University of Nevada, Reno

Robust Event Detection and Retrieval in Surveillance Video

A thesis submitted in partial fulfillment of the
requirements for the degree of Master of Science
in Computer Science and Engineering

by

Jin Jiang

Dr. Mircea Nicolescu, Thesis Advisor

May, 2014

University of Nevada, Reno
Statewide • Worldwide

THE GRADUATE SCHOOL

We recommend that the thesis
prepared under our supervision by

JIN JIANG

entitled

Robust Event Detection And Retrieval In Surveillance Video

be accepted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

Mircea Nicolescu, Ph.D., Advisor

Monica Nicolescu, Ph.D., Committee Member

Mark Pinsky, Ph.D., Graduate School Representative

Marsha H. Read, Ph. D., Dean, Graduate School

May, 2014

Abstract

We developed a robust event detection and retrieval system for surveillance video. The proposed system offers vision-based capabilities for the detection and tracking of various objects of interest, and can recognize events such as: 1. a person with certain attributes being present in the scene; 2. two people meeting; 3. people carrying bags; 4. bags being dropped; 5. bags being stolen; 6. bags being exchanged; 7. two people handshaking; 8. one person's pointing gesture. We use an improved adaptive Gaussian mixture model for background modeling and foreground detection; a connected component labeling algorithm is then employed to label the foreground pixels. A Kalman filter approach is used to build models for the entities of interest (people and bags), which is combined with color histograms for tracking. We use shape symmetry analysis and color histograms to detect people carrying bags. Our experiments demonstrate the ability to search for instances of events according to specific attributes in large video sequences.

Dedications

For my parents.

Acknowledgements

I would like to take this opportunity to thank my advisor Dr. Mircea Nicolescu for his advice and encouragement. This work would not have been possible without his help, direction and especially his patience, when I took longer than expected to finish things.

Thanks to Dr. Monica Nicolescu and Dr. Mark Pinsky for accepting to serve on my thesis committee.

I would also like to thank all my colleagues in Computer Vision Lab.

I am grateful to all of you.

Table of Contents

List of Tables

List of Figures

Chapter 1. Introduction

Intelligent video surveillance is an important research topic in computer vision and has been widely used in many applications nowadays. Benefitting from the development of computer vision (and related fields) algorithms and computational capabilities, such as the expansion of massive digital storage or the more advanced video compression and intelligent learning algorithms, video surveillance systems are more and more powerful. For example, in an airport, an automated video surveillance system may be able to detect potential threats such as leaving a bag unattended; in a store, video surveillance may be used to detect and track people and/or analyze customer flow. For families with little children, such systems may be able to detect potentially dangerous activities that could hurt a child, and alert parents to prevent them. In traffic applications, video surveillance systems could detect and track vehicles and pedestrians in order to optimize the timing of traffic lights.

The main goal of automated, video-based surveillance systems is to understand the activities in a scene as captured by a video stream, and give the (high-level) analysis results to human operators. This requires real-time processing, robust algorithms for the detection and tracking of objects of interest (such as vehicles, humans, or objects they are interacting with) and activity recognition and/or classification. In this thesis we developed a system for automated recognition and retrieval of events from surveillance video.

The rest of the thesis is organized as follows. Chapter 2 discusses the previous work in video surveillance systems and their components: background modeling and foreground detection, blob detection and tracking, human activity recognition. Chapter 3 gives a detailed description of our approach. Chapter 4 provides the experimental

results of our approach. Chapter 5 concludes this work and discusses potential directions of future work.

Chapter 2. Previous Work

Most video surveillance systems have to address the following general sequence of tasks: first, they have to detect the moving objects from either a static or moving camera (with static cameras more often being used), such as people and objects they interact with, or vehicles. In this task, background modeling and foreground detection is a common and popular approach. Background modeling involves learning a dynamic model that represents the background of a scene, which should not contain moving objects. Each current image is then subtracted from the background image and the difference is used to segment out the foreground image, that typically corresponds to the objects of interest present in the scene. Subsequently, the system should be able to track the detected objects - i.e., continuously determine the location, shape, velocity and other information about these objects, such as distance from the camera (depth). Since the foreground regions are often moving objects, tracking them is relevant as both their temporal and spatial characteristics can be useful for recognizing activities in the scene. Next, the system should be able to recognize various kinds of activities (events), either predefined or generic. For example, in traffic applications, video surveillance systems could be designed to detect vehicles, their license plates and events such as a vehicle running a red light; in a security system, it would be relevant to detect people and their activities, such as interactions between humans or with various objects. Video surveillance systems have a wide range of applications and can have many other components and functionalities as well. In the following subsections, we will discuss the tasks of a typical video surveillance system, with examples of previous work for each.

2.1 Background Modeling and Foreground Detection

Moving image regions are usually the entities most relevant in a video. In order to detect moving objects such as walking people in a sequence of images, one of the most frequently used approaches is background modeling and foreground detection, which is also an important first step in many other computer vision applications. The main goal of the background modeling and foreground detection is to classify each pixel in an image as background of foreground, based on a previously acquired model of the background (that does not contain any moving objects). Figure 1 is an example of background modeling and foreground detection.

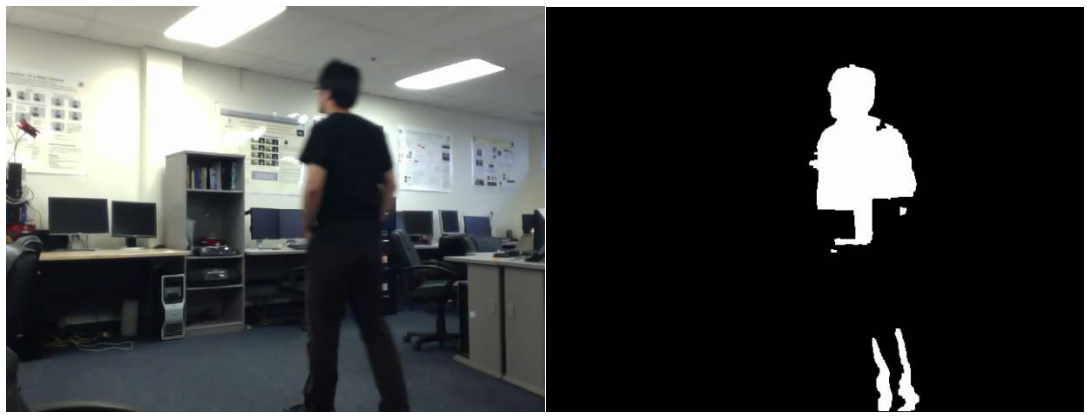


Figure 1. Original image (left); detected foreground (right).

Background modeling can be difficult since the background could be affected by many factors; for example, the background might exhibit small motions or even large changes, such as those induced by rain or snow, waving tree branches, or moving water. Illumination conditions can also have an effect, for example in an outdoor environment, the light from the sun gradually changes with time, or indoors the flickering of artificial lighting may change the illumination suddenly, and the shadows of moving objects can be a problem as well since they will be detected as foreground. The presence of random noise in images is also a difficult issue. Figure 1 illustrates a problem with the person's

lower body (which should be part of the foreground) where pixel values similar to the background result in a failure to detect this as part of the foreground. A good background model should be able to address such complex issues; otherwise foreground detection would fail and thus affect the subsequent tasks and the whole system performance.

Many algorithms have been developed for background modeling and foreground detection in recent years, with [1] being a good survey paper of such techniques. Here we give a brief introduction to some traditional algorithms.

A very simple method is to use the median [2] or the average [3] of several frames over time as the background image. By subtracting the background image from the current image and thresholding the absolute value of the difference we can classify the pixels as foreground of background:

$$\text{If } |I(\mathrm{x},\mathrm{y}) - background(\mathrm{x},\mathrm{y})| > Threshold \text{ then foreground pixel}$$

$$\text{Otherwise background pixel}$$

These methods are based on a single threshold, which is very sensitive to noise and cannot adapt to dynamic backgrounds, thus failing in the presence of changing and noisy complex backgrounds.

Statistical background modeling methods have been very popular since 1990s, when many background modeling techniques have been developed based on statistical approaches. Pfinder [4] assumes a single Gaussian model for each pixel independently in the image. This method needs a training period to learn the background model where there should be no foreground object in the scene. The mean μ and the variance σ of the model at time t+1 will be updated recursively based on their values at time t and the current pixel intensity value as follows:

$$\mu_{t+1} = \alpha F_{t+1} + (1-\alpha)u_t$$

$$\sigma^2_{t+1} = \alpha(F_{t+1} - \mu_t)^2 + (1-\alpha)\sigma^2_t$$

where F_{t+1} is the pixel intensity value and α is the learning rate. In the current image, each pixel value will be compared with its model, to check if the pixel intensity is within a threshold of the mean:

$$\left|F_{t+1} - \mu_{t+1}\right| < Th * \sigma$$

then the pixel is considered as a background pixel, otherwise a foreground pixel. Several improvements have been proposed based on the single Gaussian model. For example, the HSV color space is used instead of RGB space for modeling the background in [5]. Since HSV separates the intensity and chromatic information, using HSV can improve the robustness of the model to illumination changes.

The single Gaussian model can be used for situations where changes of background illumination and motion are small and gradual. However, this model is still not powerful enough to handle more complex situations, such as shadows and waving trees.

Based on the single Gaussian model, [7] proposed to use a Gaussian mixture model (GMM) to model background that can be more dynamic and complex. Similar to the single Gaussian model, GMM assumes that each pixel can be independently modeled by multiple Gaussian components based on the recent history of that pixel. The probability of observing the current pixel value is:

$$\mathrm{P}(X_t) = \sum_{i=1}^{K} \omega_{i,t} \cdot \eta(X_t\,,\ \mu_{i,t}\,, \Sigma_{i,t})$$

where K is the number of Gaussian components, which can be specified by the user, with 3 to 5 typically used. Parameters $\omega_{i,t}$, $\mu_{i,t}$, $\Sigma_{i,t}$ are estimates of the weight, the

mean value and the covariance matrix of the ith Gaussian component in the mixture at time t, respectively, and η is a Gaussian probability density function:

$$\eta(X_t\ ,\ \mu\ ,\Sigma) = \frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(X_t-u_t)^T\Sigma^{-1}(X_t-u_t)}$$

In [7], the covariance matrices are constrained to be diagonal by assuming that the RGB color components are independent. Thus the distribution of recently observed values of each pixel is modeled by a mixture of K Gaussians.

Regarding the initialization of the parameters (weights, mean, covariance), Stauffer et al. [7] claimed that using an expectation maximization algorithm on each pixel would be expensive. Instead, an online K-means approximation is used. When a new frame comes at time t+1, each new observed pixel value X_{t+1} is checked against the existing K Gaussians distribution until a match is found, where a match is defined as:

$$\left(\left(X_{t+1}-u_{i,t}\right)^T\Sigma_{i,t}^{-1}\left(X_{t+1}-u_{i,t}\right)\right)^{1/2} < k\sigma$$

where k is set to 2.5 in their paper. If a match is found, the pixel is then classified as the same group with that distribution, and the parameters of that Gaussian distribution are updated as follows:

$$\omega_{i,t+1} = (1-\alpha)\omega_{i,t} + \alpha$$

$$\mu_{i,t+1} = (1-\rho)u_{i,t} + \rho\cdot X_{t+1}$$

$$\sigma_{i,t+1}^2 = (1-\rho)\sigma_{i,t}^2 + \rho\left(X_{t+1}-\mu_{i,t+1}\right)\cdot\left(X_{t+1}-\mu_{i,t+1}\right)^T$$

where α is a constant learning rate, $\rho = \alpha\cdot\eta\left(X_{t+1}\ ,\ \mu_i\ ,\Sigma_i\right)$.

