Accurate and Timely Situation Awareness Retrieval from a Bandwidth Constrained Camera Network

Tuan Dao*, Amit Roy-Chowdhury*, Nasser Nasrabadi†
Srikanth V. Krishnamurthy*, Prasant Mohapatra§, and Lance M. Kaplan‡
* University of California, Riverside † West Virginia University § University of California, Davis ‡ U.S. Army Research Laboratory

***Abstract*—Wireless cameras can be used to gather situation awareness information (e.g., humans in distress) in disaster recovery scenarios. However, blindly sending raw video streams from such cameras, to an operations center or controller can be prohibitive in terms of bandwidth. Further, these raw streams could contain either redundant or irrelevant information. Thus, we ask "how do we extract accurate situation awareness information from such camera nodes and send it in a timely manner, back to the operations center?" Towards this, we design ACTION, a framework that (a) detects objects of interest (e.g., humans) from the video streams, (b) combines these streams intelligently to eliminate redundancies and (c) transmits only parts of the feeds that are sufficient in achieving a desired detection accuracy to the controller. ACTION uses small amounts of metadata to determine if the objects from different camera feeds are the same. A resource-aware greedy algorithm is used to select a subset of video feeds that are associated with the same object, so as to provide a desired accuracy, for being sent to the operations center. Our evaluations show that ACTION helps reduce the network usage up to threefold, and yet achieves a high detection accuracy of $\approx 90\%$.**

I. Introduction

Natural disasters usually have a high associated human cost; for example, the recent Nepal earthquake resulted in the death of more than 8,000 people, injury to more than 14,000, and over 300 people are still missing [1]. Today, advanced technologies can help in significantly enhancing search and rescue missions; sensors, often with camera capabilities can be deployed in the field, to provide situation awareness back to a central operations center or controller. Specifically, this is information with regards to particular objects of interest (e.g., distressed or injured humans, or animals) that would be critical in aiding search and rescue. In other situations (e.g., Boston marathon bombing), being able to quickly detect unattended objects such as luggage or backpacks using such cameras could help prevent tragic disasters from happening.

Unfortunately, in many such scenarios, in the aftermath of disasters the bandwidth is likely to be limited[1]. Blindly sending the video feeds from camera nodes in the field is likely to be infeasible because of bandwidth limitations. As information collected from multiple cameras with overlapping views tends to contain content with a high level of redundancy, transferring all raw video content from all of the cameras is also likely to be wasteful. Doing so may also delay the transfer of key information with regards to some of the objects of interest. Finally, having to look at large volumes of video may cause an inherent information overload on humans who man the central controller.

[1]Natural disasters come with at least some destruction of physical network infrastructure and this impacts communications [2].

In this work, "we seek to extract accurate situation awareness information from the camera feeds from a set of wireless cameras, and deliver it in a timely way to an operations center that handles search and rescue, when presented with bandwidth constraints." Before we describe the challenges in addressing this overarching objective and our contributions, we first formally define our view of how the situation awareness information is gathered and sent to the operations center. We envision that multiple autonomous camera-equipped sensors with possibly overlapping views, are deployed in the field and are used towards searching for particular objects of interest. The camera nodes possess processing capabilities and can locally extract situational information about the objects of interest from captured video feeds. They can then report information extracted from the videos (e.g., frames or parts of a frame) to a central controller via a wireless network with limited available bandwidth. The controller sends queries which seek for example, to determine whether there was an object of interest (e.g., a human, a vehicle, or a backpack) present at a specified location at a specified time.

Challenges: In order to achieve our overarching goal, we need to tackle some key challenges. First, we need to identify those videos that contain the "same" object of interest (at a given location and at a given time) in an automated way; in essence an object needs to be re-identified across the cameras. This is critical in elimination of redundant content (only if the camera views are capturing the same object can they be considered redundant). Depending on the location of the object relative to the camera, today's vision algorithms might not be able to categorically determine if there is a real object in view. They can only provide an assessment of the accuracy of their detection. Thus, a challenge we need to address is "how to effectively aggregate information sent from multiple cameras to improve the quality of object detection?" Finally, the transfer of all of the raw data with respect to all the detected objects may still be beyond the network capacity. Thus, how can we identify redundant content, and choose only the most relevant sub-set of this content and transfer this information back to the central node?

Our framework in brief: Towards addressing the above challenges and providing accurate and timely situational awareness with regards to objects of interest in the field, we design and implement a framework that we call ACTION. ACTION has the following component modules: (i) each individual camera has a module that aids lightweight object detection from the video collected; here we leverage state of the art computer vision algorithms (ii) a novel module that facilitates coordination across multiple cameras to aggregate information towards (a) re-identification of an object across multiple cameras and (b) using the joint camera views of the object to improve the quality of detection, and finally, (iii) a module that, based on the previous step, selects a sub-set of the video feeds for transfer to the central controller, that provide the highest accuracy given a bandwidth constraint. In what follows, we briefly describe the functions of each of the three modules. To keep the narrative clean, we focus on human detection; with minor modifications, ACTION is applicable for the detection of other types of objects (e.g., animals or luggage).

ACTION in action: To facilitate effective object detection at each camera node locally, we leverage state-of-the-art detection algorithms from the computer vision community. In brief, a sliding window (covering a block of pixels) is used on key frames to detect whether or not an object of interest is present in that block. Unfortunately, even these algorithms may suffer from false positives or false negatives due to occluded views where the objects are partially covered by other objects. Thus, as discussed below ACTION combines the information from multiple camera views to significantly enhance accuracy.

Object reidentification across cameras: The first challenge ACTION resolves is to determine if what is classified as an object of interest by one camera is also perceived to be the same object by another camera (or cameras) with an overlapping view. This is referred to as the re-identification problem (objects are re-identified across cameras). ACTION uses a novel method that maps the 2D view to a 3D location. This 3D location as perceived by the different cameras, is used jointly with the color features of the object to perform the re-identification.

Maximizing accuracy using multiple camera views given bandwidth constraints: Given the bandwidth constraints, ACTION does not exchange raw videos among the cameras. Instead, all nodes with overlapping views extract metadata with respect to detected objects, and send this metadata to a common node (called the fusion node). The fusion node jointly examines the metadata and first resolves the aforementioned re-identification problem. Next, the ACTION software at the fusion node identifies the combination of camera views of a particular object that yield the highest associated detection accuracy (lower false positive and false negative rates) while adhering to a timeliness constraint (that is determined by the bandwidth available for transferring the information). It essentially models the network as a knapsack, and the gain associated with each frame is dictated by the probability that an object is correctly detected in that frame. It then uses a greedy approach to select and send those frames that either satisfy the detection criterion, or fill the knapsack (sends all the frames that the bandwidth permits). It relays this information back to the cameras which transfer the actual frames.

