People Re-Identification from RGB-D Videos

Gabriele Gaudenzi, Fabio Pollastri and Alessio Signoracci - VRAI Group 4

***Abstract*— This paper presents the authors' workflow and results regarding the IM project proposed by *VRAI* (`http://vrai.dii.univpm.it/`) at *Università Politecnica delle Marche*, consisting in creating a video dataset of people from a top-view RGB-D sensor and developing an application able to extract features in order to classify and re-identify subjects.**

I. INTRODUCTION

Identification cameras are increasingly deployed in public places like airports, office buldings, malls or museums for surveillance purposes. Manual human monitoring of these videos is expensive, error prone and time consuming, and given the huge volume of data that a broad-coverage network of cameras can produce, it is desirable to deploy automated analysis techniques and tools to improve speed, quality and cost of surveillance[1].

In this context, *re-identification* (re-id) is defined as a process that establishes a correspondence between images of a person recorded by different cameras or in different moments, to determine whether different instances actually belong to the same subject. This process implies using a number of unique descriptors for every subject to be able to recognize it after being previously identified.

While this is a task humans do naturally, all the time and without much effort, fully automated machine-performed re-id is a non-trivial problem: even small variations in a person appearance, change of perspective, occlusion or lighting can negatively affect the accuracy and effectiveness of image/video analysis.

A typical re-id system performs two basic operations: capturing a unique person descriptor or model and then comparing two models to infer either a match or a non-match. Fig. 1 shows a schematic representation of a re-id process, detailing subtasks within each component described. The algorithm developed by the authors and presented in this paper has a similar structure as well, whose exact steps will be detailed in subsequent sections.

Project Specifications

The goal of this project was to develop an application able to identify and recognize subjects, even when they reappear on camera on a later time, using a RGB-D sensor. The functional requirements were:

1) dataset creation:
   - acquisition from RGB-D sensor;
   - manual video tagging and annotation;
2) use of a tracking algorithm:
   - definition of a relevant features set;
   - extract and store features from videos;
3) development of a re-identification algorithm that matches a subject against the features dataset.

The reminder of this paper also follows this structure, and will cover the above items in order: Section II details the creation of the video dataset used to test our re-id algorithm; Section III explains the technical details of the features extraction step; Section IV discusses the classification process, commenting on tests and benchmark results; Section V concludes the paper.

II. DATASET CREATION

While there is a number of publicly available datasets that are commonly used to test re-id models (ViPER[2], ETHZ[3] and i-LIDS for Re-ID[4] to name a few), we chose to create one ourselves to be able to use biometric-based features together with the classical appearance-based ones (i.e. colors): the former need to be extracted from the depth data acquired with a RGB-D sensor that none of the aforementioned datasets provide.

In the first phase of our project all groups cooperated to acquire RGB-D videos of 67 people, that were taken using a *Asus Xtion PRO Live* (technical specification are shown in Table I), situated on a ceiling approximately 3.7m tall. The *OpenNI2*[5] framework was used to acquire the depth data as a fourth video channel (in addition to RGB), containing the point cloud of every frame of the scene. The resulting `.oni` video files, with a 640x480 resolution and 30fps framerate, are approximately 100MB in size per second.

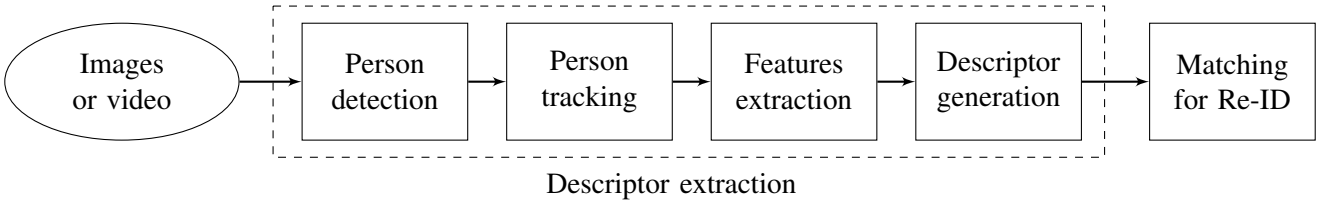

Fig. 1. Re-identification system diagram

This dataset is composed by two different groups of data[2][7]:

- the first ("single") consists of 67 top-view videos showing a single person, walking under the camera following the path illustrated in Fig. 2: this is done to show each subject from every side. These video are used as a "training set" for our application, namely to extract people's features and to generate a knowledge base of features;
- the second ("group") consists of 13 top-view videos showing multiple subjects passing one after another under the camera; each one appears two times going in opposite directions and in a different order. These are used as our application's "test set", to identify and match instances of people with their extracted features.

Each video was subsequently manually tagged, labeling each training video with a numeric ID corresponding to the subject real name, and each test video with an ID sequence corresponding to the order in which people appeared; the latter acted as a "ground truth" to rate the classifier's predicted class against.

