

Automatic Video Annotation by Motion Recognition

Himanshu Sinha, Shubham Gupta, Singh Anubhav Gajendra, Subhaska

Under Graduate Student, Department of Computer Science and Engineering, SRM IST, Chennai, India

Under Graduate Student, Department of Computer Science and Engineering, SRM IST, Chennai, India

Under Graduate Student, Department of Computer Science and Engineering, SRM IST, Chennai, India

Under Graduate Student, Department of Computer Science and Engineering, SRM IST, Chennai, India

ABSTRACT

Video annotation plays an important role in content based video retrieval. In this paper, we propose an automatic method to find out person identity in live video from a fixed camera by making use of a novel contextual information, motion pattern. When subjects move around in the Field Of View (FOV) of a camera, motion measurements of human body are simultaneously captured by two different sensing techniques, including camera and smart phones equipped with inertial sensors. Then classification models are trained to recognize motion pattern from raw motion data. To identify the subject that appeared in video from the camera, a metric of distance is defined to quantitatively measure the similarity between motion sequence recognized from video and each of those from smart phones. When a most similar sequence is detected, identity information related to the corresponding phone is used to annotate video frames, together with time and camera location. To test the feasibility and performance of the proposed method, extensive experiments are conducted, which achieved impressive results.

Keyword :- *person identification, motion recognition, sensor fusion.*

1. Introduction

With the rapid development of digital technologies, the volume of digital visual information produced by scientific, educational, medical, industrial and other applications available to users increased dramatically. Nowadays, visual data are as common as textual data. While textual documents in digital form are somehow self-describing (i.e., they provide explicit indices, such as words and sentences that can be directly used to categorize and access them), digital visual data do not provide such an explicit content description. In order to retrieve and manipulate visual media with effectiveness and efficiency, it is of necessary significance to annotate the visual content, providing an explicit content description targeted to the user needs. Image annotation is an active field of research that serves as a precursor to video annotation. A large amount of research has been carried out on image annotation in recent years. In general, these approaches can be divided into three different types. The first approach is the traditional text based annotation. In this approach, images are tagged manually by humans and images are then retrieved in the same way as text documents. Although effective, manual annotation is impractical to label a large amount of images. It is labor-intensive, time-consuming as well as prohibitively expensive, and often subjective and ambiguous. The second type of approach focuses on content based image retrieval (CBIR), where images are automatically indexed and retrieved with low level content features like color, shape, texture and spatial layout. However, recent research has shown that there is a significant gap between the low level content features and semantic concepts used by humans to interpret images. The problem with this approach is its reliance on visual similarity in measuring semantic similarity.

The features of our web app are:-

- The first approach is the traditional text based annotation. In this approach, images are tagged manually by humans and images are then retrieved in the same way as text documents. Our website provides a detailed information about the book so that the buyer can analyze all part of the book and then contact the seller.
- The second type of approach focuses on content based image retrieval (CBIR), where images are automatically indexed and retrieved with low level content features like color, shape, texture and spatial layout.

- Text-based search engine can successfully retrieve documents without understanding the content, there is usually no easy way for general users to give an query image or a low-level description of what he is looking for when using a CBIR system. The third approach of image retrieval is the automatic image annotation (AIA) so that images can be retrieved in the same way as text documents.
- We propose an automatic annotation method to annotate live video content by making use of human motion pattern. Motion measurements of human body collected from camera and inertial sensors are mapped into a common time motion space.

Eg:In this work, we propose an automatic annotation method to annotate live video content by making use of human motion pattern. Motion measurements of human body collected from camera and inertial sensors are mapped into a common time motion space. By analysing the consistency of motion pattern of the same subject, the semantic gap is tactically bridged and task of person identification in video is successfully accomplished. Main contributions of the proposed method are as follows.

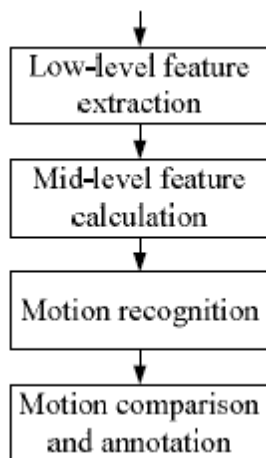


Fig. 1. Flow chart the proposed method.