Evaluations: We consider human detection (i.e., humans are the objects of interest) and evaluate ACTION's performance. Specifically, we emulate our scenario by implementing ACTION on Android devices preloaded with a dataset that consists of video sequences captured from 4 different cameras [3]. Our evaluations show that by considering views from multiple cameras, ACTION can detect $\approx 20\%$ more humans than when using the video from a single camera. Further, it achieves a very high accuracy rate of $\approx 90\%$, if an object is detected by at least 2 cameras (with a single camera it can be at most ≈ 72 %). In terms of resource usage, ACTION can reduce the bandwidth usage threefold, compared to uploading all the detected frames directly to the central controller. In addition, with ACTION, even though information from 4 cameras is used, the amount of transferred data is only ≈ 1.4 times higher than the amount of data transferred when one camera is used, while providing a significant higher detection accuracy.

II. Related work

We divide relevant related work into 4 main categories:

Object detection: We leverage state-of-the-art human detection algorithms from the computer vision community. These algorithms can be easily applied to detect other types of objects (e.g., vehicle, animals) with appropriate datasets. For human detection, different features [4][5] and machine learning techniques for building the detection model [6][7] have been proposed. We leverage the technique described in [7] to effectively detect humans in videos at individual camera nodes, in ACTION. Details on how we do so will be provided in Section III-B.

Object association across camera views: Object association refers to the identification of the same objects (e.g., humans) captured from overlapping camera views. In [8], people are assumed to stand on the ground; human positions are then computed by projecting the detected positions onto the ground-plane. Positions that are within a distance threshold on this plane, are considered as being associated with the same human. Implicitly they assume that the ground coordinate is known. However, we do not assume that this will be the case in ACTION; humans can stand on objects or on uneven ground. We provide a novel association mechanism (we call it re-identification) in such cases. In [9] and [10], color features of detected regions are leveraged (compared across cameras) to identify images of the same human. In ACTION, we jointly consider such color information with the 3D position information (obtained using our approach), relating to each human detected from different camera views, to achieve a high association or re-identification accuracy.

Detecting and tracking objects in multi-camera networks: In [11], multiple cameras collaborate to detect human heads; for each detected head position in an image, the associated 3D position of the head is estimated. Nearby head locations are identified and combined by comparing their Euclidean distances. Other efforts on object tracking [3][12] build complicated 2D to 3D mapping models which require high computational overheads. Since we seek to ensure low resource consumption, these cannot be applied in our framework. Further, data communications between the nodes is not a concern in these designs; in ACTION we seek to limit the amount of data transferred to a central controller.

Redundancy reduction in video transfers over networks: Recent related work considers the reduction of redundancy of content transferred over networks [13], [14]. In these efforts, metadata (or the entire video/image content) is exchanged between nodes and the controller to detect and suppress redundant content. However, they do not eliminate unnecessary content (e.g., views that do not contain relevant objects). They are also unconcerned about the accuracy of detection of such objects (as is the case with ACTION); in fact, some redundant content may be transferred in ACTION to improve the accuracy of object detection.

Zhang *et al.* [15] build a multi-camera surveillance system. In their system, if an object is detected by multiple cameras, only *a single* view containing the object is uploaded; *all* other

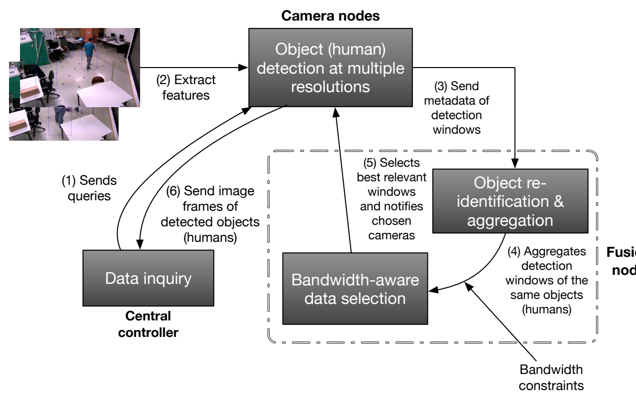
Camera nodes
Object (human) detection at multiple resolutions
(2) Extract features
(3) Send metadata of detection windows
(1) Sends queries
(6) Send image frames of detected objects (humans)
(5) Selects best relevant windows and notifies chosen cameras
Object re-identification & aggregation
(4) Aggregates detection windows of the same objects (humans)
Fusion node
Data inquiry
Central controller
Bandwidth-aware data selection
Bandwidth constraints

Fig. 1: Architecture of ACTION

views that contain the same object are treated as redundant and suppressed. They only consider sufficiently high resolution input video feeds; and thus, all objects are considered correctly detected. In ACTION, information from overlapped cameras is "aggregated" to improve the detection accuracy and views are chosen depending on the bandwidth constraints.

III. THE ACTION FRAMEWORK

In this section, we describe our framework, ACTION, for extracting and reporting situational-awareness information in bandwidth constrained multi-camera networks. To aid the narrative, we use human detection as a specific use case. In some parts of ACTION, therefore, we leverage state-of-the-art techniques in human detection from the computer vision community to identify the presence of humans from the video feeds. However, the same or very similar techniques can be used to detect other types of objects (e.g., vehicles, suitcases, animals)[2]. We begin with an overview and then, describe each of ACTION's modules in detail. Figure 1 depicts the architecture of ACTION.

A. Overview

The ACTION software is housed on three main components: camera nodes, a fusion node and a central controller. The camera nodes are deployed in the field. They may have overlapping views, i.e., a location on the field could be covered by multiple camera nodes. We assume that the camera nodes communicate with the central controller using either a cellular (i.e., 4G/LTE) connection, or a WiFi connection. The communications between individual camera nodes, and one of these chosen to be a fusion node is via a wireless channel (using 802.11 ad hoc connection temporarily set up at the scene).

We assume that a user who mans the central controller sends queries to the camera nodes. The queries seek the images of humans (the objects of interest in our use case) who appeared in the field of view within a specified period of time. Each camera node extracts features from the video sequences that it has captured towards detecting the presence of humans. When a human is found, metadata associated with the position of the human (we call this the detection window), is computed and sent to a fusion node. The content in the metadata includes information such as the timestamp, the probability (or confidence) that the detected object is a human, and location of the human in the snapshot image, etc.; more details are provided later in the section.

[2]This is because these techniques are based on machine learning and are trained using appropriate datasets collected in such settings; if an appropriate dataset with regard to the different types of objects are used to train the system, the algorithms can be used in these other cases.

Upon receiving all the metadata from the camera nodes sharing overlapping views, the fusion node first performs re-identification of the human object across the cameras (to determine which camera views correspond to the same human). We simply assume that the fusion node is one of the camera nodes which share overlapping views (this set can be determined by transmission of beacons). We arbitrarily select one of these nodes for collecting and aggregating the metadata; more intelligent algorithms can be used to choose the fusion node [16], but that is not the focus of this work.