Power consumption	below 2.5W
Field of view	58 H, 45 V, 70 D
Sensors	RGB, Depth, 2x Mic
Depth/Color image size	(640x480): 30fps
Interface	USB 2.0/3.0
Technology	Structured light[6]
Operating environment	indoor
Dimensions	18 x 3.5 x 5cm

TABLE I
ASUS XTION PRO LIVE DATASHEET

III. FEATURES EXTRACTION

We developed a script, executable from a Windows, GNU/Linux, *BSD and Mac OS terminal, that can both extract subject features and classify people according to its training data (the features set previously saved). Table II details the programming language and libraries used (the listed version numbers correspond to the latest available packages in *Arch Linux* and *AUR* repositories).

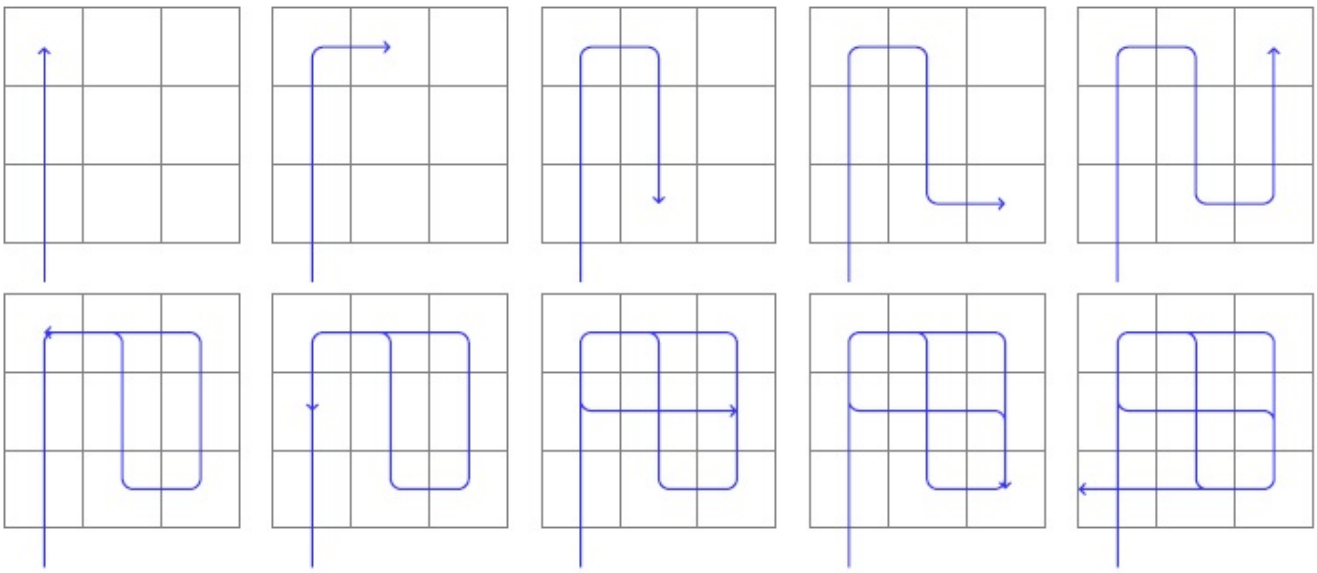

Fig. 2. Path followed by subjects in "single" videos

Python	2.7.10
OpenNI2	2.2beta2
OpenCV	2.4.11
pcl	1.7.2

TABLE II

LIBRARIES USED BY THE APPLICATION

Launching the script with

```
$ python2 features.py -f <path>
```

triggers the *features extraction* mode: it will open the `path` video file (must be a `.oni` video), analyzing both RGB and D channels frame by frame. The features we chose to use for person descriptors and are extracted from videos are:

- *height* (mm)
- *head circumference* (px)
- *shoulders circumference* (px)
- *hair color*
- *shirt color*

These features are sampled only in a limited set of frames, in particular the ones in which the person is at the center of the image. We made this choice because while writing code we noticed that depth frames values were skewed due to the camera not being perfectly perpendicular to the floor: the features are extracted only when the subject is within the $11\,\mathrm{x}\,11$ pixel box centered at $(320,\ 240)$, in the middle of the video.

Person detection was quite easy to do with the depth cannel info; the only gotcha was to detect and filter pixels with invalid values. After cleaning the frame we segmented the image to create a mask of pixel with values between a and b in Table III, corresponding to white pixels in Fig. 3. Setting the a value so high (we lose the subject's shoes) was necessary to filter out the noise caused by the floor.