2. General Framework

A flowchart of the proposed method is depicted in Fig. 1. As can be seen, low-level motion features are first extracted from camera and sensor platforms respectively. In this work, acceleration is used as the low level feature to characterize human motion, and smart phones equipped with 3-axis accelerometer are utilized as wearable sensing platforms to collect acceleration of human body. Then a set of statistical features in time and frequency domain are calculated based on the raw acceleration data in the process of mid-level feature calculation. Motion classification models are trained based on the mid-level features. When a sequence of motion is recognized from video, it is compared with each of those that were obtained from nearby smart phones. When a most similar motion sequence is found, identity information related to the corresponding smart phone is used to annotate video frames. Considering the sensing ability of camera and accelerometer on smart phones, four basic and common motion types are selected and employed, including *standing*, *walking*, *running* and *jumping*. Details of the proposed method are described in the following subsections.

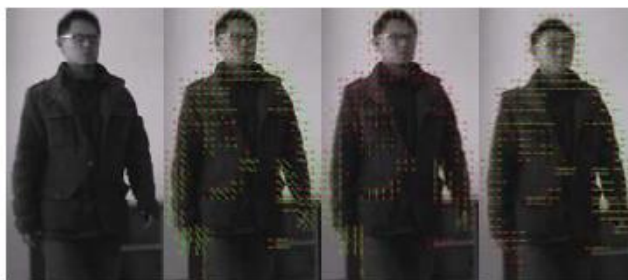


Fig. 2. Extracting optical flow from video. From left to right: original frame, flow field F , vertical flow field F_y , horizontal flow field F_x .



Fig. 3. Placement of smart phones on human body. From left to right: waist belt, jacket side pocket, chest pocket and trousers back pocket.

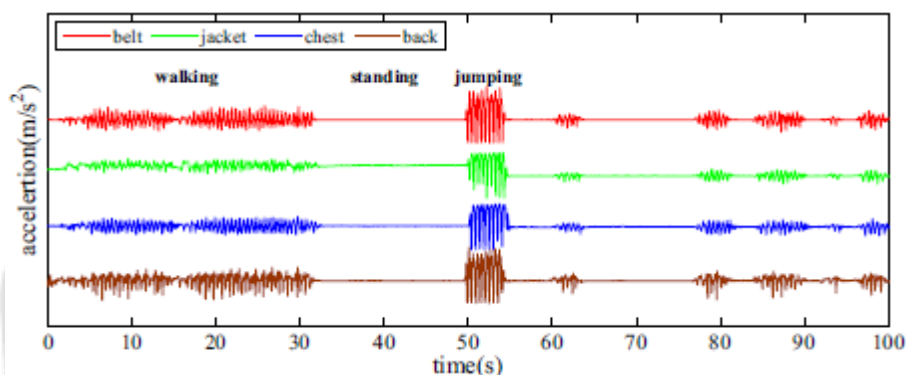


Fig. 4. Mean acceleration value corresponding to different placement of smartphones. The motion type is labelled manually.

2.1 Data Collection

To estimate acceleration of human body from video frames, first we remove the background using adaptive Gaussian Mixture Model. Then Lucas-Kanade algorithm [15] is adopted to calculate optical flow of pixels within foreground blobs corresponding to possible human bodies. Each flow field F is split into horizontal and vertical channels, F_x and F_y , which are illustrated in Fig. 2, and each channel is smoothed in temporal and spatial dimension for noise suppression and data cleansing. Pixel acceleration is defined as the second order derivative of flow vector with respect to time, and blob acceleration is approximated as the average of acceleration of all pixels within the blob. Thus, motion feature of each blob is characterized by two acceleration components.

We choose Android phone as the sensing platform to collect acceleration of human body and a preliminary experiment is conducted to assess the feasibility of different placements of smart phones on human body for acceleration collection. A participant was asked to carry four Android smart phones, three of which were placed in chest pocket, jacket side pocket, trousers back pocket respectively, and one was attached to waist belt, which are shown in Fig. 3. Then a sequence of motions defined previously were performed randomly by the participant and the average acceleration of all three axes corresponding to the four ways of placements were recorded. Results illustrated in Fig. 4 qualitatively show that all the four placements can capture motion features of human body with minor acceptable discrepancy. This test makes the choice of smart phone placement more flexible.