After re-identification, the fusion node runs a greedy algorithm to identify the "most relevant" detection windows from the plurality of camera views. In brief, we formulate the problem as a variant of the classical Knapsack problem [17] and design an efficient greedy algorithm to solve it. The available bandwidth is equally shared among all detected humans. Thus, for each human, the bandwidth allocated dictates the size of the knapsack and the accuracy requirements determines which video frames are inserted into the knapsack (chosen for transfer). The fusion node notifies all the cameras about the windows that are selected for transfer by the greedy algorithm. Upon receiving this notification, the chosen camera nodes send the relevant windows directly to the central controller. We wish to point out that the fusion node only receives the lightweight metadata information; we avoid transferring the actual video data between nodes.

B. Object detection at individual cameras

Upon receiving a query from the central controller, the camera nodes process their locally stored media source (videos/ images) to locate images of humans to respond to the query. If the media source is a video sequence, in order to reduce the processing overhead, only the key frames or video frames at pre-specified intervals are processed for human detection. This interval typically should be chosen based on the dynamics of the setting.

We do not design new computer vision algorithms for object detection in ACTION. Rather, we leverage a state-of-the-art human detection technique proposed by Dollar et al. [7]. With this technique, a detection window of size 128x64 pixels is slid across the input image to check for the presence of humans (at different positions within the image). A detection model based on such a fixed size detection window, is only effective in detecting humans whose sizes are similar to that window size. However, humans that appear in a frame could vary in size depending on the distance between the camera and each such human. To overcome this problem, each input frame is scaled to different sizes (so that humans in the view are also scaled up and down by different factors) to try to match the size of the detection window. The sliding detection window is then applied to all sizes of the input image. A detection hit on a "resized" version of the image will be mapped onto the corresponding position in the original sized frame.

When the sliding window is moved to a new position, the pixel values inside the window are computed and used as features for the human detection process. Specifically, the image is transformed into 10 different representations (called image channels). These include 3 color channels in the LUV

Algorithm 1 Adaboost learning process

```
Inputs:
  • Training samples $\{x_1, y_1\}, ..., \{x_n, y_n\}$ where $y_i = \{-1, 1\}$ indicates a negative and a positive sample, respectively
  • T: number of learning steps
  • $N$, $N'$: number of positive and negative training samples, respectively
Initialize: For each item $x_i$, set weight $w_{0,i}=\frac{1}{2N}$, $\frac{1}{2N'}$ for positive and negative samples respectively
 1: for $t = 1$ to $T$ do
 2:     Normalize weights: For each item $x_i$, set $w_{t,i} = \frac{w_{t,i}}{\sum_i w_{t,i}}$
 3:     for each feature $f_j$ do
 4:         Train a weak classifier $h_j$ as in Eq (1)
 5:         Compute the error rate $\xi_j = \sum_i w_{t,i} I(h_j(x_i) \neq y_i)$, where I is the identity function
 6:     end for
 7:     Choose the weak classifier which has the lowest error rate $\xi_t$
 8:     Set $\beta_t = \frac{\xi_t}{1-\xi_t}$, and $\alpha_t = -log\beta_t$
 9:     for each learning sample $x_i$ do
10:         Update its weight: $w_{t+1,i} = w_{t,i}\beta_t^{1-e_i}$; where $e_i = 0$ if sample $x_i$ is classified correctly, otherwise $e_i = 1$
11:     end for
12: end for
Output: The final strong classifier $H(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$
```

color space, 1 gradient magnitude channel and 6 gradient orientation channels (for more details refer to [5]). Each pixel in each channel is used as a feature associated with the detection window; therefore there is a total of 128x64x10 features (corresponding to the window size chosen). To reduce this high number of features, the Adaboost algorithm [18] is used (discussed below).

First, a training set which includes "positive" image windows (those with humans) and negative image windows (non-human images) is built offline. All the samples in this training set are resized to 128x64 pixels and have the same number of features. Subsequently, the Adaboost algorithm is applied to learn the classification rule as briefly captured in Alg. 1. The number of learning steps T is pre-defined. In each step t, a *weak classifier h* is learnt. Thus, at the end of T steps, we have T weak classifiers. The final *strong classifier H* is constructed as a linear combination of these T weak classifiers.

In its simplest form, a weak classifier h_j is a classification rule (for a detection window x) defined based on a single feature f_j. It is defined based on whether or not f_j (a positive value) is higher or lower than an associated (prespecified) threshold τ_j and a polarity ($\theta_j = \pm 1$, chosen to reflect the correct inequality), as:

$$h_j(x) = \begin{cases} 1 & f_j(x).\theta_j \leq \tau_j.\theta_j \\ -1 & otherwise \end{cases} \quad (1)$$

where, $h_i(x) = 1$ indicates that x is a positive window and a negative window otherwise. The polarity θ_i is used to determine the correct inequality i.e., whether the value of $f_i \leq \tau_i$ indicates a positive window, or otherwise.

Let N' and N be the number of negative (no human) and positive (with human) learning samples, respectively. Initially, all the positive samples x, are assigned the same weight, $\frac{1}{2N}$. The weights of the negative samples, x' are set to $\frac{1}{2N'}$. At each step t, the feature which produces the lowest error rate ξ_t (discussed below) is chosen as the weak classifier.

For each feature, the error rate is the sum of the weights of all the samples x for which $h_i(x) \neq y(x)$, where $y(x) = \pm 1$ is the label of training sample x. At each learning step, the key idea of Adaboost is to decrease the weights of those samples that are correctly classified. Specifically, the weight of sample x_i is kept unchanged if it is incorrectly classified. If x_i is correctly classified, its weight is reduced (adjusted *down*) by a factor of β_t that is a function of the error (line 8 in Alg. 1). Thus, the weights of incorrectly classified samples are higher than those samples that are correctly classified.

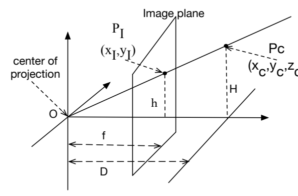

Fig. 2: Camera intrinsic information

The strong classifier H will be distributed to the camera nodes to detect humans in their captured videos/images.

C. Putting things together: Jointly considering overlapping views

When an individual camera believes that it has detected a human, it extracts metadata pertaining to the detection window in which the human is detected. It then sends this metadata to a fusion node. The fusion node is simply one of the camera nodes which share the overlapping views as described earlier. The metadata associated with each detection window includes (i) A timestamp which indicates when the human appears in the field of view, (ii) The location (a rectangular bounding box) of the detected human in a 2D image coordinate system (as discussed later, this location is converted to a location in the real 3D world using a novel approach), (iii) The probability that the human is actually captured in that window P_i; we refer to this as the detection probability. P_i is computed from the value of the strong classifier H (described in section III-B), (iv) a compact color feature of the detection window, and finally, (v) the size of the detection window.

At the fusion node, the goal is to combine the information from the detection windows obtained from the different cameras that are associated with the same human. In order to do so, the fusion node converts the 2D locations from the detection windows in the image coordinate system to a location in the 3D real world coordinate system. These locations with respect to the plurality of cameras are then compared towards re-identifying the human in different camera views. If the locations of two windows are within a threshold T_p, we consider that the two windows are most probably associated with the same human.

In many cases, there could be several people standing close to each other; in order to verify and reduce the false matches in such situations, we also compare the color features from the detected windows. If the distance associated with their color features are also within a threshold (T_c), the fusion node concludes that the two windows depict the same human.