More details about these update equations can be found in [1] [7] [41]. If none of the K distributions is matched with the current sample, the least probable distribution

k is replaced by a new distribution (Gaussian component) with the current value as its mean value, an initially high variance, and low prior weight.

The Gaussian mixture models have been well studied, illustrating their advantages and disadvantages. For example, the number of Gaussians components is constant and should be pre-determined manually; the initialization/training stage requires a quite large number of frames without any moving objects. Many improvements have been proposed to address the drawbacks of GMM. For example, in our system we used an improved adaptive GMM for background modeling [39].

Using multiple Gaussian probability distributions gives the model capability to be adaptive for scenes where there are slow multimodal variations. However, when dealing with dynamic backgrounds like waving trees, this model does not perform well. In order to deal with backgrounds that have large areas with small motions, Elgammal et al. [8] proposed a non-parametric model approach for background modeling. They use a kernel density function K to model each pixel based on the N most recent samples of pixel intensity values, as follows:

$$P(x_t) = \frac{1}{N}\sum_{i=1}^{N} K(x_t - x_i)$$

where K(.) is the kernel density function. A standard Gaussian distribution is usually used for K(.), thus we can rewrite the above formula as:

$$\mathrm{P}(x_t) = \frac{1}{N}\sum_{i=1}^{N} \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} e^{-\frac{1}{2}(x_t - x_i)^T \Sigma^{-1}(x_t - x_i)}$$

Similar to GMM, Elgammal et al. [8] constrained the covariance matrices to be diagonal by assuming that the RGB color components are independent, thus the probability density function can be further rewritten as follows:

$$\mathrm{P}(x_t) = \frac{1}{N}\sum_{i=1}^{N}\prod_{j=1}^{d}\frac{1}{\sqrt{2\pi\sigma_j^2}}e^{-\frac{1}{2}(x_t - x_i)^T/\sigma_j^2}$$

For each new frame, each pixel x_t is classified as a background or foreground pixel based on its probability density function $\mathrm{P}(x_t)$ compared with a threshold T. Elgammal used two background models: a short term model, which consists of the most recent N background sample values; and a long term model, which consists of N pixels sampled from a larger window over time. Combining the two models for foreground detection has the advantage to eliminate the persistent false positives detection from the short term model and extra false positives detection that occur in the long term model results [1][8].

Use kernel density estimation for background modeling can handle fast changes in backgrounds. However, the algorithm is not computationally efficient. Several improvements have been proposed based on KDE [42] [43] [44], we refer the reader to those papers for further details.

In addition to statistical background modeling, many other algorithms have been developed for background modeling. For example, Culbrik et al. [45] proposed to train a neural network where the background area is represented by the weights. Messelodi et al. [46] used a Kalman filter to estimate the background, where if a pixel is significantly different from its predicted value by the filter, then it is considered foreground. Clustering methods have also been used in background modeling. K-means [47] and Codebook [6] have been used to classify the samples into groups.

2.2 Blob Extraction

Background modeling and foreground detection determine whether a pixel is a background or a foreground pixel as a binary mask; however, it does not tell how pixels are related to each other in morphology. Blob Extraction (alternatively called connected component labeling) is an application in graph theory, which has been widely used in computer vision and pattern recognition to detect connected regions. A component labeling algorithm finds all connected components and assigns a unique label to all points in the same component. It is generally performed on binary images. Figure 2 is an example of a connected component labeling (figure from [25]).

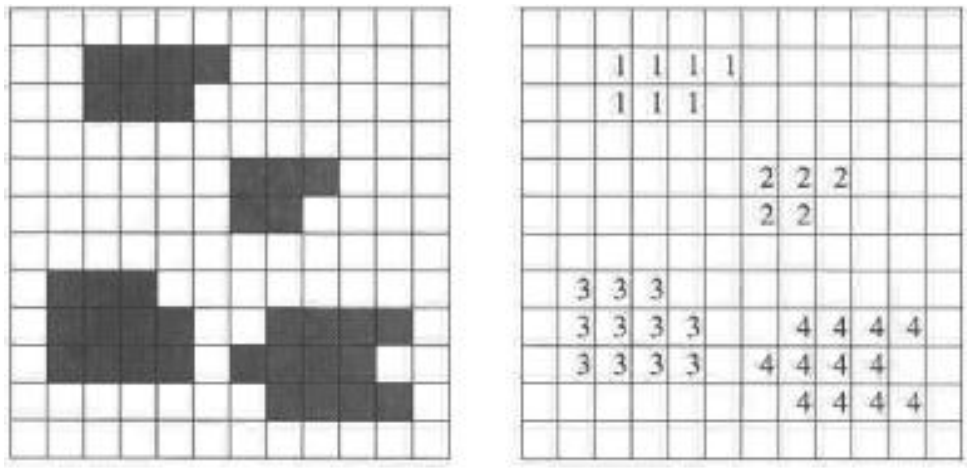


Figure 2. Example of a connected component labeling

Several algorithms for connected components labeling have been proposed. Suzuki et al. [9] classified them into four categories: (1) algorithms [10,11] that perform multiple passes over the data; (2) algorithms [12 ,14,15] that perform two passes over the data; (3) algorithms [16,17,18,19] that use hierarchical tree structures to represent the image; (4) parallel algorithms [20,21,22] for parallel machine models. All these algorithms use a scanning operation, which checks whether the surrounding pixels of a current pixel have been assigned a label already or not to determine the label for the current pixel.

The idea of the algorithms in the first category is simple and easy to implement. It simply repeats the scanning through the data back and forth until all pixels have a label. Assumes that all the foreground pixels have the value 255 (white), and all the background pixels have value 0 (black). A simple recursive labeling algorithm is to first scan the image to find an unlabeled white pixel and assign it a new label L, then recursively assign L to all of its white neighbors, and stop if there are no more unlabeled white pixels around. This procedure is repeated until no more white pixels are unlabeled in the whole image. The algorithm does not require extra space, however, it requires a very high number of iterations and is not computation efficient.

An easy to implement sequential approach in category 2 is called the two-pass algorithm [9], which requires two passes over the binary image: one pass to assign provisional labels to each pixel and record the equivalence information among pixels in a table array; the second pass to replace each provisional label by the smallest label of its equivalence class, which is usually performed by using a search algorithm such as the Union-Find algorithm with pointer based rooted trees [16,23, 24]. A simple conventional two-pass connected component labeling algorithm [25] is described as follows:

(1) Scan the image from left to right, top to bottom.

(2) If the pixel is foreground (white) and unlabeled, then:

    a) If only one of its upper and left neighbors has a label or both have the same label, then copy the label.

    b) If its upper and left neighbors have different labels, then copy the upper's label and enter the labels in the equivalence table as equivalent labels.

c) Otherwise assign a new label, which is the current label value increased by 1 to this pixel and enter this label in the equivalence table with the new label value.

(3) If there are no more foreground pixels around the current pixel to consider, go to step 2.

(4) Find the lowest label for each equivalent set in the equivalence table.

(5) Scan the image and replace each label by the lowest in its equivalent set.

In the conventional two-pass algorithm, the number of iterations depends on the geometric complexity of the image data. Suzuki et al. proposed a linear-time algorithm for labeling connected components in binary images based on sequential local operations [9]. A one-dimensional table called the label connection table T, which memorizes label equivalences information, is used for uniting equivalent labels successively during the operations in forward and backward raster directions. Suzuki's algorithm assigns the provisional labels as follows:

The first scan:

$$g(x,y) = \begin{cases} F_B, & if\ b(x,y) = F_B \\ \mathrm{m}, (m = m + 1), & if\ \forall\{i,j \in M_s\} g(x-i, y-j) = F_B \\ T_{min}(x,y), & otherwise \end{cases}$$

$$T_{min}(x,y) = min[\{T[g(x-i,y-j)] | i,j \in M_s\}].$$

where F_B means the pixel is a background pixel, m is the label value, which is initialized to 1, M_s indicates the region of the mask. The label connection table is updated simultaneously with the assignment of the provisional labels as follows:

$$\begin{cases} nonoperation, & if\ b(x,y) = F_B \\ \mathrm{T}[m] = \mathrm{m}, & if\ \forall\{i,j \in M_s\} g(x-i, y-j) = F_B \\ T[g(x-i,y-j)] = T_{min}(x,y), & if\ g(x-i,y-j) \neq F_B \end{cases}$$

After the first scan, the backward scan and the forward scan are performed alternately. This algorithm is based on only sequential local operations, and it has an execution time directly proportional to the number of pixels in connected components in the image.

2.3 Object Tracking

Object tracking is an important task in computer vision. Its task is to estimate the trajectory of an object in the image sequence as the object moves in the scene. In other words, a tracker assigns consistent labels to the detected objects in different frames of a video. Additionally, depending on the tracking domain, a tracker can also provide object-specific information, such as orientation, scale, area, or shape. A tracker usually consists of three parts [13]: (1) an appearance model (such as color information) which measures the similarity of the object between the previous and the current frame; (2) a motion model which connects the locations of the object; and (3) a search strategy for finding the most likely location.

Tracking objects can be difficult due to many reasons. When capturing the video, some information is lost by projecting the 3D world to 2D images. Various entities (for example people) can have a non-rigid complex motion as well as a non-rigid shape. Object occlusion is significant in real situations, which makes tracking even more difficult. Illumination changes and imaging noise also pose problems for object tracking.

Given these difficulties, we can simplify tracking by imposing some constraints on the motion and/or appearance of objects. It is possible to assume that the object motion is smooth with no abrupt changes. Prior information such as the number, the

size, the appearance and the shape of the objects can also be helpful in simplifying the problem. Numerous approaches for object tracking have been proposed. Here we give a brief description of these methods.

2.3.1 Kalman Filter Tracking

The Kalman filter [28] was proposed in 1960 as a recursive solution to the discrete data liner filtering problem. Since then, the Kalman filter has been used in many domains; in computer vision, the Kalman filter has been employed for object tracking and autonomous or assisted navigation. From a mathematical viewpoint, the Kalman filter and its extensions are an estimator which predicts and corrects the state of either linear or non-linear processes. This section will describe how the Kalman filter can be used in the problem of object tracking.