Standard machine learning techniques cannot be directly applied to raw acceleration data. Instead, we adopted a set of statistical features, including *mean*, *standard deviation*, *energy*, *correlation coefficients* [16], [17], which are described at length in Table I. These features are computed in a sliding window of length t_w with $t_w/2$ overlapping. Accurate assessment of motion duration requires analysis in windows of small t_w , but windows of larger t_w might carry more meaningful information of motion type and improve the recognition performance. We set t_w to 6.4 seconds which achieved best results as observed in this work.

Feature	Description
mean	Average sample value. It is calculated over each axis for accelerometer and over horizontal and vertical directions for camera.
standard deviation	Standard deviation of the samples computed in similar way as feature <i>Mean</i> .
energy	This feature is calculated as the sum of the squared discrete FFT component magnitudes of the the samples and divided by sample count for normalization.
correlation coefficients	This feature is computed as the correlation between samples of each axis for accelerometer, and between samples in horizontal and vertical directions for camera.

Table 1

Description of the features extracted from acceleration data

To prepare data for model training, we first construct a visual dataset by taking advantage of related motion types in Weizmann [18], [19] and KTH [20] datasets, as is depicted in Table II. Representative frames of our dataset are shown in Fig. 5. These video clips are first cropped and aligned. Then, five subjects were recruited to collect accelerometer measurements. Each of them carried an Android smart phone in chest pocket and performed the aforementioned motions each for a specific period of time. The final dataset was about 5 hours long. However, as the motion of *standing* is not considered in both Weizmann and KTH datasets, to complement our visual dataset, the five subjects were asked to stand in the FOV of a camera when we collect acceleration data from smart phones for *standing*. The resulted video clips were included into our visual datasets, as is shown in Fig. 5f. The data collection was controlled by an Android application we created. This application provides a simple graphic interface for user to enter their names, start and stop data collection, and manually label the motion being performed. The data were collected at about 20 samples per second and saved on the phone and processed later via USB.

Motion type	Data Source	Data Specification
walking	<i>walk</i> from <i>Weizmann</i>	9 subjects perform <i>walking</i> in same environment. Video resolution is 180x144. Frame rate is 25 fps.
	<i>walking</i> from <i>KTH</i>	25 subjects perform <i>walking</i> several times in 4 different scenarios. Video resolution is 160x120. Frame rate is 25 fps.
running	<i>run</i> from <i>Weizmann</i>	Similar to <i>walk</i> of <i>Weizmann</i> .
	<i>running</i> from <i>KTH</i>	Similar to <i>walking</i> of <i>KTH</i> .
jumping	<i>jump</i> from <i>Weizmann</i>	Similar to <i>walk</i> of <i>Weizmann</i> .

Table 2:

Description of the motion dataset

2.2 Motion Recognition

Once the dataset was prepared, we make use of WEKA machine learning toolkit to recognize motion from the extracted features. Six different classifiers were evaluated with ten-fold cross validation (10-fold CV), including Decision Tree(J48), Decision Table, Multilayer Perceptron, Naive Bayes, Bayesian Network and Logistic Regression. The classification results (mean \pm standard deviation) of 10-fold CV for motion recognition from video and smart phone are presented in Table III. The aggregate recognition rates of all the motion is listed in Table IV. Based on this evaluation, we can quantitatively conclude that the motions defined in section III is clearly discernable with each other and could be recognized from acceleration features of both video and smart phones accurately.

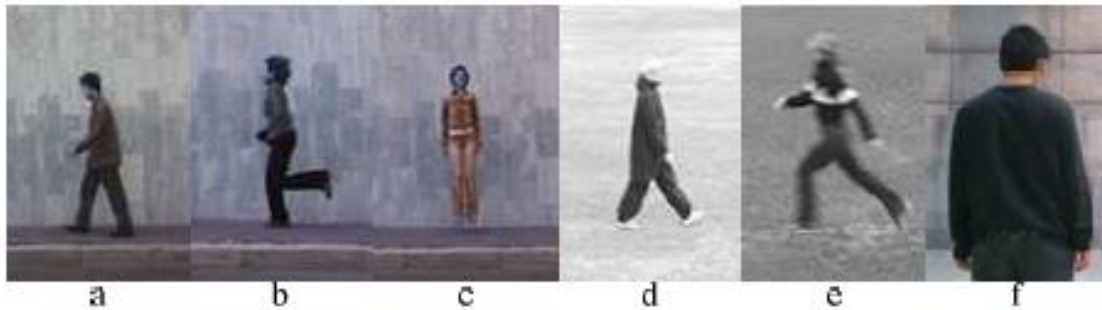


Fig. 5. Sample frames of our visual dataset. ‘a’, ‘b’ and ‘c’ are walk, run and jump from Weizmann. d and e are walking and jumping from KTH. ‘f’ is motion of standing in this work.