In the following, we provide details on how ACTION computes the 3D location in the real world and uses the color features in association.

Estimation of the 3D location of the detected human: We use the camera calibration information of an individual camera to *estimate* a 3D location $P_w(x_w, y_w, z_w)$ of a human in the real world coordinate system (with a pre-defined origin and axes agreed upon by all cameras) from its 2D location $P_I(x_I, y_I)$ in the image coordinate system. The camera calibration information consists of the intrinsic information $\mathbf{K}$, the rotation information $\mathbf{R}$ and the translation information $\mathbf{T}$ of the camera [19]. $\mathbf{R}$ provides information with regards to the angle the camera is tilted with respect to the three axes in the real world coordinate system. $\mathbf{T}$ provides information about the location of the camera itself in the real world coordinate system. $\mathbf{R}$ and $\mathbf{T}$ form the extrinsic information of the camera.

The $\mathbf{R}$ and the $\mathbf{T}$ matrices are used to convert the 3D location $P_w(x_w, y_w, z_w)$ of a human in the real world coordinate system to a 3D location $P_C(x_C, y_C, z_C)$ (of the same human) in the camera coordinate system where, the camera itself is the origin and the axes are tilted or rotated in conjunction with the camera. This is captured in Eq 2:

$$P_C = \mathbf{R} * P_w + \mathbf{T} \tag{2}$$

where, "$*$" represents the product operation (of the matrices).

The intrinsic matrix $\mathbf{K}$, which contains information about the camera "projection point" and the focal length, helps convert the 3D location P_C into a 2D location P_I in the image plane, as shown in Fig. 2. In essence, it is simply a projection of the 3D object onto the 2D plane in the camera's view.

The task of converting the position of the detected human in the image plane to a position in the real world coordinate system consists of the following steps: (i) convert the 2D location P_I in the image plane to a 3D location P_C in the camera coordinate system, and (ii) convert the 3D location P_C to the 3D location P_w in the real world coordinate system. It is a challenge to accurately compute P_C because of the lack of the third dimensional information when converting a point from a 2D world to one in a 3D world. As evident from Fig. 2, multiple 3D objects can project to the same 2D object in the image coordinate system (e.g., any point along the line connecting P_I and P_C is projected onto the same point P_I in the image plane). Note here that the distance D between the detected human and the camera is not known.

In ACTION, we estimate the value of P_C by considering a plurality of human or object heights and determining whether or not two camera views converge in their 3D location estimates for *any* of these heights. In more detail, let h be the height of the detected human in the image coordinate system (the height of the detection window). Let H be the height of the human in real world. D is computed as:

$$\frac{D}{f} = \frac{H}{h} \tag{3}$$

where f is the distance between the center of projection (see Fig.2) of the camera to the image plane (this is the camera's focal length and is provided in the camera's intrinsic information $\mathbf{K}$). Once D is known, P_C can be computed from P_I easily.

We assume a set of possible values of the height of the detected human, $H = \{H_1, H_2, ...H_n\}$ (e.g., from 3ft to 6ft, etc.)[3], and come up with the corresponding values of D_j using Eq 3 (for each $H_j, j \in \{1, n\}$). The values of P_I (the 2D location) and each D_j (distance to the camera) are then used to compute the possible values of $P_C = \{P_{C1}, P_{C2}, ..., P_{Cn}\}$, as depicted in Fig. 2. Finally, the corresponding values of P_w are computed for each P_{Cj}, using Eq 2.

Each camera node sends its intrinsic information (including the focal length), location and orientation computed based on the reference world coordinate system (which are used to calculate $\mathbf{R}$ and $\mathbf{T}$ [19]) to the fusion node[4]. The camera nodes update the fusion node if the positions or orientations of the cameras change. With respect to each detection window, its size and the coordinates P_I of the center point of the rectangular window are also included as metadata. This allows the fusion node to compute P_w, with respect to the window for each considered $H_j, j \in \{1, n\}$. Subsequently, the fusion node groups the "nearby" windows (if the Euclidean distance between the P_w values, for any j, is less than a pre-defined threshold T_p) from multiple cameras, together into candidate sets. Note that this requires a pairwise comparison of the P_ws from each camera for all values of j.

Using color and texture features: Unfortunately, humans in the field can stand close to the others and this can lead to false matches, i.e., wrong inferences can be made with respect to re-identification, if only the Euclidean distances were considered (as computed in the above discussion). Therefore, ACTION also compares the color and texture features in the detected windows of the candidate sets to reduce the number of false matches.

We use the "Mean Color" feature proposed by Hirzer et al. [21], which is extracted from each detection window in the above step. Each detection window is resized to 64 $\times$ 128 pixels to ensure that the sizes of the color features of all windows are the same, regardless of the original sizes of the windows (which reflect the sizes of the objects captured in the frames). After this step, the mean color feature of a detection window is a 55,000-dimensional vector. Given this large dimension, Euclidean distance is not an effective measure for comparing the feature vectors between detection windows. First, we use principal component analysis (PCA) to reduce the dimensionality of the color features. Subsequently, we use the Mahalanobis distance (proposed in [22]) to compare the features. The Mahalanobis distance between two vectors x_i and x_j is defined as:

$$d(x_i, x_j) = \sqrt{(x_i - x_j)^T \mathbf{A} (x_i - x_j)} \tag{4}$$

where $(x_i - x_j)^T$ is a transposed matrix, and $\mathbf{A}$ is called the Mahalanobis matrix and is learnt from a training set. Our training set consists of two subsets. The first subset S, contains images of the same humans (with the same timestamp) collected from different cameras; the second subset D contains images of different humans (with the same timestamp) collected from different cameras. Matrix $\mathbf{A}$ is trained to minimize the distances between the elements in S while maximizing the distances between the elements in D. The problem is formulated as a constrained optimization problem, and an iterative framework based on a binary search is used to find an optimal matrix $\mathbf{A}$ (see [22]). If the Mahalanobis distance between the color features of two detection windows (which have been already matched up with respect to position) is less than or equal T_c, we assume that the humans in the two windows are the same.

[3]If ACTION is used to detect other types of objects, appropriate heights of the objects (e.g., 4ft to 6ft for sedan cars) should be used.

[4]Camera calibration information for distributed fixed camera networks can be effectively obtained using prior approaches on computer vision (e.g.,[20]).