The c threshold in Table III was defined as the minimum height to detect a person in the frame; if the pixel box condition described above is also satisfied the

a) **Subject mask (lower bound)**	150 (mm)
b) **Subject mask (upper bound)**	2500 (mm)
c) **Subject min height**	1500 (mm)
d) **Min. shoulders contour area**	5000 (px)
e) **Min. head contour area**	2500 (px)
f) **Head mask threshold**	150 (mm)
g) **Shoulders mask threshold**	500 (mm)

TABLE III

(ARBITRARY) THRESHOLDS USED IN OUR APPLICATION

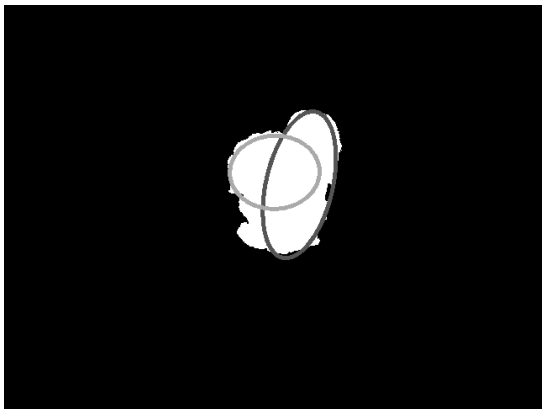

Fig. 3. Subject mask, with head and shoulder area highlighted

features collection begins:

- subject height is calculated as the maximum depth pixels value falling inside the Fig. 3 mask;
- the depth frame is segmented with a new mask that selects pixels with a value between the extracted height h and $h\ -\ f$ (in Table III), then the head circumference is calculated with:

$$C_{eq} = \sqrt{4\,\pi A_{eq}} \tag{1}$$

  where A_{eq} is the area of the mask contour, returned by *OpenCV* function `cv2.contourArea()`. Shoulders circumference is extracted in the same way, with a mask selecting pixels with values between $h\ -\ f$ and $h\ -\ g$ (in Table III) and using (1). In order to get believable values we impose that returned A_{eq} should be higher than e and d (in Table III), for the head and shoulders respectively. Fig. 3 highlights the fitted ellipses of head and shoulders contours, in light gray and dark gray respectively.
- hair color and shirt color are sampled from the RGB frame with the head and shoulders masks generated on the previous step. Unfortunately, during development we noticed that the color and depth streams are not synced, meaning that applying masks calculated from a depth frame to the corresponding color frame left out a portion of the subject while including a portion of the floor. To get around this, we performed the morphological operation of *erosion* on masks images with a $10\,\mathrm{x}\,10$ kernel before application. We sample the *dominant* color[2] as a feature, i.e. the color that appears the most frequently, expressed in the *HSV* colorspace as we found it to be more expressive than *RGB*, in accord with the general results shown

in [9]. Image histogram is calculated with 18 bins for *H* channel and just 4 bins for *S* and *V*, effectively ignoring small color variations.

We take as final features the average reported height and head circumference, the maximum shoulders circumference (because sometimes they are not detected, or only one shoulder is selected), and the most common reported H, S, V values. As a final step, these are normalized according to values in Table IV, in order to have the same weight for all the features during the classification process. These values are written in a `.csv` file, following the subject ID, where every row is a descriptor.

Height	2500
Head circ.	500
Shoulders circ.	500
H value	17
S value	3
V value	3

TABLE IV
NORMALIZATION VALUES USED

IV. CLASSIFICATION

Launching the aforementioned script with

```
$ python2 features.py -k <path>
```

triggers the *classification* mode: it will open the `path` video file (must be a `.oni` video), extracting subjects features as explained in Section III and matching them with one of the descriptors in `features_id.csv`, assigning them an ID. The chosen classifier is *K-Nearest Neighbor*[8] as implemented in *OpenCV* (`cv2.KNearest()`), with $k = 1$ so the subject is given the class of their nearest neighbor. We chose K-NN because is one of the simplest and fastest classifying methods, while giving accurate enough results. The classifier is trained using the already mentioned `features_id.csv` as "training set" (`KNearest.train()`), and assigns to new subjects the same class of their *nearest neighbor*, where "nearest" means the one with the minimum Euclidean distance, interpreting features descriptors as $n-$dimensional vectors in $\mathbb{R}^n$.