2.3 Motion Comparison and Annotation

When a subject carrying a smart phone move around in the FOV of a camera, we try to find out his identity by comparing the motion sequence recognized from video frames recorded by the camera and each of those obtained from smart phones carried by all nearby subjects. When a most similar motion sequence from smart phone is detected, the identity information related to that phone is used to annotate the subject in the camera FOV. To this end, a distance metric is needed to measure the similarity between two motion sequences. Assume that a motion sequence $f_i(t) = m$, $i \in \mathbb{N}$, $t_s \leq t \leq t_e$, where m is assigned to values from 1 to 4 which corresponds to *standing*, *walking*, *running*, *jumping* relatively, and t_s and t_e are the start and end time of the motion sequence. This assignment is devised in order to reflect the intrinsic difference between these motions. For example, the intensity difference between *jumping* and *standing* is more distinctive than that between *jumping* and *running*. Thus, *running* is arranged more closer to *jumping* than *standing*. The distance metric is defined in (1).

$$d(f_1(t), f_2(t)) = \left(\int_{t_s}^{t_e} |f_1(t) - f_2(t)|^2 dt \right)^{\frac{1}{2}}, \quad (1)$$

The theoretical minimum value of $d(f_1(t), f_2(t))$ is 0. This happens when $f_1(t)$ and $f_2(t)$ are exactly the same. The larger the value, the more dissimilar $f_1(t)$ and $f_2(t)$ are. When identity of the subject in video is found, video frames from t_s to t_e can be labeled with a tag in the format of $\{id, \{ \langle m_1, t_1 \rangle, \langle m_2, t_2 \rangle, \dots, \langle m_n, t_n \rangle \}, l_c\}$, where id is the subject identity, m_i is motion type which has a duration of t_i in seconds, and l_c is the location of the camera FOV.

3. System Architecture

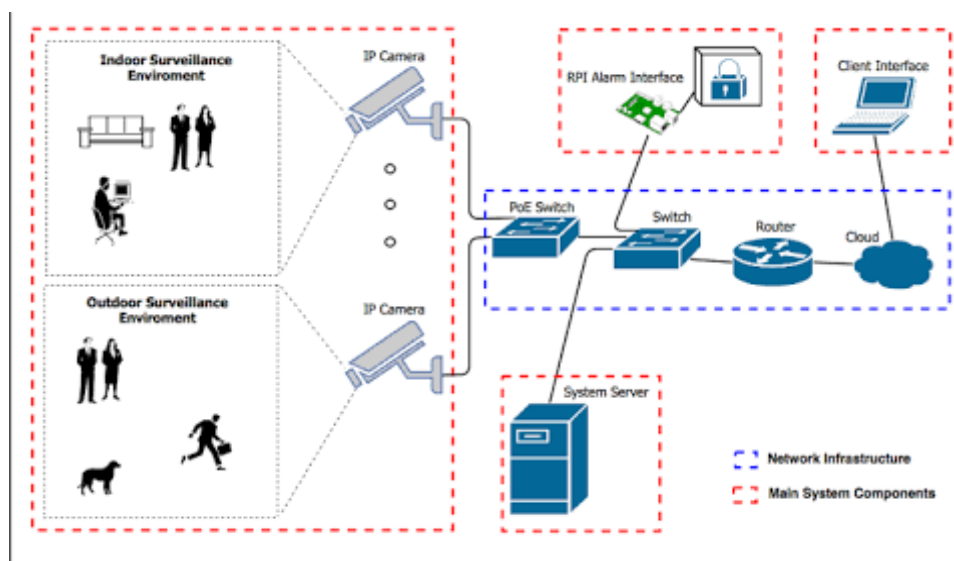


Fig -6: System Architecture

3.1 Hardware Requirements

Raspberry Pi 3 (Any other Version will be nice)

Camera with wi- fi enabled feature

Breadboard and jumper cables

5v Relays

220 or 100 ohms resistor

Diodes

Jumper Wires

Connection Blocks

LEDs to test.