Algorithm 2 Greedy algorithm for choosing detection windows for a detected human

```
Inputs:
   $\tau$: Transfer time for this human
1: Remove detection windows $W_i$ whose $\frac{size(W_i)}{B_i} > \tau$
2: Sort remaining detection windows (in descending order) by "profit densities" $PD_i = \frac{|\log F_i| B_i}{size(W_i)}$
3: Compute k=min{j∈1,...,n}: $\sum_{i=1}^{j} \frac{size(W_i)}{B_i} > \tau$
4: Compute $V_1^{k-1} = \sum_{i=1}^{k-1} |\log F_i|$ and $V_k = |\log F_k|$
5: if $V_1^{k-1} > V_k$ then
6:     Choose $\{W_1, ..., W_{k-1}\}$
7: else
8:     Choose $W_k$
9: end if
```

D. Transferring the most relevant information given bandwidth constraints

We assume that the bandwidth from each camera node to the central controller is known (Tools such as iPerf can be used for short durations for determining this [23]). We assume that the bandwidth information is also conveyed to the fusion node as part of the metadata. For each detected human, we seek to select and transfer the most relevant camera views sufficient to achieve a desired detection probability P (e.g., $P = 0.9$), within a pre-specified delay. However, achieving the detection probability and this delay simultaneously may be impossible depending on the available bandwidth. Thus, we modify our objective to maximizing the detection probability (of humans), subject to an available bandwidth constraint, which is formalized in Eq 5.

$$maximize_{P_i \in \{0,1\}} \quad P = 1 - \prod_i (1 - P_i) = 1 - \prod_i F_i \quad (5)$$
$$\text{subject to} \quad \sum_i \frac{size(W_i)}{B_i} \leq \tau.$$

In Eq (5), P_i, $F_i = 1 - P_i$, and $size(W_i)$ are the detection probability, false detection probability and size of a detection window W_i (in bytes) from each detected human, respectively; the index i varies over all views of that human. τ is the delay requirement specified in seconds and B_i bytes/second is the bandwidth from camera i to the central controller (note that the channel conditions and contention would dictate this bandwidth). The value of P is computed based on detection probabilities associated with the windows considered for transfer. Specifically, if multiple detection windows W_i agree on a human presence at a given place, at the query specified time, the probability of a false detection at that location (when all cameras yield incorrect results) is $F = \prod_i (1 - P_i) = \prod_i F_i$; thus, $P = (1 - F)$. Thus, those windows which maximize P, are chosen for transfer, while adhering to the bandwidth allocated.

The objective in Eq (5) is equivalent to the minimization of $\prod_i F_i$, $F_i \in (0, 1]$ which in turn, is equivalent to the minimization of $\sum_i \log F_i$, for all $F_i \in (0, 1]$, with the same bandwidth constraint as before. Since $\log F_i \leq 0$ for all possible values of F_i (F_i values are less than 1 since they represent probabilities), the minimization of $\sum_i \log F_i$ is equivalent to the maximization of $\sum_i |\log F_i|$ (since $\log F_i$ is negative and monotonic). Thus, the optimization problem in Eq (5) is equivalent to the following problem in Eq 6.

$$maximize \quad V = \sum_i |\log F_i| \quad (6)$$

Algorithm 3 Algorithm for sending best relevant detection information to the central controller

```
Inputs:
   $N$: Number of humans present in the field
   $B_j$: Available bandwidth of each camera $j$
   $P$: Desired detection probability for each human
   $\tau$: Delay requirement //for all detected humans

Initialize: Remaining humans, $n = N$

    for each human $H_i$, $i = 1$ to $N$ do
2:      Available transmission time $\tau_i = \frac{\tau}{n}$
        False detection rate $F_{Hi} = 1$
4:      Select detection windows using the Greedy algorithm (Al. 2) with corresponding $\tau_i$
        for each selected windows $W_{ij}$ do
6:          Send $W_{ij}$ //view $j$ for human $i$
            $\tau_i = \tau_i - \frac{size(W_{ij})}{B_j}$
8:          $F_{Hi} = F_{Hi}(1 - P_{ij})$ //update false detection rate
            if $F_{Hi} \leq (1 - P)$ then
10:             break; move to next human;
            end if
12:     end for
        $\tau = \tau - (\frac{\tau}{n} - \tau_i)$ //share leftover time for other humans
14:     $n = n - 1$ //number of humans need to send data
    end for
```

$$\text{subject to} \quad \sum_i \frac{size(W_i)}{B_i} \leq \tau$$

The above problem can be mapped onto a Knapsack problem [17]. The maximum tolerable delay (specified) for transferring the information pertaining to each user, to the central controller, corresponds to the knapsack size. The goal of the fusion node then, is to choose the views that maximize the sum of the utilities ($|\log F_i|$) of the objects (camera views) that are placed into the knapsack. Unfortunately, the problem as defined above is known to be NP hard [17]. Therefore, ACTION uses a well known greedy algorithm (detailed in [24]) to fill the knapsack. As the name suggests, the algorithm greedily chooses the most relevant windows from the multiple cameras for being sent back to the central controller in response to its query, while adhering to the delay constraint given the bandwidth.

With the greedy algorithm, the detection windows are sorted (in descending order) at the fusion node, in terms of their "profit densities" which are defined as $PD_i = \frac{|\log F_i| B_i}{size(W_i)}$. Here, note that in the general case, the detection windows from the different camera nodes vary in terms of the number of bytes. The knapsack is filled from this sorted list; the view with the highest profit density is inserted first and so on. Let W_k be the first window (as the list is traversed) that causes a violation to the bandwidth constraint; here, k=min{j∈1,...,n} such that $\sum_{i=1}^{j} \frac{size(W_i)}{B_i} > \tau$. The greedy algorithm then chooses either the set of windows $\{W_i, ..., W_{k-1}\}$ or the single window W_k, depending on whether the value $V_1^{k-1} = \sum_{i=1}^{k-1} |\log F_i|$ is higher or lower than $V_k = |\log F_k|$, respectively. It is shown in [24] that the profit V_G obtained using the greedy algorithm ($V_G = max\{V_1^{k-1} = \sum_{i=1}^{k-1} |\log F_i|, V_k = |\log F_k|\}$) is guaranteed to be $\geq \frac{1}{2} V_{OPT}$, where V_{OPT} is value of the Knapsack when filled optimally [24]. The pseudocode for the greedy algorithm is shown in Algorithm 2.

Operations in Practice: The pseudocode of our bandwidth-aware data selection process at the fusion node is presented in Algorithm 3. Specifically, we divide the available time equally among all of the humans that are detected[5]. For example, if a 1 second period is available to transfer the information and there are 3 humans detected, we allocate 0.33 seconds to transfer the detection windows associated with each human. For each human H_i, we apply the greedy algorithm (Algorithm 2) with available transfer time τ_i of that human to select best relevant windows to fill the knapsack. The relevant detection windows are transferred in order (we assume that the fusion node tells each camera node when and what frames to transmit) until one of two conditions is met (i) the duration for sending data with respect to that human expires, or (ii) the required detection accuracy with respect to this human has been met.

We modify the Knapsack problem to ensure that we do not waste bandwidth if the detection probability is higher than a sufficient (desired) value. In other words, if a desired P is achieved (i.e., the detection accuracy is met) but there is leftover bandwidth (time) after the associated, "sufficient" set of detection windows relating to the human are transferred, the residual time (bandwidth) is equally shared for the transfer of windows with respect to the other humans, as shown on line 13 of the algorithm.

