob detection encourages the optimization of these algorithms. Exercises can be constructed that explore the most efficient coding styles for each process on a given platform. In addition, the runtime cost of each feature can be weighted by its accuracy or effectiveness in a given task, resulting in an 'effort' metric.

Discussion and Conclusions

The video processing chain described here can lead to student experiments in several different directions. Background modeling and object detection techniques can be explored in more detail, and sophisticated methods for feature ranking, classification and tracking can all be explored within this framework. The suggested exercises illustrate the educational value of this approach for multidisciplinary courses involving topics such as image processing, embedded systems, and artificial intelligence. Engaging class discussions can arise by allowing students to review literature on their own, implement their techniques of choice, and compare their results.

Prerecorded datasets are necessary in video processing experiments to properly evaluate technique variations against a common baseline. However, a live demonstration platform that captures imagery, processes it in real time, and displays the results offers many benefits. First, students' interests are piqued by seeing their algorithms in action in a complete system—a more satisfying result than studying a few algorithms that fit into a hypothetical larger picture. Second, such a system allows fluid, dynamic interaction with the test environment for exploring unpredicted scenarios. Third, the demonstration motivates students' cognizance of the real-time processing requirement and its consequences. The cost of extra processing becomes more real when one observes frame delays or slow response times. Likewise, the cost of simpler algorithms to improve runtime becomes more apparent when one observes obvious errors in scene analysis. Finally, by visualizing algorithm behavior first-hand, students are prompted to investigate new problems by considering desired behaviors and new applications for these systems.

To these ends, we make available a software package[25] that implements a basic real-time video analysis system. This system exploits the OpenCV library[29] and demonstrates functions for image capture from a USB webcam, manipulation of image color spaces, superposition of text and bounding boxes onto an image, real time input from a keyboard to control process parameters, and live display of images from various points of the analysis process. Manipulation of sensor settings on UVC-compliant cameras is also demonstrated. The analysis process consists of a simple but effective adaptive background model called approximated median[30], change detection based on the maximum component difference criterion, and blob detection. The system can be experimented with as-is, alternative algorithms can be inserted easily, and higher level functions for object tracking or classification can be added. The software is written in C and C++. Instructions are given for configuring a computer with Ubuntu, OpenCV and other necessary packages. The software can also be run on a Windows platform with minimal modification.

Bibliography

1. M. Piccardi. "Background subtraction techniques: A review," Proc. IEEE Int'l Conf. on Systems, Man and Cybernetics: Vol. 4, pp. 3099 – 3104, 2004.
2. R.J. Radke, S. Andra, O. Al-Kofahi, and B. Roysam. "Image change detection algorithms: A systematic survey," *IEEE Transactions on Image Processing*, Vol. 14, No. 3, pp. 294–307, 2005.
3. S. Cheung and C. Kamath. "Robust techniques for background subtraction in urban traffic video," *Proc. SPIE,* Vol. 5308, 881 (12 pages), 2004. http://dx.doi.org/10.1117/12.526886
4. O. Masoud and N.P. Papanikolopoulos. "A novel method for tracking and counting pedestrians in real-time using a single camera," *IEEE Transactions on Vehicular Technology*, Vol. 50, No. 5, pp. 1267–1278, 2001.
5. A.M. Cretu, E.M. Petriu, P. Payeur, and F.F. Khalil. "Deformable object segmentation and contour tracking in image sequences using unsupervised networks," Proc. of the 7th Canadian Conf. on Computer and Robot Vision (CRV 2010), pp. 277-84, 2010.
6. V. Syrris and V. Petridis. "Statistical descriptors for human actions classification," 2009 17th Mediterranean Conf. on Control and Automation (MED), pp. 412-15, 2009.
7. F. Serratosa, N.A. Gomez, and R. Alquezar. "Combining neural networks and clustering techniques for object recognition in indoor video sequences," Proc. 11th Iberoamerican Congress in Pattern Recognition, CIARP 2006. (Lecture Notes in Computer Science Vol. 4225), pp. 399–405, 2006.
8. Y. Kong and L. Wang. "Moving target classification in video sequences based on features combination and SVM," Proc. 2010 Int'l Conf. on Computational Intelligence and Software Engineering, CiSE 2010, 2010
9. Y. Gurwicz, R. Yehezkel, and B. Lachover. "Multiclass object classification for real-time video surveillance systems," *Pattern Recognition Letters*, Vol. 32, No. 6, pp. 805–815, April 2011.
10. H. Li and J. Cao. "Detection and segmentation of moving objects based on support vector machine," Proc. 3rd Int'l Symposium on Information Processing, ISIP 2010, pp. 193–197, 2010.
11. A. Sun, M. Bai, Y. Tan, and J. Tian. "SVM based classification of moving objects in video," Proc. SPIE 7496, 749607, 2009. http://dx.doi.org/10.1117/12.832622
12. S.S. Ali, M.F. Zafar, and M. Tayyab. "Detection and recognition of human in videos using adaptive method and neural net," Proc. 2009 Int'l Conf. of Soft Computing and Pattern Recognition, pp. 604-609, 2009.
13. M.K. Hu. "Visual pattern recognition by moment invariants," *IRE Transactions on Information Theory*, Vol. IT-8, pp. 179–187. 1962.
14. A. Gepperth, J. Edelbrunner, and T. Bucher. "Real-time detection and classification of cars in video sequences," Proc. 2005 IEEE Intelligent Vehicles Symposium, pp. 625-31, 2005.
15. J. Li, C. Shao, W. Xu, and J. Li. "Real-time system for tracking and classification of pedestrians and bicycles," Transportation Research Record, No. 2198, pp. 83-92, December 1, 2010.
16. S.E. Watkins, R.J. Stanley, A. Gopal, and R.H. Moss. "Surveillance of pedestrian bridge traffic using neural networks," *Proc. SPIE*, Vol. 7292, 72922Q (12 pages), 2009. http://dx.doi.org/10.1117/12.815523