AC lamp to Test

Memory card 8 or 16GB running Raspbian Jessie

2n222 transistors

3.2 Proposed System

This surveillance system detects motion of any object within its camera's range, captures the image and stores it in Raspberry Pi. You can retrieve images from Raspberry Pi any time. Since Raspberry Pi camera is static-sensitive, take good grounding measures before you touch it. Connect the camera as shown in Fig. 1.



Fig. 7: Raspberry Pi and the camera

3.3 Installation of camera

Open Raspiconfig utility using the command:

```
[stextbox id="grey"]$ sudo raspi-config[/stextbox]
```

Now, enable the camera as shown in Fig. 8.

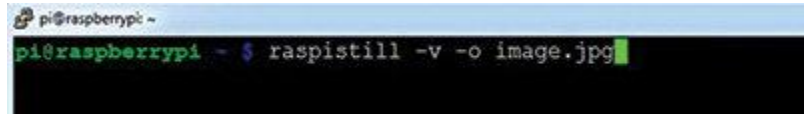


Fig. 8: Enabling the camera

If your firmware is old, you need to update and upgrade it using the command mentioned below to see the camera enable option:

```
[stextbox id="grey"]$ sudo apt-get update
$ sudo apt-get upgrade[/stextbox]
```

After enabling and connecting the camera to Raspberry Pi, use following commands to check if it is working well. The first command is raspistill, which is used to take still pictures. For example, if you want to capture an image, type the following command (also see Fig. 9):



```
pi@raspberrypi ~$ raspistill -v -o image.jpg
```

Fig. 9: Capturing an image

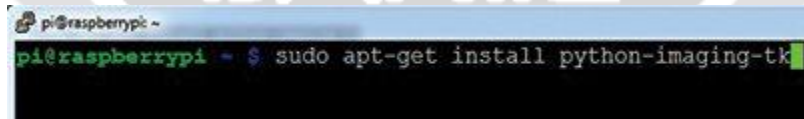
```
[stextbox id="grey"]$ raspistill -v -o image.jpg[/stextbox]
```

If red LED of the Raspberry Pi camera glows for some time, it shows that the image has been recorded. You may check the image under Home folder in Raspberry Pi to ensure that the camera is properly installed.

3.4 Capturing Motion

We have two methods for motion detection. First method uses the light-weight motion detection Python script while the second method uses 'motion' software to detect motion and capture images.

Method 1: Using Python script. Python script depends on Python Imaging Library, which is meant for analysing and manipulating images. So we have to install the library using the following command (also see Fig. 10):



```
pi@raspberrypi ~$ sudo apt-get install python-imaging-tk
```

Fig. 10: Installing Python script

```
[stextbox id="grey"]$ sudo apt-get install python-
imaging-tk[/stextbox]
```

We can get the script from the link mentioned below:

```
[stextbox id="info"]https://github.com/skl/raspberry-pi-cam/archive/master.zip[/stextbox]
```


Make the script executable by using the command (also see Fig. 11):



Fig. 11: Making the script executable

```
[textbox id="grey"]$ chmod +x picam.py[/textbox]
```

The script is designed to store images in a directory named picam under Home directory, so create it as shown below before executing the script:

```
[textbox id="grey"]$ mkdir ~/picam[/textbox]
```

We shall be ready to run the script after giving the following command:

```
[textbox id="grey"]$ ./picam.py[/textbox]
```

The script will turn on red LED of the Raspberry Pi camera and start taking low-resolution images. It will then compare them and look for movement by comparing the pixels in the images. If it detects any changes in the pixels, the script will capture a higher-resolution image.

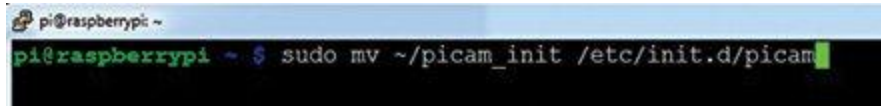
The script is written in such a way that it stores only high-resolution images. All these images are saved in the ~/picam folder which we had created. Note that, if the camera is placed in a windy area then set the threshold variable in config file to a higher value than the default.