Denote X_k as the state vector at time k, Z_k as the measurement (or the observed) vector at time k, then the Kalman filter equation is:

$$X_k = AX_{k-1} + BU_k + W_k$$

$$Z_k = HX_k + V_k$$

where A is the state transition model applied to the previous state X_{k-1}, U is an optional control vector to the state X, B is the optional control input model applied to U, W is the process noise which is usually assumed to be Gaussian noise with mean 0, H is the observation model which maps the true state vector to the observed data vector, and V is the observation noise which is assumed to be Gaussian noise with mean 0 as well. The Kalman filter consists of two phases: the first is predict, which uses the previous state vector to predict the current state vector; the second phase is update,

which combines the predicted current state vector and current observation information to refine the state estimate. Below is how the model is predicted and updated:

Predicted state estimate: $\hat{X}_{k|k-1} = A_k \hat{X}_{k-1|k-1} + B_k U_k$

Predicted estimate covariance: $P_{k|k-1} = A_k P_{k-1|k-1} A_k^T + Q_k$

where P is the posterior error covariance matrix and Q is the covariance of Gaussian noise of W. The update phase is as follows:

Measurement residual: $\hat{Y}_k = Z_k - H_k \hat{X}_{k|k-1}$

Residual covariance: $S_k = H_k P_{k|k-1} H_k^T + R_k$

Optimal Kalman gain: $K_k = P_{k|k-1} H_k^T S_k^{-1}$

Updated state estimate: $\hat{X}_{k|k} = \hat{X}_{k|k-1} + K_k \hat{Y}_k$

Updated estimate covariance: $P_{k|k} = (I - K_k H_k) P_{k|k-1}$

The Kalman filter has been used in object tracking, where the object's geometric information such as location, shape, size, center can be seen as the state in the Kalman filter, and the observed state of the object is used to update the equation.

2.4 Bag Detection

While detecting and tracking people is an important task in video surveillance systems, detecting and tracking objects that people interact with is also important and even more challenging. In this section, we focus on the detection and tracking of bags that people carry in an image sequence.

The W4 system [31] use an approach called backpack [32], which combines two basic observations to analyze people carrying objects. The first observation is that human body shape is considered to be symmetric; the second is the periodic motion

exhibited by people walking. Backpack utilizes each person's silhouette, which is represented by a projection histogram. The vertical and horizontal projection histograms are computed by projecting the binary foreground region onto axes perpendicular to and along the major axis, respectively. Backpack computes a center axis for each person's silhouette model by using Principal Component Analysis [33]. Figure 3 is an example of the silhouette model, the center axis. Figure 4 is an example of the projection histogram (figure 3, 4,5 are from [31]).

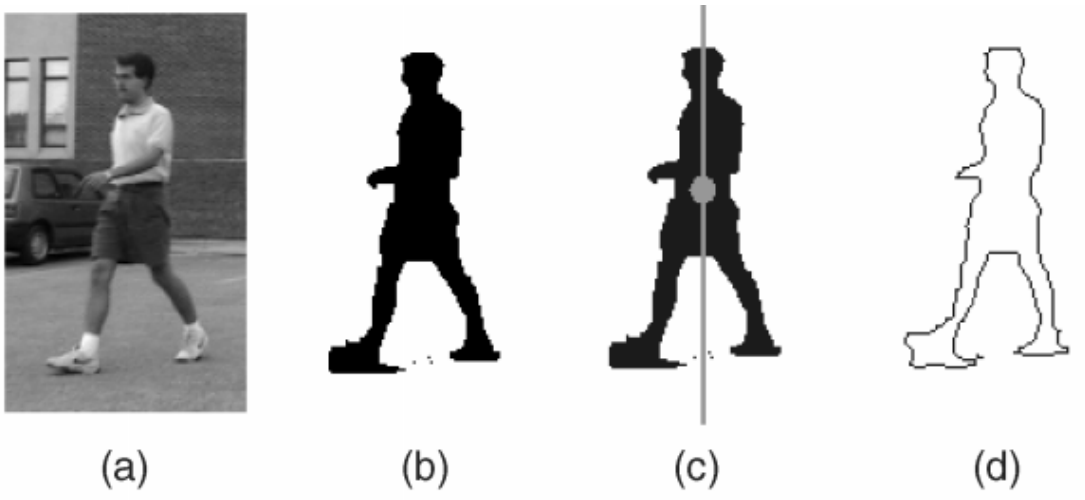


Figure 3.Silhouette based shape feature:(a) input image, (b) detected foreground, (c)major axis, (d)contour of its boundary

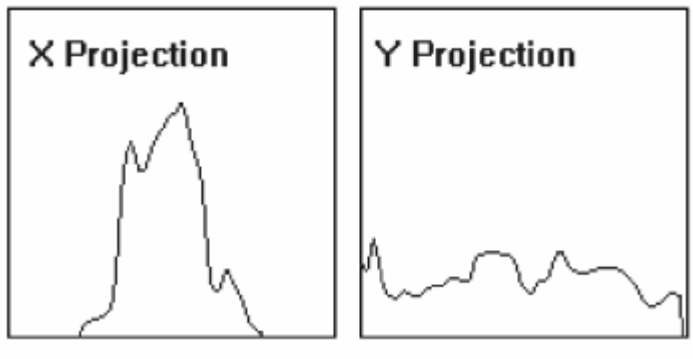


Figure 4. Horizontal (left) and vertical projection histogram

After generating the information above, a simple symmetry analysis is conducted on the information based on the body axis and the silhouette. Figure 5 illustrates this, where the gray areas are considered non-symmetric and the black areas are symmetric.

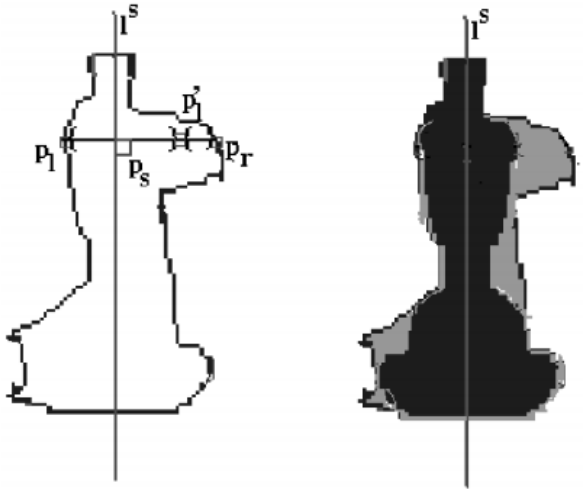


Figure 5. Example of non-symmetry analysis

Periodicity information is combined with the symmetry analysis to decide whether pixels correspond to objects carried by a person. The periodicity is computed by using the similarities of the last N projections over time. A non-symmetric region which does not have significant periodic motion is considered as an object carried by a person, otherwise it corresponds to a body part.

Similar to [31], approaches in [32] and [34] proposed methods to detect carried objects by comparing temporal templates against view-specific exemplars generated offline for unencumbered pedestrians. A likelihood map of protrusions, obtained from the match, is combined in a Markov random field for spatial continuity, from which they obtain a segmentation of carried objects using the maximum posteriori probability (MAP) solution.

2.5 Video Surveillance and Human Activity Detection

2.5.1 Video Surveillance

Video surveillance is an important research area concerning the real-time observation of entities of interest (e.g., people, vehicles) in an environment, resulting in a description of the activities of the objects within the environment. In general, a video surveillance system consists of the following procedures: background modeling and foreground detection, blob extraction, blob tracking, and if necessary, trajectory analysis and activity analysis. There are several video surveillance systems that have been developed.

Pfinder [4] is a real-time system for tracking people and interpreting their behavior. The system uses a multi-class statistical model of color and shape to obtain a 2D representation of head and hands in a wide range of viewing conditions. Pfinder has been successfully used in a wide range of applications including wireless interfaces, video databases, and low-bandwidth coding.

KidRooms [35] is a tracking system based on “closed world regions.” These are regions of space and time in which the specific context of what is in the regions is known. These regions are tracked in real-time domains where object motions are not smooth or rigid and where multiple objects are interacting. It was one of the first multiple people, fully automated, interactive, narrative environment ever constructed using non-encumbering sensors.

W4 [31] is a real time visual surveillance system for detecting and tracking multiple people and monitoring their activities in an outdoor environment. W4 employs a combination of shape analysis and tracking to locate people and their parts (head, hands, feet, and torso) and to create models of people's appearance so that they can be tracked through interactions and occlusions. It can determine whether a foreground

region contains multiple people and can segment the region into its constituent people and track them. W4 can also determine whether people are carrying objects, can segment objects from their silhouettes and construct appearance models for them so they can be identified in subsequent frames. W4 can recognize events involving people and objects, such as depositing an object, exchanging bags, or removing an object.

Chapter 3. Description of Our Approach

3.1 Overview of Our System

The proposed vision-based event detection and retrieval system for surveillance is able to recognize events such as: 1. One person being present in the scene; 2. Two people meeting; 3. Two people shaking hands; 4. One person performing a pointing gesture; 5. People carrying bags; 6. People dropping bags; 7. People stealing bags; 8. Two people exchanging bags. All these events can be constrained by adding specific visual attributes for the people involved. Figure 6 is the flow diagram of our system.

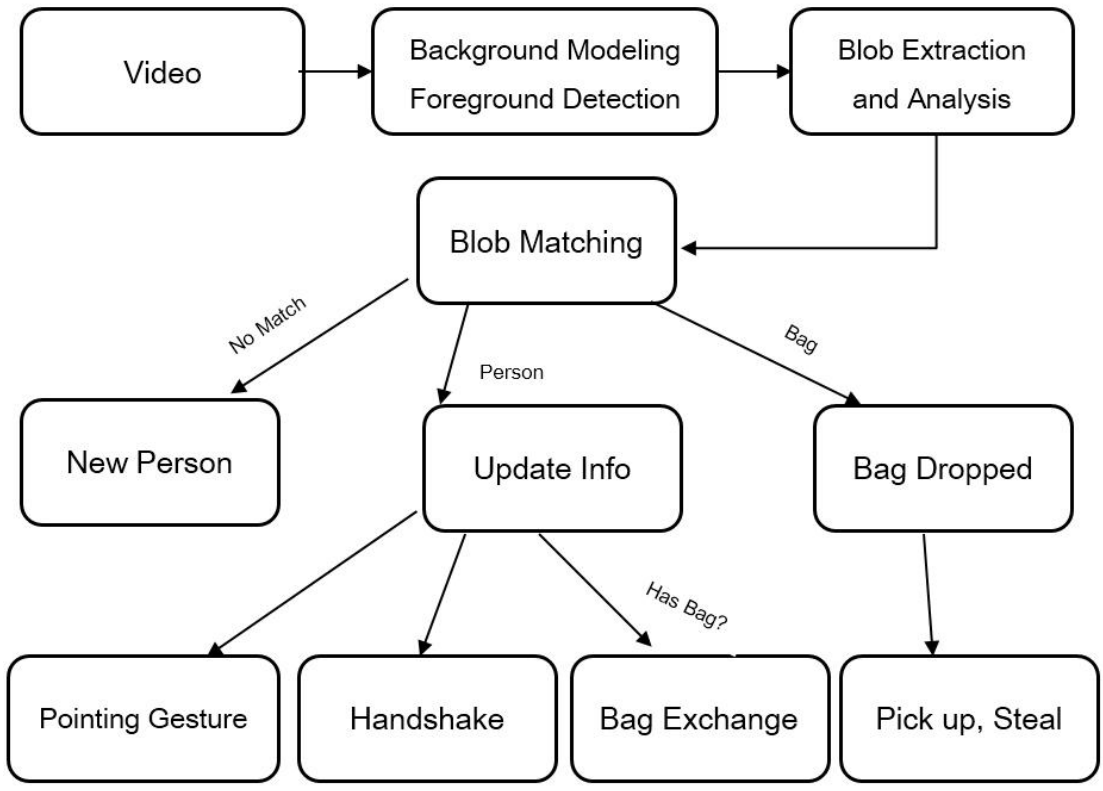


Figure 6. System Overview

In order to take advantage of depth data for better foreground detection and tracking, we use a single Kinect to capture videos as the input for our system. The following paragraphs provide a brief description of how we process the captured visual data.