17. A. Yilmaz, O. Javed, and M. Shah. "Object tracking: A Survey," *ACM Computing Surveys*, Vol. 38, No. 4, pp. 1–45, 2006.
18. M.S. Rahman, A. Saha, and S. Khanum. "Multi-object tracking in video sequences based on background subtraction and SIFT feature matching," Proc. Fourth Ann. Int'l Conf. on Computer Sciences and Convergence Information Technology, pp. 457–462, 2009.
19. S. Zhang, H. Yao, and P. Gao. "Robust object tracking combining color and scale invariant features," *Proc. SPIE*, Vol. 7744, 77442R (8 pages), 2010. http://dx.doi.org/10.1117/12.863844
20. J. Durand and S. Hutchinson. "Real-time object tracking using multiresolution critical point filters," Proc. IEEE Int'l Conf. on Robotics and Automation (ICRA 2003), Vol. 2, pp. 1682–1687, 2003.
21. T. Dharamadhat, K. Thanasoontornlerk, and P. Kanongchaiyos. "Tracking object in video pictures based on background subtraction and image matching," Proc. IEEE Int'l Conf. on Robotics and Biomimetics (ROBIO 2009), pp. 1255–1260, 2009.
22. L. Zhang and R. Wu. "Tracking object using object-strips color feature," Proc. WASE Int'l Conf. on Information Engineering (ICIE 2010), pp. 212–215, 2010.
23. S. Apewokin, B. Valentine, R. Bales, L. Wills, and S. Wills. "Tracking multiple pedestrians in real-time using kinematics," Proc. IEEE Conf. on CVPR Workshops, pp. 1–6, 2008.
24. A. Tavakkoli, M. Nicolescu, and G. Bebis. "A spatio-spectral algorithm for robust and scalable object tracking in videos," Int'l Symposium on Visual Computing: 2010. G. Bebis et al. (Eds.): ISVC 2010, Part III, LNCS 6455, pp. 161–170. Springer, Heidelberg (2010).
25. Missouri University of Science and Technology, "Smart Engineering: Educational Resources," (2012). Available WWW http://smarteng.mst.edu/educationalresources/.
26. PETs 2006 dataset. Available ftp://ftp.pets.rdg.ac.uk/pub/PETS2006/.
27. S. Apewokin, B. Valentine, D. Forsthoefel, D.S. Wills, L.M. Wills, and A. Gentile. "Embedded real-time surveillance using multimodal mean background modeling," in *Embedded Computer Vision* (B. Kisacanin, S. Bhattacharyya, and S. Chai, eds., (Springer-Verlag, London, U.K., 2009), Chap. 8, pp. 163–175.
28. B.K.P. Horn. *Robot Vision* (McGraw-Hill, New York, NY, U.S., 1986), pp 71–77.
29. "OpenCV," (2012). Available WWW http://opencv.willowgarage.com/wiki/.
30. N.J.B. McFarlane and C.P. Schofield. "Segmentation and tracking of piglets in images," *Machine Vision and Applications*, Vol. 8, No. 3, pp. 187–193, 1995.