IV. Implementation

In this section, we describe the implementation of ACTION. Our implementation consists of a fusion application, a detection application and a server application. The fusion application runs on a pre-specified Android phone and receives metadata relating to the detection windows, from multiple instances of the detection application that are in turn running on Android smartphones. The detection applications interact with the fusion application, receive fusion decisions from the fusion node and finally upload chosen detection windows to a server application which represents the central controller.

The detection application: We implement our detection application on Asus ZenFone II smartphones with Android 5.0 OS. These are used as the camera nodes in our prototype. The application was written in both Java and native C++ (JNI). We partly use the source code provided by the authors of [7], to convert the input image into different channels and to extract its features for human detection (as described in III-B). We use the OpenCV library for all other image processing tasks.

Note here that we use smartphones, as they are popularly used for video capture today [25], to *emulate* camera nodes with local processing capabilities. In a real scenario, one could envision static, programmable cameras (e.g., Pixy [26]) that are mounted on ceilings or walls are used. Such cameras could capture higher resolution videos; however, we believe that today's smartphones already offer very high resolutions and devices that are similar architecturally can be used for this purpose. However, we acknowledge that with other platforms, the results could differ from those that we present in Section V. For example, other platforms may have more powerful batteries and processing capabilities.

An input image is scaled into 23 different sizes for human detection. In order to reduce the computational overheads, we only compute image features at 3 specific base scales; image features at other sizes are estimated based on the features determined at the base scales, as described in [27].

Camera node operations: Upon receiving a query from the server application, our detection application reads the input video sequence and checks for humans in every 10^{th} frame (once every 0.5 seconds). In practice, this parameter should be chosen based on the area of the monitored field, number of deployed cameras and the moving speed of tracked humans. The value 0.5s chosen in our implementation is based on the availability of the ground truth information in our data set [3], which is described later in section V. When a human is found, the application extracts the metadata associated with the detection window as described in section III-C and uploads this information to the fusion node.

The output of the human detection algorithm is a detection score of the window. The higher the score, the higher the probability that the window contains an image of a human. We convert the detection score to a corresponding detection probability using the training data as follows. The detection scores are quantized into 20 bins. For each bin value x, the probability is given by the ratio of the number of *correct* detection windows (based on the ground truth information) to all the detection windows that have a detection score of at least x.

To extract the color features from the detection windows, we first resize all the windows into 64x128 pixels. This ensures that the color features of all detection windows will have the same length. The extracted mean color feature of each window is a 55,000 dimensional vector. To reduce the communication overhead, we apply PCA to reduce this size to 40 dimensions, as in [21], before uploading the information to the fusion node. Further, we compress the detection window (using the jpeg format) and use the compressed version for transfer.

The fusion application: The fusion application is written in Java and can be executed on any Android smartphone. Currently, we statically assign one of the camera nodes to act as the fusion node; however, in practice, nearby camera nodes can be grouped into clusters, and for each cluster, the fusion node can be chosen based on which of the nodes has highest computational power or residual energy.

Fusion node operations: The fusion node will receive metadata with regards to detection windows from multiple surrounding cameras. For each detection window, the fusion considers a set of 17 possible heights of common humans (from 120 cm to 200 cm) and converts the 2D location in the image to a set of 17 possible 3D locations of the detected human. Locations and color features of the detection windows are used to detect and associate windows that capture the same human. Specifically, we use the threshold $T_p = 1.2m$ for location, and $T_c = 18,000$ for color (Mahalanobis) distance to group detection windows. We show the effectiveness of using these values in Section V.

When the metadata with regards to a new detection window arrives, the fusion node checks to see if it can be correlated with the metadata from that of another camera node. If it cannot find a correlation with any previously received metadata, the fusion node considers the window to be the first window of a newly detected human and creates a new group for it. When metadata associated with other windows corresponding to the same human arrive, they will be grouped together. Locations of individual windows in a group are averaged to compute the

[5]In ACTION, a record with respect to a human is put into a queue when all the information related to that human is available at the fusion node. Information relating to the human whose associated record has the earliest time stamp is considered first for transfer. Other policies are possible (prioritization of records as per other criteria) but we do not consider this.

location of the centroid position, which is used to represent the whole group. For each group, there are multiple positions P_w of the centroid corresponding to the considered heights, H_j; in order to be considered to belong to a group, the newly received metadata must reflect a position that is within the threshold T_p with respect to one of the n, P_w values (i.e., corresponding to each of the heights H_j). Further, the distance between the color feature of the window and that of at least one of the windows in the group must be within T_c.

In cases when one detection window can be grouped into different groups, the fusion nodes includes the new window in the group which has the minimum Mahalanobis color distance to the window, since "color distance" is more distinctive than "position distance" if people are close to each other (these are the primary reasons for errors in reidentification based on position alone).

After partitioning the detection windows into groups, the fusion node chooses the most relevant windows from each group based on the greedy algorithm described in section III-D, and notifies the camera nodes if they have the chosen windows. Subsequently, the camera nodes transfer image data directly to the central controller.

The server application: The server application runs on a Linux server; its only duty is to send queries with specific time-stamps, and receive content (parts of frames with detected humans) from the camera nodes after being instructed to do so by the fusion node.

V. Evaluations

In this section, we evaluate both the accuracy as well as the efficiency (in terms of resource consumption) of ACTION. We begin with a description of the datasets we use and how we determine the ground truth. Later, we provide our results.

A. Training and test datasets

1) Data set

We use the "Multi-camera multi-object tracking" data set, made available by the Computer Graphics and Vision group at Graz University of Technology [3]. The dataset contains 6 indoor video scenarios. Each scenario has 4 different synchronized video sequences, captured by 4 different cameras. In each video, there are about 4 to 6 people walking around in the same room. At different times in the videos, the people can be separated spatially or could be closely clustered together. We use the first video sequence as the training set to calibrate ACTION, and the other sequences as the test data.

The human detection model is built using the Inria pedestrian dataset [4]. This contains thousands of images of humans in different poses. Thus, the detection model is not calibrated by the small set of humans that appear in our first data set, and can be used to detect humans in general cases.

2) Obtaining ground-truth information

The data set that we consider is annotated with information that provides ground truth at a coarse-grained level. Specifically, once every 10 frames, the 3-D, real world coordinates of the "foot" of each person in the frame, is provided. As discussed earlier, we can convert these co-ordinates to the 2-D coordinates in the image coordinate system of each camera (using Eq (2)). The data set also provides the identifier of the human the foot belongs to (human ID).

Each detection window is a rectangular area within the frame as discussed earlier. We check if the aforementioned 2D coordinates (of the "foot") fall within each detection window. Each detection window which contains such coordinates, is associated with a human ID. In cases where there is more than one human in the detection window, we may have false positives. To eliminate these, we mark those windows and manually check the ground truth information (to determine which human is in the detection window).

B. Improving detection accuracy with overlapped camera views

1) Setting thresholds for accurate detection

Below, we first describe how we determine the thresholds T_p (the Euclidean distance between the plausible 3D locations of a detected human, from the perspective of multiple cameras) and T_c (the Mahalanobis distance between the color features that correspond to the same human, from different cameras).