First, the image frames containing both RGB and depth data from Kinect are used by the background modeling and foreground detection module. The detected foreground is then used by the blob extraction module to generate blobs; the blobs are then processed by a blob analysis module to decide whether they are noise (as the foreground detection does not guarantee perfect results), or contain one or multiple people.

Second, the color histograms of these blobs are computed, and combined with the Kalman filter tracker to match the blobs with existing people and bags. If a blob is matched with a person, the information of the person will be updated; if it is matched with a bag, then the bag might be dropped since itself is an individual blob; if this happens, further analysis is done to confirm whether it is a dropped bag or not; if the blob is not matched with anything, then we assume that the blob might be a new person.

Third, after updating people and bags status, another module is used to detect activities involving people and bags. We consider the following activities in this bag-related context: bag being dropped by a person, being picked up, being stolen, and bag being exchanged between two people; for people-only activities, we detect these following events: people walking, people meeting, people handshaking and one person pointing something to another person. In the following sections, we describe the details of our approach for each module.

3.2 Background Modeling and Foreground Detection

As discussed before, background modeling and foreground detection is a very important step prior to other procedures in many computer vision applications. The results of background modeling and foreground detection are usually propagated to

higher level modules and thus have heavy impact on the quality of the whole system. For our system, we tested several algorithms and decided to use an improved adaptive Gaussian mixture model [39] for foreground detection, which is very fast and produces generally good results.

The Gaussian mixture model that we used can automatically choose the number of Gaussians that are needed for each pixel, thus fully adapting to the scene characteristics. As discussed in section 2.1, the equation used to update the weights of a Gaussian mixture model is:

$$\hat{\pi}_m = \hat{\pi}_m + \alpha(o_m^{(t)} - \hat{\pi}_m)$$

where $\hat{\pi}_m$ is the weight of Gaussian m in a pixel model, α is an exponentially decaying envelope to limit the effect of old samples, $o_m^{(t)}$ is set to 1 if the new sample is close to the Gaussian m, otherwise 0, where the distance is defined as the Mahalanobis distance. The method used to update the weight is by adding (as prior knowledge) coefficients c to the equation.

Since we are using a multinomial distribution, prior knowledge can be utilized for better estimation. We denote a coefficient c_m as the number of samples that belong to Gaussian m a priori, then the Dirichlet prior [40] is:

$$\mathrm{P} = \prod_{m=1}^{M} \pi_m^{c_m}$$

By using the Dirichlet prior, we accept that the Gaussian m exists only if there is enough evidence from the previous samples.

Assume that the number of samples belonging to Gaussian m from time 1 to t is:

$$n_m = \sum_{i=1}^{t} o_m^{(i)}.$$

Taking the Dirichlet prior into account, the Maximum Likelihood Estimation is:

$$\frac{\partial}{\partial \hat{\pi}_m}\left(logL + logP + \lambda\left(\sum_{m=1}^{M} \hat{\pi}_m - 1\right)\right) = 0$$

where $\mathrm{L} = \prod_{m=1}^{M} \pi_m^{n_m}$, and λ is the Lagrange multiplier to constrain the weights to sum to 1. According to [41], we use negative coefficients $c_m = -c$. Solving the MLE, we get:

$$\hat{\pi}_m^{(t)} = \frac{\sum_{i=1}^{t} o_m^{(i)} - c}{t - Mc}$$

If we assume that $\mathrm{c/t}$ is fixed to $c_T = c/T$ with some large T and $1 - \mathrm{M}c_T \approx 1$, we can rewrite the formula above as a recursive equation:

$$\hat{\pi}_m = \hat{\pi}_m + \alpha\left(o_m^{(t)} - \hat{\pi}_m\right) - \alpha c_T$$

where $\alpha = 1/\mathrm{t}$. The equation above is used as the weight update equation in our system. For each pixel, we start modeling by using only one Gaussian distribution, and new Gaussians are added if none of the current Gaussian models are matched with the current sample. Gaussians whose weight becomes negative will be discarded. Figure 7 and Figure 8 show example results using the method described above.

Figure 7. Original image (left); detected foreground (right)

Figure 8. Original image (left); detected foreground (right)

3.3 Blob Extraction and Analysis

Background modeling and foreground detection determine whether a pixel belongs to the background or foreground. In order to integrate the foreground pixels into regions, we use a blob extraction algorithm. In our system, we use a two-pass connected component labeling algorithm as discussed in Section 2.2, as it provides good results in terms of both accuracy and speed.

Once the blobs are obtained, we cannot use the raw blobs directly for subsequent processing for several reasons. First, the background modeling and foreground detection does not guarantee absolutely reliable results - for example, when there is a large area of false foreground detection due to abrupt changes of illumination, the resulting blobs should not be used for further processing and thus must be ignored. In our approach we impose simple constrains for the blobs by limiting their size, width and height that cannot exceed a certain ratio of the image size. Second, multiple objects may be adjacent and connected, and they will be detected as a single blob in the blob extraction step. To make the subsequent processing easier, at this time we need to decide whether there are multiple people in a single blob. Figure 9 shows a situation when two people are adjacent, with only one blob detected.

Figure 9. Original image where two people are adjacent (left) ; detected foreground of the original image, with only one blob (right).

We use a similar approach to W4 that deals with people in groups [31]. Our method has two phases: first, we try to determine whether there are multiple people in a blob by identifying the number of potential heads; second, if the number of potential heads is larger than 1, the blob will be split into sub-blobs, which will be used to match with existing people, where the matching method is described later in this chapter.

Considering a general situation where a group of people are adjacent, we assume that their heads lie on the silhouette boundary and are visible most of the time. Based on this assumption and the fact that the head is always at the top of a blob, we use the vertical projection histogram of the binary silhouette to detect heads. Between each two adjacent peaks at the top of the blob, we denote the number of pixels that belong to the head by $n1$, and denote the number of pixels that are between two peaks but do not belong to the blob (the "gap" between two heads) by $n2$. We threshold the two numbers and their ratio in order to decide whether they may be two heads.

If the number of potential heads n is larger than 1, the blob is split into n sub-blobs equally. These sub-blobs are then verified by matching with the existing people

models, according to the method described later. If a sub-blob is not matched, it is discarded.

This simple multiple people analysis performs well in our experiments. For the frames with people totally occluded by others, they are ignored since we employ other mechanisms to deal with such issues during later stages of processing.

3.4 Human Tracking

Tracking is another important step in video surveillance systems. We use a method that combines Kalman filter tracking and a simple blob matching approach. In this section, we describe our method for human tracking in detail.

After the blobs are obtained from the blob extraction and analysis module, the blobs are matched with existing people and bags. We combine two types of information: color histograms and geometric information, trying to obtain the best tracking (matching) results. When a new person is detected, we initialize a Kalman filter tracker for the person. When a new blob matches existing people, for the geometric information we calculate the size and position difference D_i between the blob in the current frame and all people's true position and size in the previous frame. Since the motion of people may be abrupt and unstable, in order to get a smooth motion we also use the true position and size vector to update the Kalman filter tracker. By comparing the prediction of the position and size from the filter with the blob in the current frame we obtain the difference KD_i. We use the smaller of KD_i and D_i as the final distance FD_i for each person, where FD_i are further normalized to (0,1). For color information, we compute the color histogram of each blob, compare it with each people's color histogram by using the Bhattacharyya distance [37] to get a similarity

value C_i. Then the distance FD_i and the color histogram similarity C_i are combined by a weight w to get the match score for each person i:

$$\text{Match Score} = w * FD_i + (1 - w) * C_i$$

We use the highest match score as the best match with the blob if it exceeds a threshold. If matched, the model for that person is updated with the blob information, including the position, size, width, height, and the color histogram. The Kalman filter model is updated by the new position and size.

If the none of the match scores is higher than a threshold, then none of the people are considered to match with the blob. If this happens for several consecutive frames, then the blob is considered as a new person, and a Kalman filter is initialized for it.

3.5 Bag Detection, Tracking and Related Activities Detection

In this section we describe our approach to detect people carrying bags, and bag-related activities: bag drop, pickup, steal, and exchange.

3.5.1 Bag Detection

We use an approach inspired from W4 [31] to detect people carrying bags. The major difference between our approach and W4 is that we utilize color information. We assume that the color of the bag is different from the clothing color of the person who carries the bag. Here we describe our approach to bag detection in detail.

Similar to W4, we define a center axis of a person. W4 [31] calculates the axis by applying Principal Component Analysis to the silhouette pixels. The best fit axis is constrained by minimizing the sum of absolute perpendicular distances to that axis.

This method does not consider the fact that the bag can affect the center axis position, since the bag affects the silhouette of the whole blob (person and bag). Here, we use a different method to determine the center axis; instead of considering the whole blob silhouette, we only use the top part of the blob, which is the head. We compute the center axis based on the position of the head in the image. We consider the center axis of the head as the center axis of the whole body.

In case of a new person (for which a bag has not been detected yet), we perform a symmetry analysis as follows. First we determine the pixels that are non-symmetric with respect to the center axis. Pixels are classified into two groups based on whether they are in the bottom right or bottom left side of the axis. Then we count the number of the pixels in each group and compute the color histogram of the pixels in each group. The two histograms are then compared with the histogram of the symmetric pixels at the bottom by using the Bhattacharyya distance [37].

Assume that the number of non-symmetric pixels of the two sides are Cl and Cr, and the results of the histogram comparison are Hl and Hr respectively. We use a threshold T_c as the minimum number of pixels for a bag. We use the following method to determine whether a person is carrying a bag or not:

$$\mathrm{p}(\text{carry bag on left} \mid \text{person}) = w * (\max(0, (Cl - T_c)/N) + (1 - w) * \mathrm{Hl}$$

$$\mathrm{p}(\text{carry bag on right} \mid \text{person}) = w * (\max(0, (Cr - T_c)/N) + (1 - w) * \mathrm{Hr}$$

where N is a constant. We use the larger p from above and compare it with a threshold T_b to determine if the person is carrying a bag at time t. If p is larger than T_b in a certain period of time $(\mathrm{t} - \mathrm{k}, \mathrm{t} + \mathrm{k})$, then the person is considered to be carrying a bag. Figure 10 is an example showing a person carrying a bag.