For running the script during booting, we need an init script that runs the picam.py script and kills it before shutting down the Raspberry Pi. To get the script, issue the command given below:

```
[textbox id="grey"]$ wget http://pastebin.com/AfqbjQrb
```

```
-O picam_init[/textbox]
```

Move the script into its correct location using the command (also see Fig. 12):

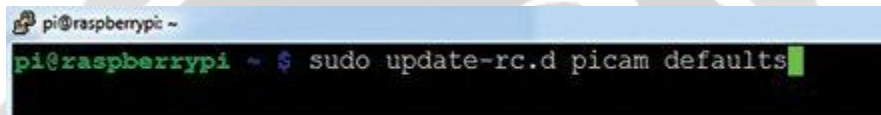


```
pi@raspberrypi ~
pi@raspberrypi ~ $ sudo mv ~/picam_init /etc/init.d/picam
```

Fig. 12: Moving the script to desired location

```
[stextbox id="grey"]$ sudo mv ~/picam_init /etc/init.d/
picam[/stextbox]
```

Next make the script executable using the command (also see Fig. 13):

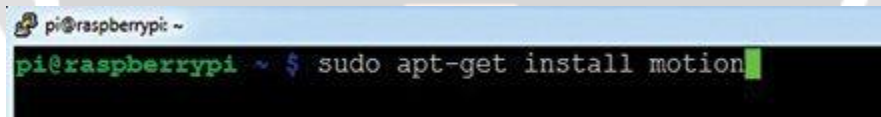


```
pi@raspberrypi ~
pi@raspberrypi ~ $ sudo update-rc.d picam defaults
```

Fig. 13: Making the script executable

```
[stextbox id="grey"]$ sudo chmod +x /etc/init.d/picam[/stextbox]
```

To make the script start during booting, type the command given below (also see Fig. 14):



```
pi@raspberrypi ~
pi@raspberrypi ~ $ sudo apt-get install motion
```

Fig. 14: Making the script start at boot time

```
[stextbox id="grey"]$ sudo update-rc.d picam default[/stextbox]
```

The script will now start and shut down along with the Raspberry Pi. We can also control it manually. That is, to stop the script, use the command:

```
[stextbox id="grey"]/etc/init.d/picamstop[/stextbox]
```

To start the script, use the command:

```
[stextbox id="grey"]/etc/init.d/picamstart[/stextbox]
```

Method 2: Using motion software. To install motion software, use the command below (also see Fig. 15):



```
pi@raspberrypi ~ $ sudo apt-get install -y libjpeg62 libjpeg62-dev libavformat53
libavformat-dev libavcodec53 libavcodec-dev libavutil51 libavutil-dev libc6-dev
zlib1g-dev libmysqlclient18 libmysqlclient-dev libpq5 libpq-dev
```

Fig. 15: Installing motion software

```
[stextbox id="grey"]$ sudo apt-get install
motion[/stextbox]
```

The standard motion packages do not yet work with Raspberry Pi camera. For that we have to install the special binary code known as motion-mmml. So install following dependencies (also see Fig. 16):



```
pi@raspberrypi ~ $ sudo apt-get install -y libjpeg62 libjpeg62-dev libavformat53
libavformat-dev libavcodec53 libavcodec-dev libavutil51 libavutil-dev libc6-dev
zlib1g-dev libmysqlclient18 libmysqlclient-dev libpq5 libpq-dev
```

Fig. 16: Installing all the dependencies

```
[stextbox id="grey"]$ sudo apt-get install -y libjpeg62
```

```
libjpeg62-dev libavformat53 libav
```

```
format-dev libavcodec53 libavcodec-
```

```
dev libavutil51 libavutil-dev libc6-
```

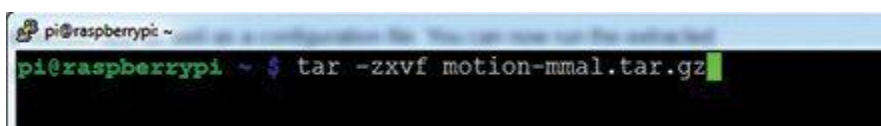
```
dev zlib1g-dev libmysqlclient18
```

```
libmysqlclient-dev libpq5 libpq-dev[/stextbox]
```

After installing all the dependencies, download motion software from the link:

```
[stextbox id="info"]https://www.dropbox.com/s/xdfcxm5hu71s97d/motion-mmml.tar.gz[/stextbox]
```

Extract it using the command (also see Fig. 17):



```
pi@raspberrypi ~ $ tar -zxvf motion-mmml.tar.gz
```

Fig. 17: Extracting the folder

```
[stextbox id="grey"]tar -zxvf motion-mmal.tar.gz[/stextbox]
```

This will extract the motion application as well as a configuration file. Run the extracted motion application along with its configuration file using the command (also see Fig. 18):



```
pi@raspberrypi ~ $ ./motion -n -c motion-mmalcam.conf
```

Fig. 18: Running the motion application

```
[stextbox id="grey"]./motion -n -c motion-mmalcam.conf[/stextbox]
```

While motion application is running, open the browser and type the IP address of your Raspberry Pi with the 8081 port (for example, 192.128.2.79:8081), as shown in Fig. 13, to see the live streaming video.