Appendix: Top-level Processing Code

This Appendix shows the top-level source code of the video analysis process, and demonstrates the flow of data from loading images, detecting changes between the current image and the background model, performing binary erosion to clean the binary image, detecting blobs (high spatial concentrations of foreground pixels) in the binary image, and computing the feature set for each blob. More detailed code for these operations can be found at the website[25].

```
// Main loop for processing interesting part of sequence for (N = Start + Preamble; N <= End; N++)
{
    sprintf(Path, "%s/%s/%05d.jpg", SEQ_DIR, Seq, N);        // Get the directory containing the stored images.
    TFN = N-(Start+Preamble);                                // Calculate the true frame number (in case Start is not 0).
    load_success = Load_Image(Path, Frame);

    /* Find foreground pixels by doing change detection between the current image and the background model.  */
    MMM_Change_Detection_1(mmm_model, Frame, sil_image_00, mmm_predom_image, Width, Height,
            MCDth, Cth);

    bin_erosion(sil_image_00, sil_image_01, Width, Height, Stel, 0x5, 0x5, 1);   // Morphological erosion.
    bin_frame->imageData = (char *) sil_image_01;
```

```
      /* Find blobs in the binary silhouette image. Detected blobs are stored in the Contacts0 linked list. */
blob_count1 = BlobFind2D(sil_image_01, 0, Width-1, 0, Height-1, Width, RowSumTh, ColSumTh, MinSizeX,
            MinSizeY, GapSize, Contacts0, 1);


/*  Make grayscale Foreground image with Background zeroed out
       This grayscale image is used to calculate OpenCV Moments and HuMoments
*/
for(yy=0; yy<Height; yy++)     {
      for(xx=0; xx<Width; xx++)      {
            SilIndex1 = yy*Width+xx;
            if(sil_image_01[SilIndex1] == 255){
                  I = 3*(SilIndex1);
                  gray_image[SilIndex1] = (Frame[I]+Frame[I+1]+Frame[I+2])/3;
            }
            else {     gray_image[SilIndex1] = 0; }
      }
}
gray_frame->imageData = (char *) gray_image;

/* At this point, we have preliminary foreground silhouettes and localized blobs. */

cvSet(convex_frame, cvScalar(0));
convex_image = (unsigned char *) convex_frame->imageData;

ContactNum = Contacts0->Get_Size();

/* Iterate over contact list; calculate features for each contact.   */
/* Each node of the Contact0 linked list is a structure that contains all of the feature information for a target. */
if(ContactNum > 0)
{
      temp = Contacts0->Get_Head();
      while(temp != NULL)
      {
            box_height = (1+ temp->maxy - temp->miny);
            box_width = (1+ temp->maxx - temp->minx);
            box_aspect_ratio = (float)box_height/(float)box_width;

            Get_EulerNumber(sil_image_01, temp, Width, Height, euler_number);
            Get_Perimeter(sil_image_01, temp, Width, perimeter);
            Get_CentroidOffset(sil_image_01, temp, Width, centroid_offset_x, centroid_offset_y);
            Get_Convex_Hull(bin_frame, convex_frame, convex_image, temp, Width, sil_area, hull_area);

            // Simple ratio parameters
            solidity = (float)sil_area/(float)hull_area;
            compactness = (float)12.56636*sil_area/(perimeter*perimeter);

            Get_Ellipse(bin_frame, temp, Width, ell_center_x, ell_center_y, ell_mjr_axis, ell_mnr_axis,
            ell_orientation, ell_eccentricity);

            Get_Moments(sil_image_01, Frame, temp, Width, obj_M2, obj_M3, obj_M4, obj_skew, obj_kurt);
            Get_OCVMoments(gray_frame, temp, cv_moments, cv_hu);
```

```
            // Print the features of each target to file.
            // "FN, x, y, H, W, AR, A, PER, CA, SLD, CMP, COX, COY, EN, S, K, M2, M3, M4, MJL, MNL,
            ECC, OR, HU1, HU2, HU3, HU4"
            fprintf(Report,
            "\n%d,%d,%d,%d,%d,%f,%d,%f,%d,%f,%f,%f,%f,%d,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,
            %f", N, temp->xpos, temp->ypos, box_height, box_width, box_aspect_ratio, sil_area, perimeter,
            hull_area, solidity, compactness, centroid_offset_x, centroid_offset_y, euler_number, obj_skew,
            obj_kurt, obj_M2, obj_M3, obj_M4, ell_mjr_axis, ell_mnr_axis, ell_eccentricity, ell_orientation, -
            log10(cv_hu->hu1), -log10(cv_hu->hu2), -log10(cv_hu->hu3), -log10(cv_hu->hu4));

            temp = temp->next;
        }
    }



    /* Draw bounding boxes around blobs  */
    DrawBlobBox(sil_image_01, Width, Height, Contacts0, 1);

    bin_frame->imageData = (char *) sil_image_01;
    Contacts0->ClearList();

    // Update background model with current frame at end of loop, after frame has been processed.
    Process_Frame_BG(mmm_model, Frame, Width, Height, MCDth, CTH);

    // Decimate Multimodal Mean model periodically to prevent overflow, and to decay outdated modes.
    if((TFN % DECRATE) == 0)   {
        Decimate_BGM(mmm_model, CTH, Width*Height);            }
}
```

Biographical Information

DR. M. RYAN BALES is a Research Engineer with the Sensors and Electromagnetic Applications Lab of Georgia Tech Research Institute. His interests are in video and image processing, radar, and digital systems. He is a member of IEEE and HKN. His Ph.D. is from the Georgia Institute of Technology (2011). Contact: mrbales@ieee.org.

DR. STEVE E. WATKINS is Professor of Electrical and Computer Engineering at Missouri University of Science and Technology, formerly the University of Missouri-Rolla. His interests include educational innovation. He is active in IEEE, HKN, SPIE, and ASEE including service as the 2009 Midwest Section Chair. His Ph.D. is from the University of Texas at Austin (1989). Contact: steve.e.watkins@ieee.org