Figure 10. Original image, a person carrying a bag (left); detected foreground (right)

When a bag is detected with a person, we use the same method to track the bag as for tracking the person; we build a Kalman filter for the bag, combined with the color histogram to track the bag in subsequent frames. When a person is detected carrying a bag, we use a different (lower) threshold for color histogram comparison and a different (lower) threshold for the total number of pixels to track the bag. The reason is that bag detection should use more restrictive constraints to prevent false positives.

3.5.2 Bag Drop Detection

When a bag is dropped by a person, the bag itself becomes a blob. We use this information to detect whether a bag is being dropped. For each new blob, we compare it with the existing bags as well as the existing people. For each existing bag, we compute the match score with the blob using a method similar to the one used for computing the match score between the blob and people, except that we add the total number of pixels. The match score is computed as follows:

$$\text{Match Score} = w1 * FD_i + \text{w2} * C_i + w3 * (S_b - S_{bi})$$

where FD_i is the smaller value of the distance estimated from observed bags and the prediction of the Kalman filter of the i^{th} bag, C_i is the result of histogram comparison, and S_b, S_{bi} are the number of pixels of the blob and the i^{th} bag.

When a match score is larger than a threshold, we hypothesize that the bag may be dropped by a person at time t. If this happens for a certain number of times in the time interval $(t - k, t + k)$, then we infer that the bag is indeed dropped.

3.5.3 Bag Pickup and Steal Detection

When a bag is detected as dropped, the information for the bag is saved, including pixel coordinates and their RGB values. This information can be used for detecting whether someone picks up the bag later. If a bag is dropped, then in later frames the pixels values at the position where the bag is dropped will be monitored. If the change of the RGB values in these positions is larger than a threshold for a period of time, then the bag is considered to be picked up by someone. If the person who picked it up is different from the one who dropped it, then the person is considered to steal the bag.

3.5.4 Bag Exchange Detection

Detecting a bag being exchanged between two persons is performed as follows. We assume that the two people are close and have no large motion in a period of time. If this condition is satisfied, then we examine the distance between the bag and the two people; if the distance of the bag to the original owner is getting larger and the distance to the other person is getting smaller for a certain number of frames in a time interval, then the bag is considered to be exchanged from the owner to the other person.

3.6 Human Activity Recognition

Human activity recognition is one of the major goals of video surveillance applications. In our system, we focus on activities between two people: (i) handshaking, and (ii) one person pointing something to a second person. In this section, we describe how these activities are detected.

3.6.1 Handshaking Recognition

The method we used in handshake detection is quite intuitive. First, there are at least two people being almost static and their blobs are close and connected; second, the hands of the two people are connected in a line/curve. We utilize these two criteria to perform the detection. Figure 11 is an example of two people handshaking.

Figure 11. People shake hands

We first check whether two people that are close to each other are connected, and whether they are relatively static. Since they should be connected, they must come from the same parent blob, as they are the sub-blobs obtained from splitting the parent

blob. If these conditions are satisfied, then the following method is used to detect handshake.

We define a rectangle window, whose vertices are located at the intersection of the two vertical axes, the higher headline and the center horizontal axis of the two people. Then we count the number of pixels h that belong to the foreground (hands), as well as the number of pixels b that belong to the background . We compare the two numbers with two thresholds Th and Tb, as well as the ratio of the two numbers h/b with a threshold T to hypothesize whether a handshake may be occurring. If this happens for a number of consecutive frames, then we infer a handshake event.

3.6.2 Pointing Somewhere Recognition

We use a similar approach to detect pointing gestures. We assume that two people are close to each other and they stay still for a while. If this condition is satisfied, then we detect whether one person stretches out a hand by the following method. We use the center axis, the leftmost/rightmost vertical lines, the 2/5 upper body line and the top horizontal line to form a rectangle, as shown in Figure 12. Within the rectangle, we count the number of non-symmetric pixels that belong to the person (denoted by N_1), the pixels that belong to the background (denote by N_2), and compute their ratio $R = N_1/N_2$. If N_1, N_2 and R satisfy some thresholds for a number of consecutive frames, the pointing gesture is inferred.

Figure 12. A person's pointing gesture (left); detected foreground objects (right).

3.7 System Interface and Functionality

We developed an interface for our system, based on Qt under Ubuntu. The interface is shown in Figure 13.

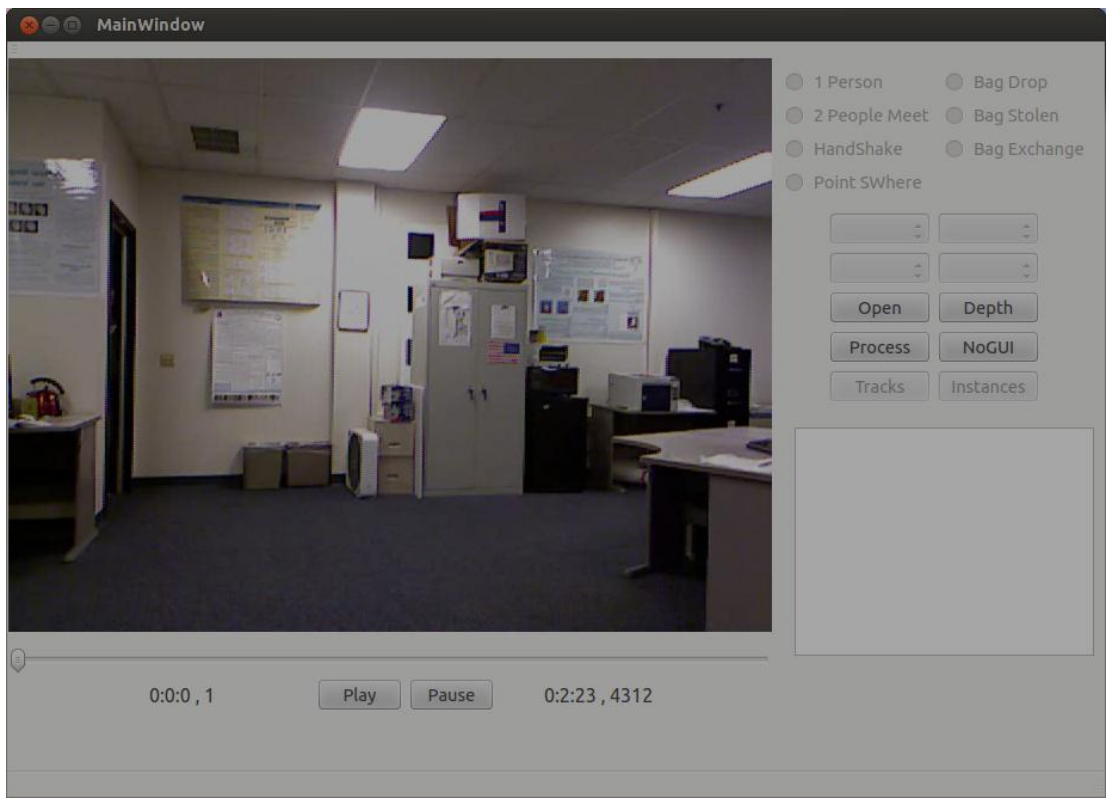


Figure 13. System interface

The OPEN button opens an RGB video and the associated results file, while the DEPTH button opens a Depth video and is optional. When no correspondent depth video is available, the system will process RGB video only. The PROCESS button is used to process the video. Time and frame numbers are showed at the bottom.

When the processing is completed, the results are saved to a file, which can be opened and used to see the results directly, without repeating the processing. The result file contains the video path, frame index, people and bag information. The information for a person includes its identifier, position, color, whether is carrying a bag (and if so the bag identifier), whether it is involved in handshaking, in a pointing gesture, etc. The information for a bag includes its identifier, position, whether it is dropped, picked up, stolen, or exchanged, and the related person's information.

The TRACK button is used to generate tracks (instances) of events, according to visual attributes specified through the interface. Clicking on each track will play the corresponding portion of the video where the event was detected.

Chapter 4. Experimental Results

We tested our approach on a computer equipped with an Intel Core 2 Duo CPU E8500 at 3.16GHz. Two videos taken with a Kinect camera from different locations and at different times have been used.

4.1 One Person Tracking Results

For the one person tracking, we specify a color through the interface, then the people with clothing of that color will be tracked. In our system, the accuracy of one person tracking is nearly 100%. Figure 14 shows results of one person tracking.

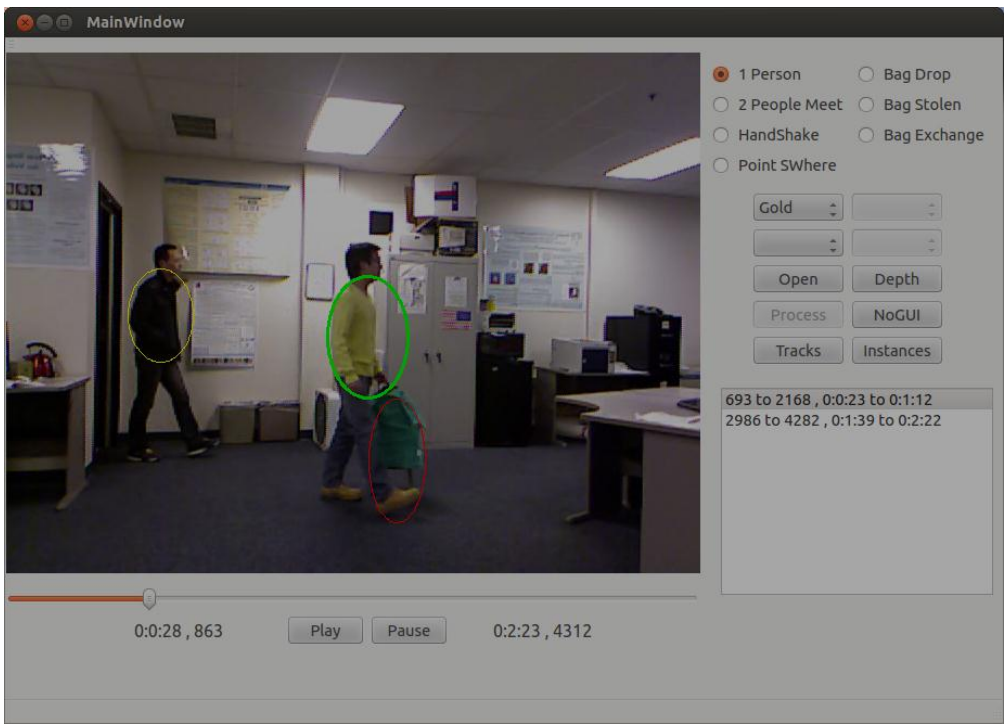


Figure 14. Example of one person tracking; a person with the selected color “Gold” is shown with a thick green ellipse in the image.

4.2 Two People Meeting Detection

The user can specify one or both colors for detecting and tracking people meeting. Figure 15 shows an example where we select “Gold” as the color of the first person, and “Any” for the second person.