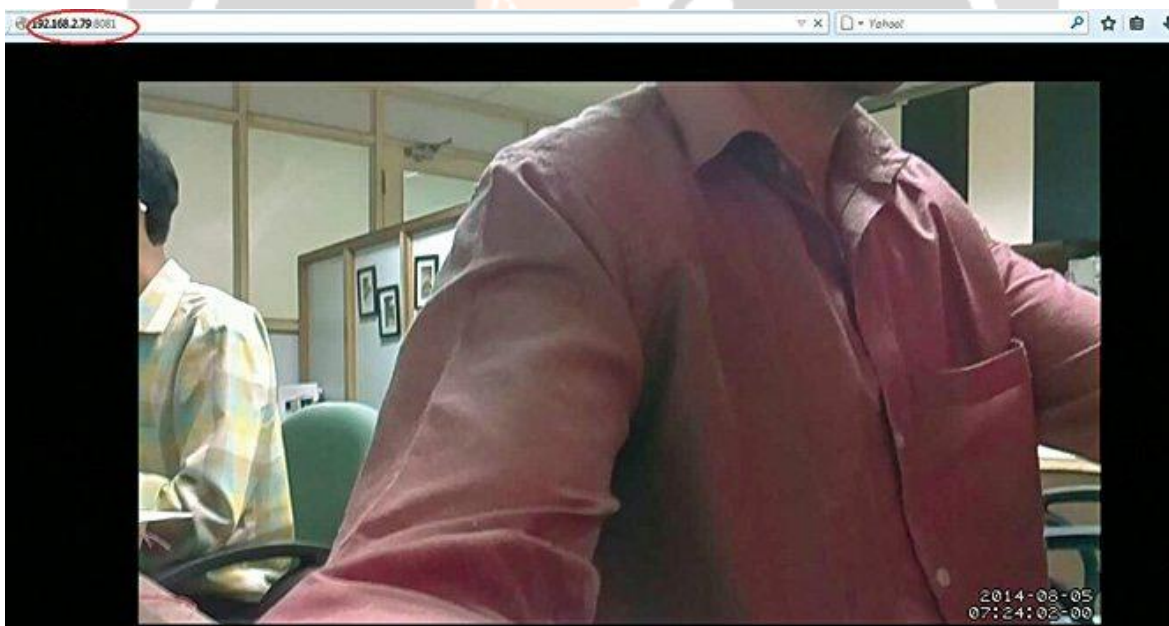


Fig. 19: Streaming video

When camera detects the motion it will also capture the image and store it in Raspberry Pi Home folder, as shown in Fig. 20.