From the training video sequence, we create two different sets: (a) set S contains pairs of detection windows that show the same human, captured at the same time by different cameras, and (b) set D contains pairs of windows that show different humans captured by different cameras. For each detection window, we compute the possible positions of the human (based on a set of considered heights) as described in section III-C, and compute the minimum Euclidean distance (with respect to all considered heights) between each pair of windows in set S and in set D. We show the CDF of the distances in Fig. 3. Based on the results, we set threshold $T_p = 1.2m$; from the figure we see that this results in the detection of more than 80% of the windows of the same human with about $\approx 30\%$ false matches. An increase in this threshold would result in a higher false positive rate; a decrease would reduce the number of correct detections (true positives). This seems like a reasonable compromise. Next, we compute the Mahalanobis distance between the color features of each pair in set S, and in set D. We show the CDF of this distance in Fig. 4. Based on the results, we set threshold $T_c = 18,000$. With this threshold, we are able to detect more than 90% of windows with the same human, with an expense of $\approx$25% false matches (false positives).

We point out that it is important to achieve a high true positive rate while keeping the false positive rate low. The former would reduce bandwidth usage as fewer windows of the same human need to be transferred; however, incorrect matching of different humans might cause the missing of the transfer of data associated with a particular person. *When both the position and color are combined, we are able to achieve a true positive rate of $\approx$ 91% and a false positive rate (when windows containing different humans are incorrectly grouped) of $\approx$ 9 %.* ACTION classifies a group of detection windows as "correctly matched," based on a majority rule. If at least half of the detection windows correspond to the same human (based on the ground truth information), the matching is considered correct; otherwise it is considered incorrect.

2) Does the use of more cameras yield better performance?

Next, we perform experiments to determine the benefits of using a plurality of cameras in ACTION. One of the metrics we use is what we call the *recall* value. This value is the number of times that a human is correctly identified from among all the times she appears in the captured videos (transferred to the central controller). It is expressed as a percentage. If multiple cameras are used, at least one of the cameras needs to correctly detect the human. We measure how the recall

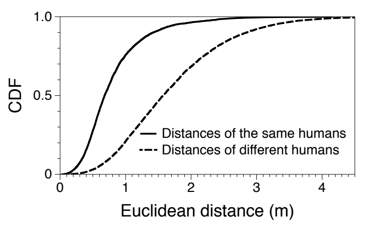

Fig. 3: Distances between windows of the same and different humans

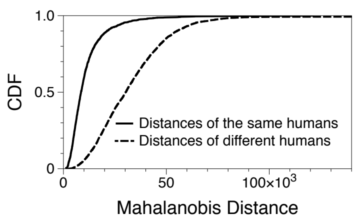

Fig. 4: Distances between color features of the same and different humans

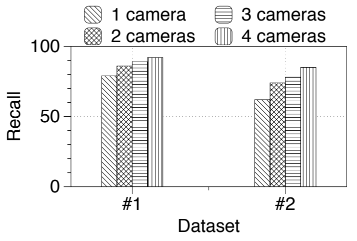

Fig. 5: Average recall values with different numbers of cameras

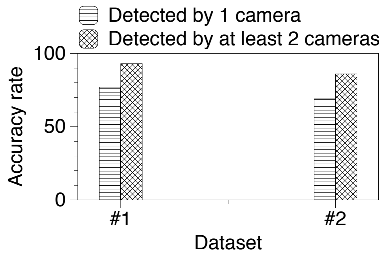

Fig. 6: Accuracy rate when combining data from more than one camera

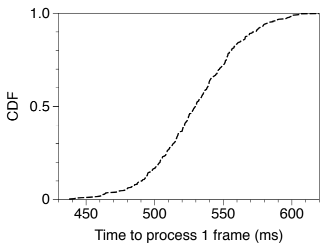

Fig. 7: Image processing time

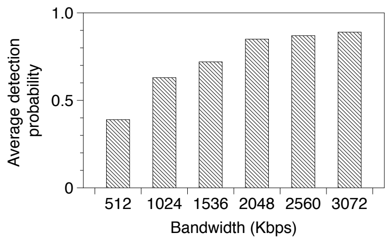

Fig. 8: Average detection probability under different bandwidth constraints

value changes, when different numbers of cameras are used for human detection.

When only one camera is used, all its detection windows that reflect the presence of a human are transferred. If multiple cameras are used, we posit a requirement of 0.9 on the detection probability P. In other words, the fusion node requires the transfer of detection windows until this requirement is met or the time constraint imposed (by the bandwidth requirement) does not allow any additional transfers. In these experiments, that time constraint is set to 0.5 seconds.

In Fig. 5, we show the results from both of our data sets. As one might expect, the use of a higher number of cameras results in a higher recall value. However, the improvements depend on the extent to which humans are occluded from camera views. In the first data set, humans are separated with little occlusion. Thus, the increase is only modest as seen in the figure. However, in the second dataset, the people are close to each other and typically there is higher occlusion. Here, the use of a plurality of cameras with ACTION results in a significant performance improvement. Specifically, with only one camera, the recall value is $\approx$ 64 %. It increases to more than 85 % with four cameras.

We next evaluate ACTION in terms of the *precision* of detection (aka the accuracy rate). Specifically, from among all the detection windows reported to detect humans, the accuracy rate represents the fraction that are correct reports. In the case of multiple cameras, we require that at least two of them correctly identify the (same) human (majority rule is applied as discussed earlier). We again observe that the use of multiple cameras significantly improves the accuracy rate. With dataset 1, the improvement is about 15 % while with dataset 2, the improvement is about 20 %.

3) Impact of bandwidth constraints

In our next experiment, we illustrate how the greedy algorithm for transferring relevant detection windows in ACTION, performs as we vary the available bandwidth. The individual nodes process the video sequences and upload metadata to the fusion node once every 10 frames. The fusion node determines the set of detection windows to be uploaded (as described in Section III-D). The corresponding cameras are required to transfer the selected information in 0.5 seconds. For ease of disposition, we assume that the system is homogeneous i.e., the bandwidth between the controller and each camera node is identical (we set this in our implementation). However, the results can easily carry over to heterogeneous settings. We again posit a requirement of 0.9 on the detection probability.

The results, presented in Fig. 8, show that under strict bandwidth constraints only a small fraction of the detection windows associated with each human is transferred. Thus the detection probability requirement is not met; the achieved detection probabilities are really low. However, as more bandwidth is available, a higher number of detection windows associated with each human can be transferred. The detection probability increases gradually. However, there is a "saturation point," (available bandwidth = 2048 kbps) after which there is sufficient bandwidth to transmit enough windows for achieving the required detection probability P; beyond this point there is a negligible improvement in P (if at all), even with a bandwidth increase.

C. Resource usage

In this section, we seek to evaluate ACTION in terms of quantifying the bandwidth savings that it provides and the processing overhead.