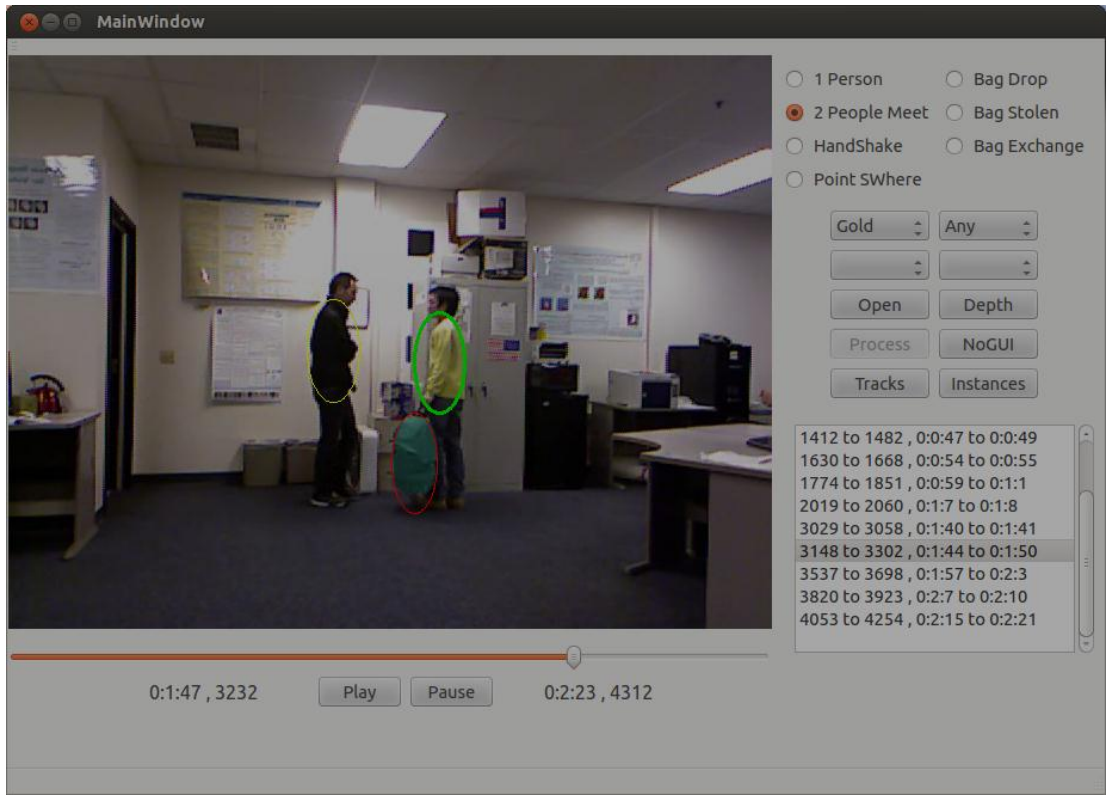


Figure 15. Two people meeting detection

4.3 Bag Drop

In figure 16, we specify a color for the person who drops a bag, while in figure 17, we are interested in bag drop events involving persons with any visual attributes.

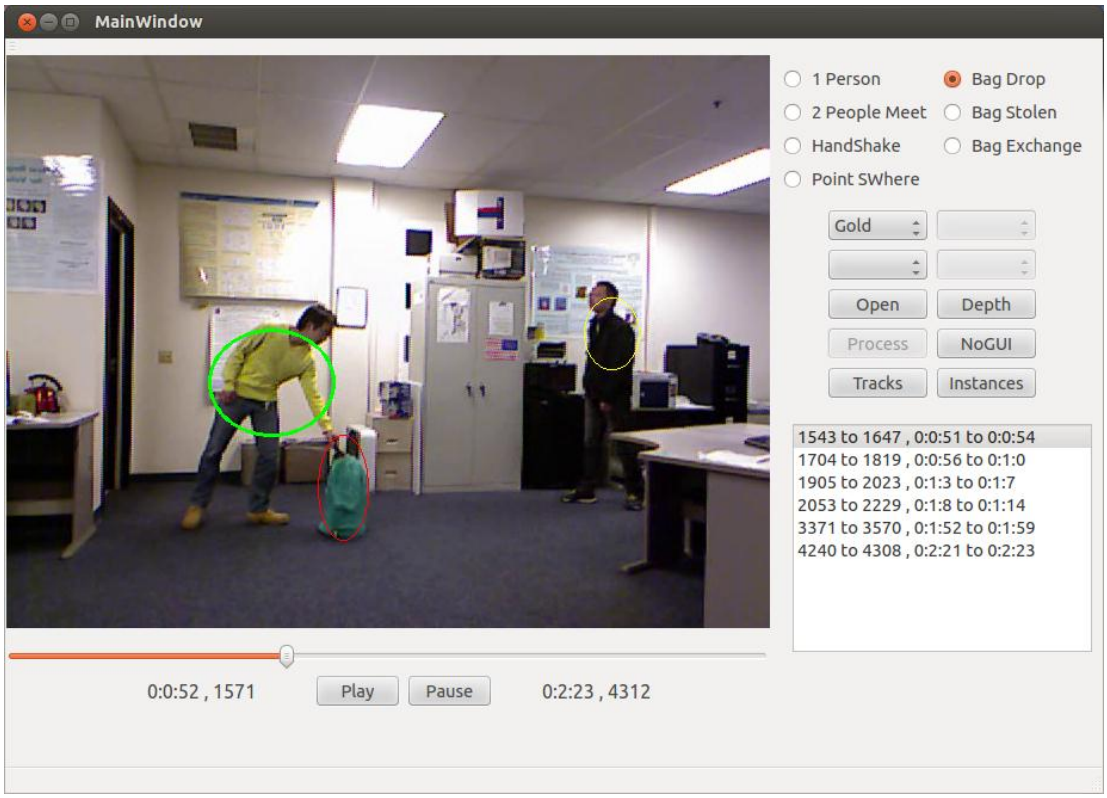


Figure 16. A person with color “Gold” drops a bag.

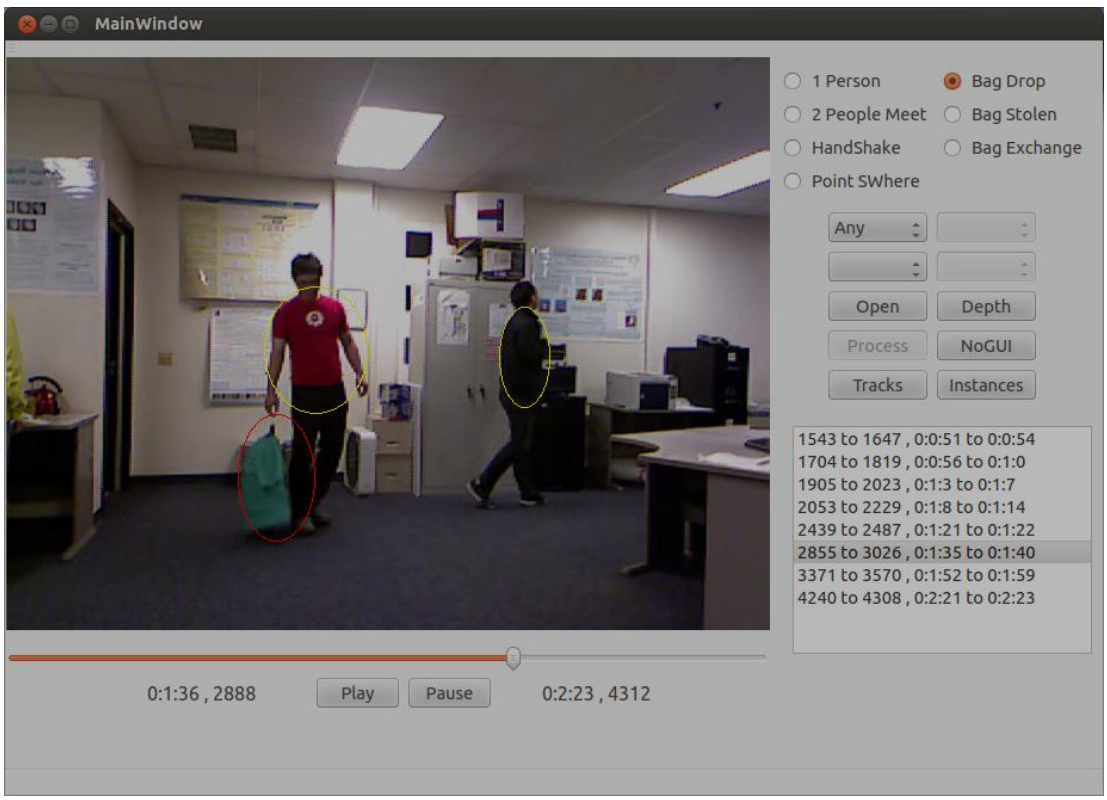


Figure 17. Any person drops a bag.

4.4 Bag Steal

In this experiment, the person with the gold color is the owner of the bag; at frame 2080 the person dropped the bag; at frame 2227, another person (in red) stole the bag. Figure 18 is an example of the owner dropped a bag, Figure 19 is an example of another person stole the bag.