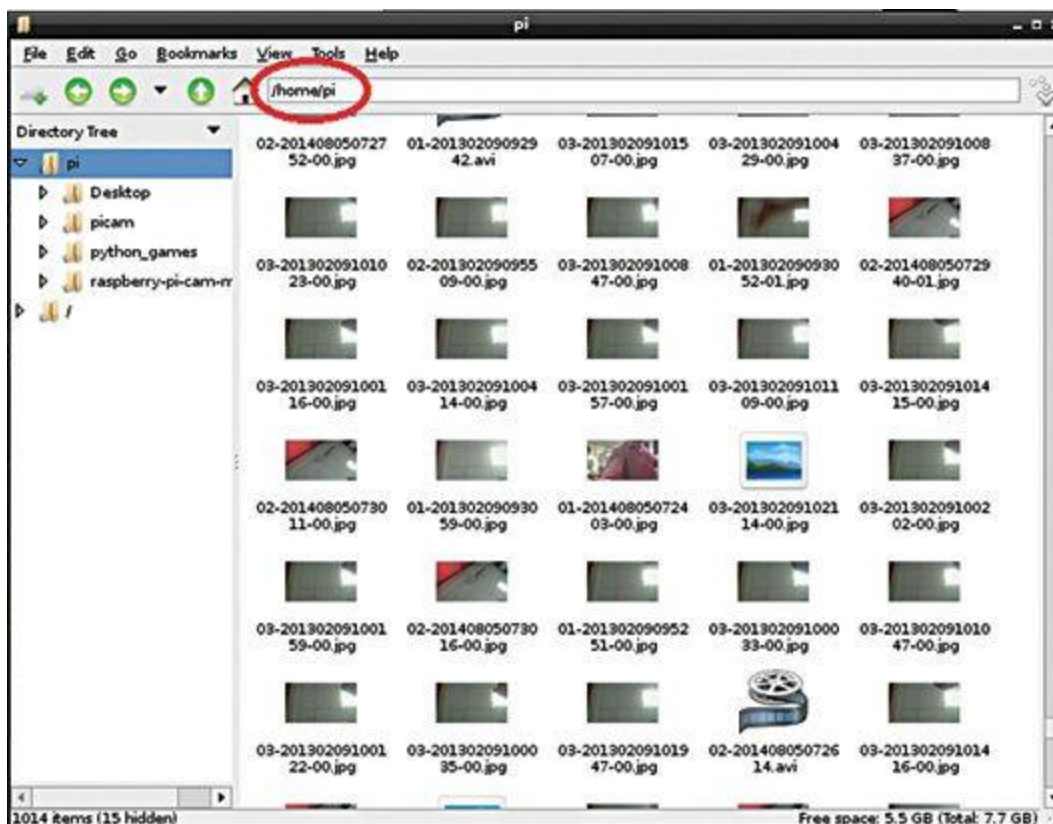


Fig. 20: Path of the folder

4. CONCLUSIONS

In this paper, we propose a novel method to annotate video content by fusing information from different sensors, that is, analyzing motion pattern sensed by camera and smart phones. The task of person identification is effectively accomplished by comparison of motion sequences with motion type as a side benefit for video annotation. However, the method has its limits. First, the adoption of smart phones may break the unobtrusive feature of camera sensing. Subjects needs to carry smart phones in order to be identified. Second, currently only one subject was allowed in the camera FOV. In the future, we plan to figure out robust human detection and tracking techniques to simultaneously identify multiple subjects in camera video.

5. REFERENCES

- [1] M. M. Islam, D. Zhang, and G. Lu, "Automatic categorization of image regions using dominant color based vector quantization," in *Computing:Techniques and Applications, 2008. DICTA'08. Digital Image.* IEEE,2008, pp. 191–198.
- [2] N.-C. Yang, W.-H. Chang, C.-M. Kuo, and T.-H. Li, "A fast mpeg-7 dominant color extraction with new similarity measure for image retrieval," *Journal of Visual Communication and Image Representation*, vol. 19, no. 2, pp. 92–105, 2008.
- [3] Y. Liu, D. Zhang, and G. Lu, "Region-based image retrieval with high-level semantics using decision tree learning," *Pattern Recognition*, vol. 41, no. 8, pp. 2554–2570, 2008.
- [4] Y. Lu, L. Zhang, Q. Tian, and W.-Y. Ma, "What are the high-level concepts with small semantic gaps?" in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on.* IEEE, 2008, pp.1–8.

- [5] J. Li and J. Z. Wang, "Real-time computerized annotation of pictures,"

Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 30, no. 6, pp. 985–1002, 2008.

- [6] S. M. R. V. Liyan Zhang, Dmitri V. Kalashnikov, "Context-based person identification framework for smart video surveillance," *Machine Vision and Applications*, 2013.
- [7] X. Geng, K. Smith-Miles, L. Wang, M. Li, and Q. Wu, "Context-aware fusion: A case study on fusion of gait and face for human identification in video," *Pattern recognition*, vol. 43, no. 10, pp. 3660–3673, 2010.
- [8] N. O'Hare and A. F. Smeaton, "Context-aware person identification in personal photo collections," *Multimedia, IEEE Transactions on*, vol. 11, no. 2, pp. 220–228, 2009.
- [9] Z. Stone, T. Zickler, and T. Darrell, "Autotaggingfacebook: Social network context improves photo annotation," in *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW'08. IEEE Computer Society Conference on*. IEEE, 2008, pp. 1–8.