Bandwidth usage: Fig. 9 shows the bandwidth usage in three different scenarios: (i) when images of all detection windows from a single camera, are transferred directly to the command node, (ii) when all 4 cameras transfer images of all detection windows directly to the command node, and (iii) ACTION is used and a target detection probability of $\approx 90\%$ is required with regards to each detected human. In the former two scenarios, ACTION is not used. We show the results for the case where the input video is processed every (i) 0.5s and (ii) 2s, as in the previous experiment. Without ACTION, the volume of data transferred by the 4 cameras is around 3 times as compared to when ACTION is used. With ACTION, typically, only data from the best 1 or 2 cameras need to

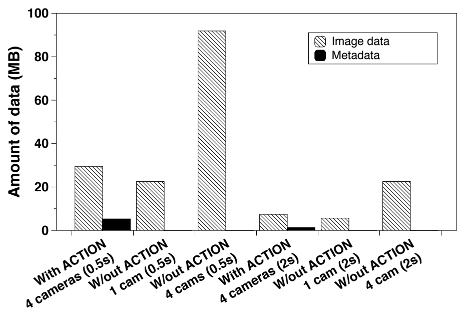

Fig. 9: Bandwidth usage

be transferred; thus a significant amount of unnecessary data transfers is avoided. Note here that if more than 4 (overlapping) cameras are available, one could conceivably achieve even a higher reduction in bandwidth usage. Further, note that with ACTION (since only the most relevant information is transferred), the total amount of transferred data (from 4 cameras) is only $\approx$ 1.4 times the data transferred by 1 camera.

Processing times on individual nodes: Fig 7 shows the distribution of the time taken to process 1 frame towards detecting a human on our Android smartphone. The resolution of our test data is 1024x768 pixels. With this setup, the maximum processing time is $600ms$; in other words the local algorithm for human detection on our smartphone-based individual camera nodes in ACTION, achieves a processing rate of $\approx 1.7 frames/sec$. This in turn suggests that a platform such as a smartphone is sufficiently powerful to process the video in near real-time. Since video frames can be processed well in advance of a query, we believe that the system is sufficiently responsive and deployable in real contexts.

VI. Conclusions

In this paper, we consider the problem of retrieving situation awareness information from a multi-camera network in scenarios such as natural disasters where the bandwidth is limited due to compromised infrastructure. In such scenarios, the cameras cannot all transfer their content to a central controller handling search and rescue operations. Thus, we seek to only transfer those camera feeds that can provide highly accurate input to the controller while ensuring the timeliness of the transferred content. Towards this, we design and implement a framework, ACTION. ACTION (i) uses state of the art computer vision algorithms to detect objects of interest (e.g., humans or animals), (ii) uses novel approaches to jointly consider views from multiple cameras and determine the views that yield the best accuracy with respect to an object of interest, and (iii) only transfers the best views to a controller while adhering to bandwidth constraints. We implement ACTION on a smartphone based testbed and show that it achieves a high accuracy of ≈ 90 % in terms of detecting humans who are considered as objects of interest, while reducing the bandwidth consumption threefold.

VII. Acknowledgment

This work was partially supported by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-09-2-0053. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on. This work was also partially supported by the NSF CPS grant 1544969.

References

[1] "Nepal earthquake death toll reaches 8,635, over 300 missing," Press Trust of India, 2015.

[2] A. M. Townsend and M. L. Moss, "Telecommunications infrastructure in disasters: Preparing cities for crisis communications," Center for Catastrophe Preparedness and Response, NYC, 2005.

[3] H. Possegger, S. Sternig, T. Mauthner, P. M. Roth, and H. Bischof, "Robust real-time tracking of multiple objects by volumetric mass densities," in *CVPR*, 2013.

[4] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *CVPR*, 2005.

[5] P. Dollar, Z. Tu, P. Perona, and S. Belongie, "Integral channel features," in *BMVC*, 2009.

[6] Q. Zhu, S. Avidan, M.-C. Yeh, and K.-T. Cheng, "Fast human detection using a cascade of histograms of oriented gradients," in *CVPR*, 2006.

[7] P. Dollar, R. Appel, S. Belongie, , and P. Perona, "Fast feature pyramids for object detection," in *PAMI*, 2014.

[8] X. Dai and S. Payandeh, "Geometry-based object association and consistent labeling in multi-camera surveillance," in *IEEE CAS*, 2013.

[9] K. Nummiaro, E. Koller-Meier, T. Svoboda, D. Roth, and L. V. Gool, "Color-based object tracking in multi-camera environments," in *DAGM*, 2003.

[10] M. Tan and S. Ranganath, "Multi-camera people tracking using bayesian networks," in *ICICS*, 2003.

[11] H. Iwaki, G. Srivastava, A. Kosaka, J. Park, and A. Kak, "A novel evidence accumulation framework for robust multi-camera person detection," in *ICDCS*, 2008.

[12] H. Possegger, T. Mauthner, P. M. Roth, and H. Bischof, "Occlusion geodesics for online multi-object tracking," in *CVPR*, 2014.

[13] U. Weinsberg, Q. Li, N. Taft, A. Balachandran, V. Sekar, G. Iannaccone, and S. Seshan, "CARE: Context aware redundancy elimination in challenged networks," in *ACM HotNets*, 2012.

[14] T. Dao, A. K. Roy-Chowdhury, H. V. Madhyastha, S. V. Krishnamurthy, and T. La Porta, "Managing redundant content in bandwidth constrained wireless networks," in *ACM CoNEXT*, 2014.

[15] T. Zhang, A. Chowdhery, P. V. Bahl, K. Jamieson, and S. Banerjee, "The design and implementation of a wireless video surveillance system," in *ACM MobiCom*, 2015.

[16] Y. Wang, Q. Wang, Z. Jin, and N. Saxena, "Improved cluster heads selection method in wireless sensor networks," in *IEEE GreenCom*, 2010.

[17] H. Kellerer, U. Pferschy, and D. Pisinger, "Knapsack problems," 2010.

[18] P. Viola and M. Jones, "Robust real-time object detection," in *International Journal of Computer Vision*, 2001.

[19] R. Y. TSai, "A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf tv cameras and lenses," in *IEEE Journal of Robotics and Automation*, 1987.

[20] Z. Zhang, T. Tan, K. Huang, and Y. Wang, "Practical camera calibration from moving objects for traffic scene surveillance," *IEEE TCSVT*, 2013.

[21] M. Hirzer, P. M. Roth, M. Koestinger, and H. Bischof, "Relaxed pairwise learned metric for person re-identification," in *ECCV*, 2012.

[22] S. Xiang, F. Nie, and C. Zhang, "Learning a Mahalanobis distance metric for data clustering and classification," in *PR*, 2008.

[23] "iPerf - network bandwidth measurement tool." http://iperf.fr/.

[24] C. P. Gomes and R. Williams, "Approximation algorithms," in *Search Methodologies*, ch. 18, Springer, 2005.

[25] P. Boutin, "New apps to post videos with ease," 2011.

[26] "CMU Pixy camera." http://bit.ly/1UIln1O.

[27] P. Dollar, S. Belongie, and P. Perona, "The fastest pedestrian detector in the west," in *BMVC*, 2010.