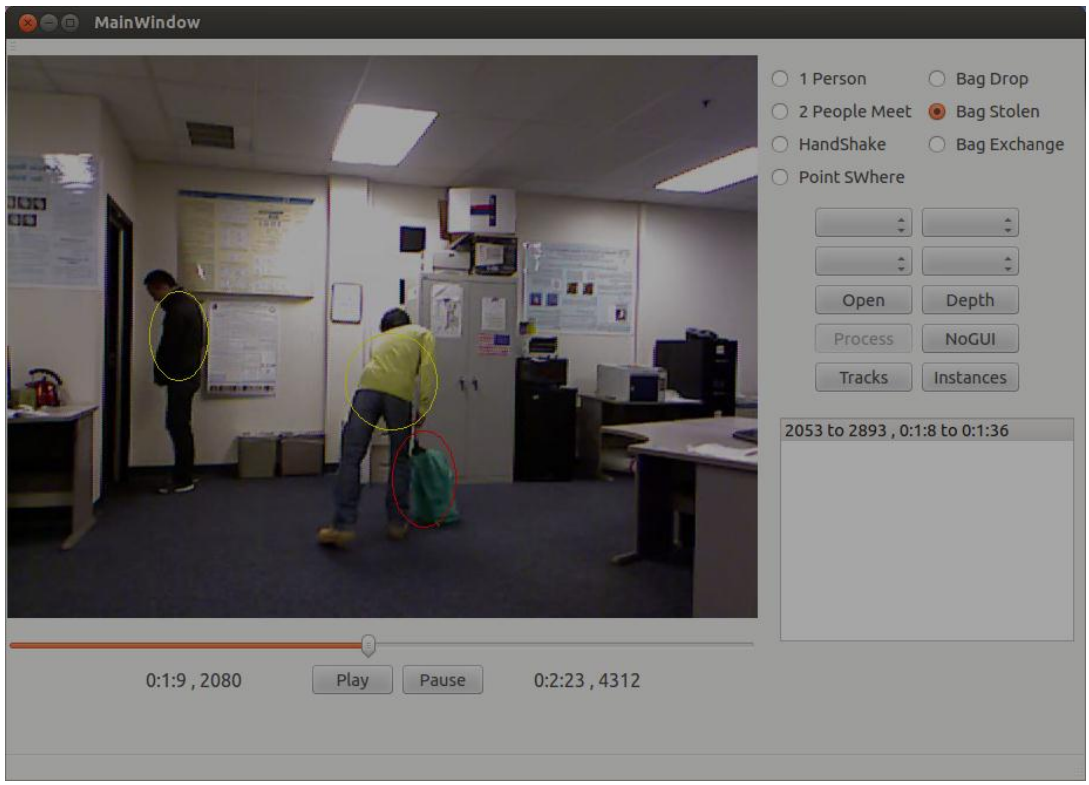


Figure 18. Owner dropped a bag

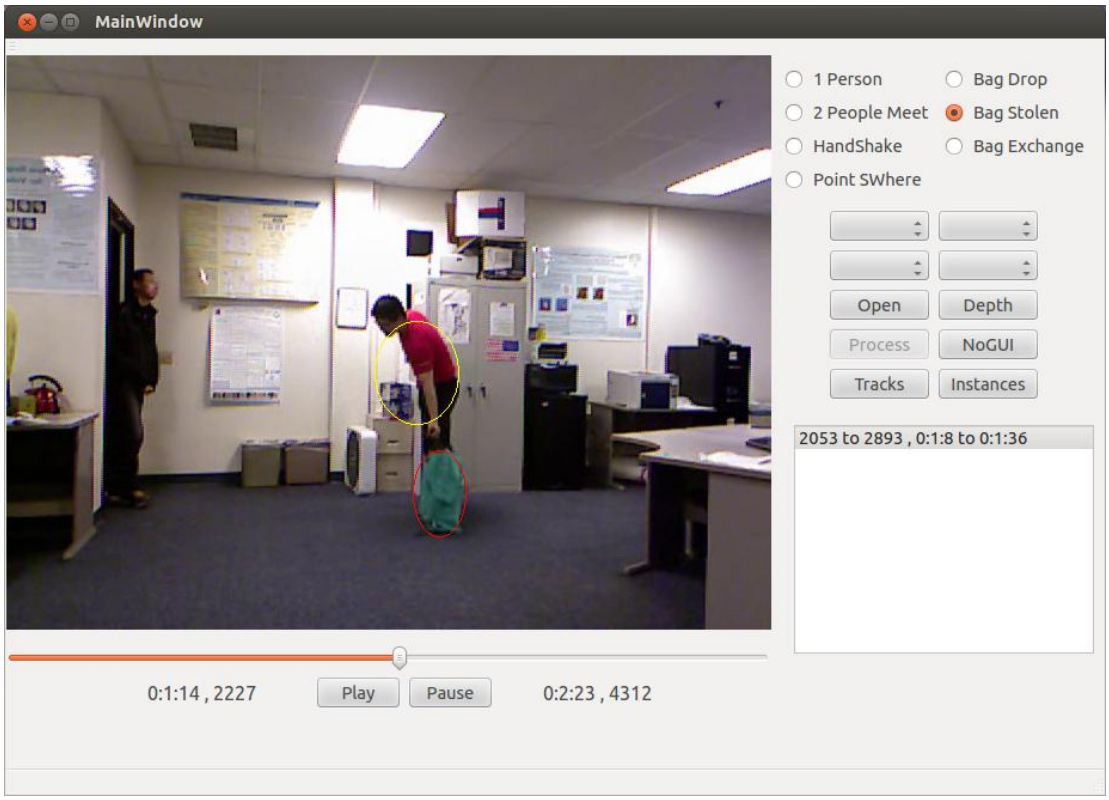


Figure 19. A person stole the bag

4.5 Bag Exchange

Figure 20 is an example of bag exchange between two people.

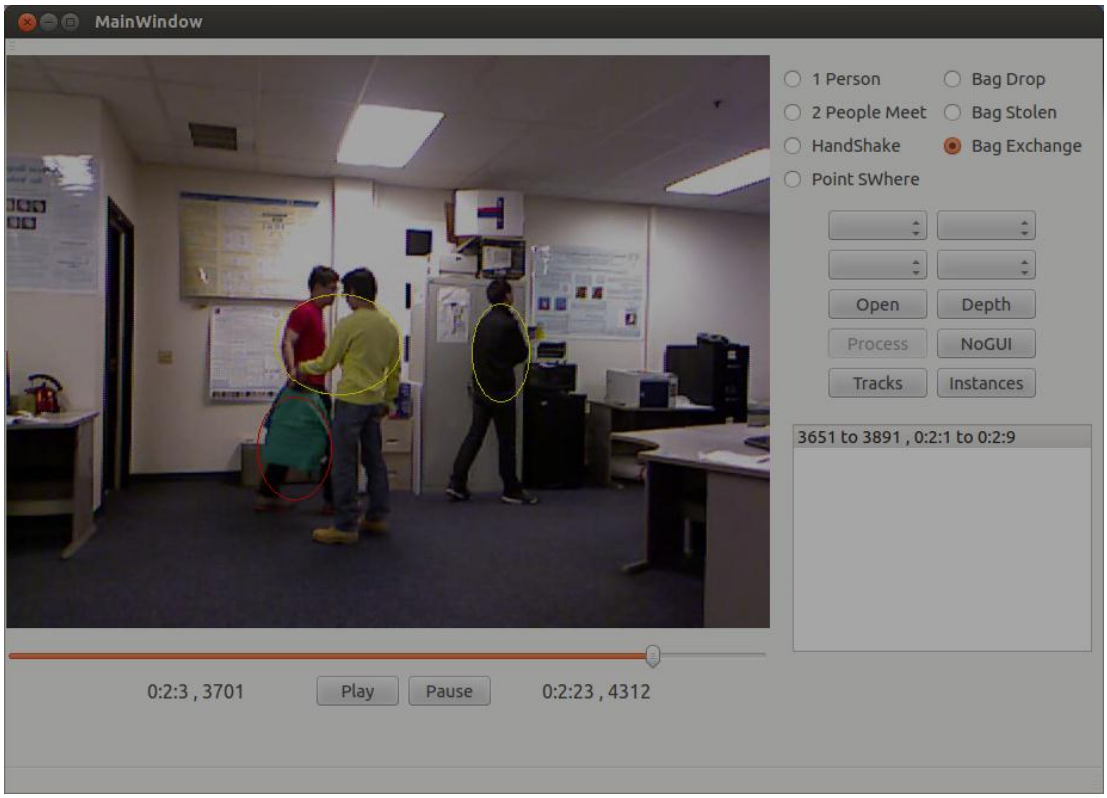


Figure 20. A person is handing a bag to another person.

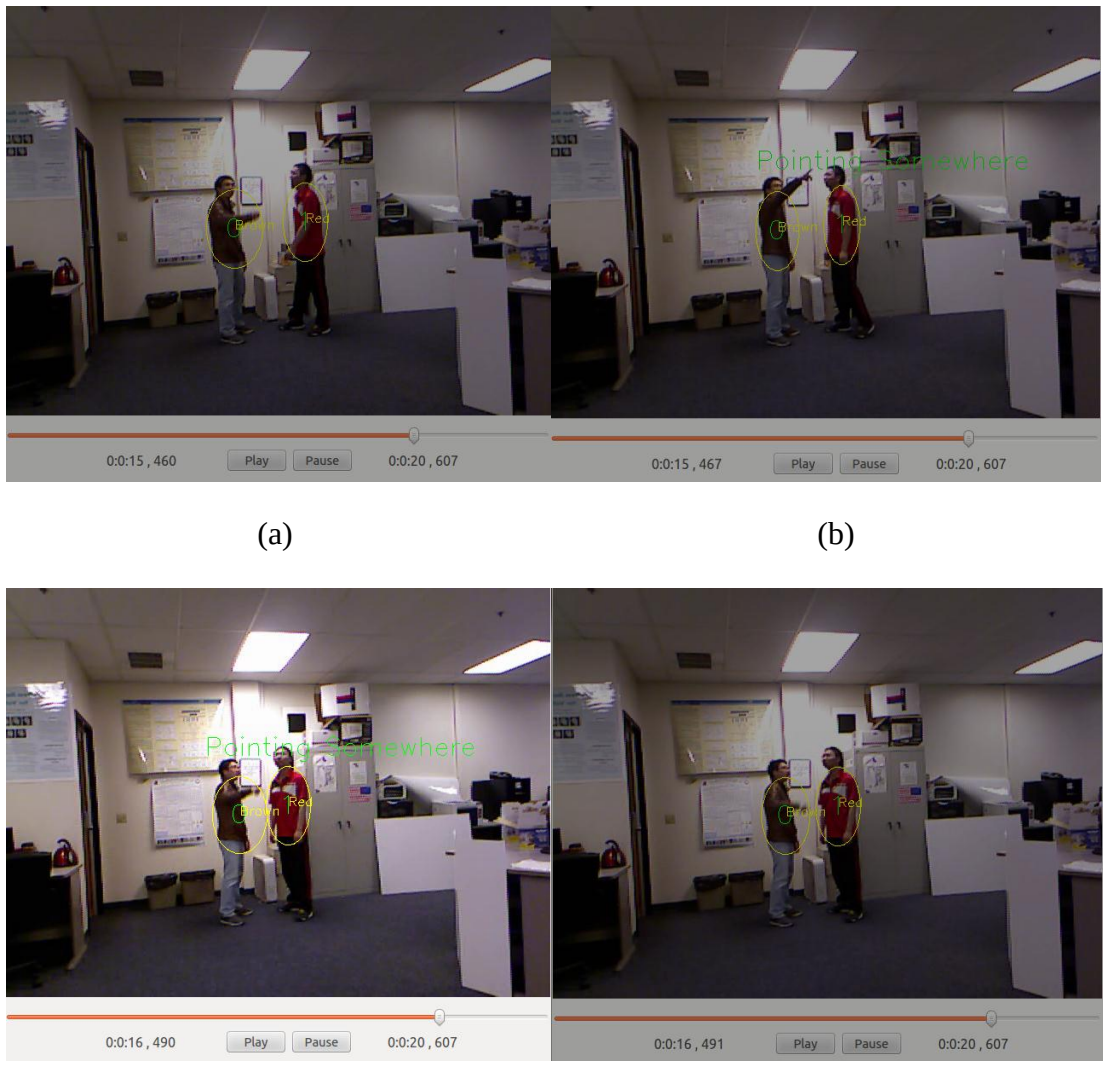
4.6 Handshake

Figure 21 shows results for handshake detection between two people. In frame 283, the two people attempt to have a handshake; from frame 288, their hands start to be connected; from frame 294, the system starts to recognize the activity, which means that it only took 6 frames to detect the event; the system stops detecting the activity at frame 306, and the hands are taken back at frame 314.

Figure 21. Example of handshake (a) people attempt to shake hands; (b) people start to shake hands; (c) handshaking starts to be detected; (d) last frame where the handshaking is detected; (e) Hands are taken back.

4.7 Pointing Gesture

In our experiment, the pointing gesture started at frame 460; the system detected the gesture from frame 467 to 490; the gesture ended at frame 491. Figure 22 shows the results.