Test results

We did a little test using a subset of our video dataset, extracting features from 10 "single" videos (to use as training set), and successively trying to classify the subjects in 2 of the 13 "group" videos (our test set); the subjects in the latter 2 videos are all in the chosen "single" videos. In order to benchmark the algorithm performance we reported the results of the classification on a confusion matrix. Each column of the matrix represents the instances in a predicted class while each row represents the instances in an actual class. Through the confusion matrix, we evaluate the performance of the algorithm defining the following quantities for each subject:

- TP (*True Positive*): the number of times the subject has been correctly identified;
- TN (*True Negative*): the number of times other subjects have not been classified as the subject;
- FN (*False Negative*): the number of times the subject has been misclassified as one of the other subjects;
- FP (*False Positive*): the number of times other subjects have been mistakenly classified as the subject;

From the above quantities we derived the following metrics to assess the performance of the classification algorithm:

- *Detection Rate* $DR = TP \,/\, (TP + FN)$: probability that a subject is classified correctly;
- *False Negative Rate* $FNR = FN \,/\, (TP + FN)$: probability that a subject is wrongly classified as one of the other subjects;
- *False Positive Rate* $FPR = FP \,/\, (TN + FP)$: probability that other subjects are wrongly classified as the subject;
- *Overall Accuracy* $OA = (TP + TN) \,/\, (TP + FP + TN + FN)$: accuracy of the classification algorithm in classifying subjects;

As additional metrics, we calculated the training error executing the script in "classification mode" on the "single" videos and the script execution times. All the metrics presented above are reported in (2) and tables V, VI and VII.

$$\begin{bmatrix} TP & FP \\ FN & TN \end{bmatrix} = \begin{bmatrix} 13 & 7 \\ 7 & 173 \end{bmatrix} \tag{2}$$

Generally, the most important parameter to evaluate how well an algorithm performs is OA, but in this case we consider it not very significant due to high TN. For this reason and for the nature of our dataset, the DR is an alternative interesting metric: as showed in Table VI, the value of DR is 65%, which means that our algorithm has a sufficient ability to re-identify people using the 5 features selected. A low FPR means that,

	Predicted class									
	4	5	6	27	28	29	30	31	32	33
4	1	0	1	0	0	0	0	0	0	0
5	0	2	0	0	0	0	0	0	0	0
6	0	0	1	0	0	0	1	0	0	0
27	0	0	0	2	0	0	0	0	0	0
28	0	0	0	0	1	0	1	0	0	0
29	0	0	0	0	0	0	1	0	1	0
30	0	0	0	0	0	0	2	0	0	0
31	0	0	0	0	0	0	0	2	0	0
32	0	0	0	0	0	0	0	1	1	0
33	0	0	0	0	0	0	1	0	0	1

TABLE V

CONFUSION MATRIX FOR OUR TEST

DR	65%
FNR	35%
FPR	3,9%
Train. error	16,7%
OA	93%

TABLE VI

MEASURED PERFORMANCE METRICS

for each subject it is quite unlikely that other subjects are classified as they. A 35% FNR expresses that we should improve the misclassification error. Finally a training error of 16,7% suggests that a better features extraction could be implemented.

V. CONCLUSIONS

Our script shows the possibility of re-identify people by means of a fixed set of features, with sufficient performances. To achieve better results, further developments could concern:

- adding new features to the subjects descriptor, or improve current features extraction;
- a different classifying algorithm;

	"Single"	**"Group"**
Avg. time	15.49s	33.45s
Min. time	12.33s	22.57s
Max. time	19.44s	44.33s

TABLE VII

TEST EXECUTION TIMINGS

- different weights for the features, to give more importance to more relevant or efficient features;
- explore new techniques of features extraction;
- further tweaking of the arbitrary thresholds during mask creation and features detection.

REFERENCES

[1] P. Tu, G. Doretto, N. Krahnstoever, A.G.A. Perera, F. Wheeler, X. Liu, J. Rittscher, T. Sebastian, T. Yu, K. Harding, *An intelligent video framework for homeland protection*, Proceedings of SPIE Defence and Security Symposium - Unattended Ground, Sea, and Air Sensor Technologies and Applications IX, 2007.

[2] D. Gray, H. Tao, *Viewpoint invariant pedestrian recognition with an ensemble of localized features*, Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008, Part I. LNCS, vol. 5302, pp. 262–275. Springer, Heidelberg (2008).

[3] A. Ess, B. Leibe, L. Van Gool, *Depth and Appearance for Mobile Scene Analysis*, Proc. IEEE Int'l Conf. Computer Vision, 2007.

[4] W.-S. Zheng, S. Gong, T. Xiang, *Associating Groups of People*, Proc. British Machine Vision Conf., 2009.

[5] OpenNI2 Downloads and Documentation, `http://structure.io/openni`, The Structure Sensor.

[6] B. Freedman, A. Shpunt, M. Machline, Y. Ariel, *US Patent US2010/0118123*, 2010.

[7] W. Zheng, S. Gong, T. Xiang, *Person re-identification by probabilistic relative distance comparison*, Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on. IEEE, 2011, pp. 649–656.

[8] T.M. Cover, P.E. Hart, *Nearest neighbor pattern classification*, IEEE Trans. Inform. Theory, IT-13(1):21–27, 1967.

[9] C. Liu, S. Gong, C.C. Loy, S. Lin, *Person re-identification: what features are important?*, Proceedings of the 12th European conference on Computer Vision, ECCV Workshops, 2012, pp. 391–401.