(a) (b)

(c) (d)

Figure 22. Example of Pointing gesture: (a) Pointing gesture started; (b) Gesture being recognized (c) Last frame being recognized; (d) Pointing gesture ended.

Chapter 5. Conclusion

We presented an system that can robustly detect and retrieve events from large surveillance video, which has the following capabilities:

1. Detect and track multiple people and bags;
2. Detect bags being dropped, picked up, stolen, and exchanged;
3. Detect handshaking between two people;
4. Detect one person's pointing gesture to another person.

5.1 Discussion

We use an efficient background modeling and foreground detection technique, which is based on a mixture of Gaussians model. The parameters of the model are constantly and recursively updated, the number of Gaussians for each pixel is also adaptively updated.

We use a stable connected component labeling algorithm to extract blobs from the detected foreground pixels. We also use a simple but reliable method to detect multiple people in a single blob that achieves very high accuracy.

We developed a novel method that combines geometric and color information to track people and bags, while other algorithms such as Camshift [48] were found to be rather unstable and unreliable in our experiments (especially in the presence of occlusion). Our approach for tracking is quite reliable, with an accuracy for tracking people as high as 95% when good results were obtained in foreground detection.

We use shape symmetry analysis and color information to detect people carrying bags. With good results for bag detection, bag drop and bag steal events have been recognized very reliably.

We also developed algorithms for human activity recognition. We detect two kinds of activities between two people: handshaking and pointing gestures. In our experiments, the results showed that our methods achieve both a high accuracy rate and a correct duration for the detected events.

5.2 Future Work

Our system heavily depends on the results of background modeling and foreground detection. As the modeling technique we used in our system is quite sensitive to illumination conditions, we noticed that the exposure in the video captured by Kinect would change sometimes due to the instability of the fluorescent lighting. More effort should be dedicated to develop more stable algorithms for background modeling and foreground detection.

For bag detection we assumed that the color of the bag is somewhat different from the color of the lower body of the person, which is not always the case. Methods that can detect and track the bag under complex condition in real-time video are desirable.

While handshaking and pointing gestures are relatively simple activities, detecting more complex activities and gestures would be very relevant for surveillance applications.

Reference

[1] Bouwmans, T. (2011). Recent advanced statistical background modeling for foreground detection-a systematic survey. *Recent Patents on Computer Science*, *4*(3), 147-176.

[2] McFarlane, N. J., & Schofield, C. P. (1995). Segmentation and tracking of piglets in images. *Machine Vision and Applications*, *8*(3), 187-193.

[3] Lee B., Hedley M. (2002) Background Estimation for Video Surveillance, IVCNZ 2002, pages 315-320, 2002.

[4] Wren, C. R., Azarbayejani, A., Darrell, T., & Pentland, A. P. (1997). Pfinder: Real-time tracking of the human body. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, *19*(7), 780-785.

[5] Fran ̧cois, A. R., & Medioni, G. G. (1999, June). Adaptive color background modeling for real-time segmentation of video streams. In *Proceedings of the International Conference on Imaging Science, Systems, and Technology* (Vol. 1, No. 121, pp. 227-232).

[6] Kim, K., Chalidabhongse, T. H., Harwood, D., & Davis, L. (2005). Real-time foreground–background segmentation using codebook model. *Real-time imaging*, *11*(3), 172-185.

[7] Stauffer, C., & Grimson, W. E. L. (1999). Adaptive background mixture models for real-time tracking. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on* (Vol. 2). IEEE.

[8] Elgammal, A., Harwood, D., & Davis, L. (2000). Non-parametric model for background subtraction. In *Computer Vision—ECCV 2000* (pp. 751-767). Springer Berlin Heidelberg.

[9] Suzuki, K., Horiba, I., & Sugie, N. (2003). Linear-time connected-component labeling based on sequential local operations. *Computer Vision and Image Understanding*, *89*(1), 1-23.

[10] Haralick, R.M., Some neighborhood operations, in: Real Time/Parallel Computing Image Analysis, Plenum Press, New York, 1981, pp. 11–35.

[11] Hashizume, A., Suzuki, R., Yokouchi, H., Horiuchi, H., & Yamamato, S. (1990). An algorithm of automated RBC classification and its evaluation. *Bio Medical Engineering*, *28*(1), 25-32.

[12] Gotoh, T., Ohta, Y., Yoshida, M., & Shirai, Y. (1987, October). Component labeling algorithm for video rate processing. In *Hague International Symposium*(pp. 217-224). International Society for Optics and Photonics.

[13] Babenko, B., Yang, M. H., & Belongie, S. (2011). Robust object tracking with online multiple instance learning. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, *33*(8), 1619-1632.

[14] Komeichi, Masatoshi, et al. "Video-rate labeling processor."*1988 Intl Congress on Optical Science and Engineering*. International Society for Optics and Photonics, 1989.

[15] Lumia, R., Shapiro, L., & Zuniga, O. (1983). A new connected components algorithm for virtual memory computers. *Computer Vision, Graphics, and Image Processing*, *22*(2), 287-300.

[16] Dillencourt, Michael B., Hanan Samet, and Markku Tamminen. "A general approach to connected-component labeling for arbitrary image representations." *Journal of the ACM (JACM)* 39.2 (1992): 253-280.

[17] Gargantini, I., & Tabakman, Z. (1982). Separation of connected component using linear quad-and oct-trees. In *Proc. 12th Conf. Numerical Mathematics and Computation* (Vol. 37, pp. 257-276).

[18] Hecquard, J., & Acharya, R. (1991). Connected component labeling with linear octree. *Pattern recognition*, *24*(6), 515-531.

[19] Samet, H. (1981). Connected component labeling using quadtrees. *Journal of the ACM (JACM)*, *28*(3), 487-501.

[20] Bhattacharya, P. (1996). Connected component labeling for binary images on a reconfigurable mesh architecture. *Journal of Systems Architecture*, *42*(4), 309-313.

[21] Choudhary, A., & Thakur, R. (1994). Connected component labeling on coarse grain parallel computers: an experimental study. *Journal of Parallel and Distributed Computing*, *20*(1), 78-83.

[22] Hirschberg, D. S., Chandra, A. K., & Sarwate, D. V. (1979). Computing connected components on parallel computers. *Communications of the ACM*,*22*(8), 461-464.

[23] Fiorio, C., & Gustedt, J. (1996). Two linear time union-find strategies for image processing. *Theoretical Computer Science*, 1*54*(2), 165-181.

[24] Tarjan, R. E. (1975). Efficiency of a good but not linear set union algorithm.*Journal of the ACM (JACM)*,*22*(2), 215-225.

[25] http://www.cse.unr.edu/~bebis/CS791E/Notes/ConnectedComponents.pdf

[26] Chang, F., Chen, C. J., & Lu, C. J. (2004). A linear-time component-labeling algorithm using contour tracing technique. *computer vision and image understanding*, *93*(2), 206-220.

[27] Wu, K., Otoo, E., & Shoshani, A. (2005, April). Optimizing connected component labeling algorithms. In *Medical Imaging* (pp. 1965-1976). International Society for Optics and Photonics.

[28] Kalman, R. E., & Bucy, R. S. (1961). New results in linear filtering and prediction theory. *Journal of basic engineering*, *83*(1), 95-108.

[29] Hostetler, L. (1975). The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Transactions on information theory*,*21*(1), 32-40.

[30] Comaniciu, D., Ramesh, V., & Meer, P. (2000). Real-time tracking of non-rigid objects using mean shift. In *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on* (Vol. 2, pp. 142-149). IEEE.

[31] Haritaoglu, I., Harwood, D., & Davis, L. S. (2000). W 4: Real-time surveillance of people and their activities. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, *22*(8), 809-830.

[32] Haritaoglu, I., Cutler, R., Harwood, D., & Davis, L. S. (1999). Backpack: Detection of people carrying objects using silhouettes. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on* (Vol. 1, pp. 102-107). IEEE.

[33] Jolliffe, I. (2005). *Principal component analysis*. John Wiley & Sons, Ltd.

[34] Damen, D., & Hogg, D. (2012). Detecting carried objects from sequences of walking pedestrians. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, *34*(6), 1056-1067.

[35] Bobick, A., Davis, J., Intille, S., Baird, F., Cambell, L., Irinov, Y., ... & Wilson, A. (1996). Kidsroom: Action recognition in an interactive story

environment.*Mass. Inst. Technol., Cambridge, Perceptual Computing Tech. Rep, 398*.

[37] Kailath, T. (1967). The divergence and Bhattacharyya distance measures in signal selection. *Communication Technology, IEEE Transactions on*, *15*(1), 52-60.

[38] Augimeri, A., Fortino, G., Rege, M. R., Handziski, V., & Wolisz, A. (2010, October). A cooperative approach for handshake detection based on body sensor networks. In *Systems Man and Cybernetics (SMC), 2010 IEEE International Conference on* (pp. 281-288). IEEE.

[39] Zivkovic, Z. (2004, August). Improved adaptive Gaussian mixture model for background subtraction. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on* (Vol. 2, pp. 28-31). IEEE.

[40] Stephens, M., & Donnelly, P. (2003). A comparison of bayesian methods for haplotype reconstruction from population genotype data. *The American Journal of Human Genetics*, *73*(5), 1162-1169.

[41] Zivkovic, Z., & van der Heijden, F. (2004). Recursive unsupervised learning of finite mixture models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, *26*(5), 651-656.

[42] Ianasi, C., Gui, V., Toma, C. I., & Pescaru, D. (2005). A fast algorithm for background tracking in video surveillance, using nonparametric kernel density estimation. *Facta universitatis-series: Electronics and Energetics*, *18*(1), 127-144.

[43] Tavakkoli, A., Nicolescu, M., & Bebis, G. (2006, May). An adaptive recursive learning technique for robust foreground object detection. In *proceedings of the*

International Workshop on Statistical Methods in Multi-image and Video Processing (in conjunction with ECCV06).

[44] Mittal, A., & Paragios, N. (2004, July). Motion-based background subtraction using adaptive kernel density estimation. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on* (Vol. 2, pp. II-302). IEEE.

[45] Culibrk, D., Marques, O., Socek, D., Kalva, H., & Furht, B. (2007). Neural network approach to background modeling for video object segmentation. *Neural Networks, IEEE Transactions on, 18*(6), 1614-1627.

[46] Messelodi, S., Modena, C. M., Segata, N., & Zanin, M. (2005). A kalman filter based background updating algorithm robust to sharp illumination changes. In*Image Analysis and Processing–ICIAP 2005* (pp. 163-170). Springer Berlin Heidelberg.

[47] Butler, D., Sridharan, S., & Bove Jr, V. M. (2003, April). Real-time adaptive background segmentation. In *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on* (Vol. 3, pp. III-349). IEEE.

[48] Allen, J. G., Xu, R. Y., & Jin, J. S. (2004, June). Object tracking using camshift algorithm and multiple quantized feature spaces. In *Proceedings of the Pan-Sydney area workshop on Visual information processing* (pp. 3-7). Australian Computer Society, Inc..