Technion - Computer Science Department

Center for Intelligent Systems

-Project Book-

Person Re-Identification

Application for Android

Instructor: Yair Atzmon

Project by: Dmitry Altshuler

Table of contents

Introduction

One of the security field related problems is a person re-identification via security cameras. In general, an object re-identification and in particular a person re-identification process faces a few challenges. The camera is usually of a low resolution, which doesn't allow identifying a unique data. The illuminations of the captured pictures are not the same. It probably will be different on the indoor and the outdoor cameras. Also the pose and the geometry of the captured person are usually not the same.

In this project I have implemented an android application which uses a color based algorithm for person re-identification. This application consists of two parts. The algorithmic part includes an implementation of the re-identification algorithm with C++ using the OpenCV Library. The user interface is implemented on the android platform which uses the algorithm through the JNI layer.

Two sets of the same person pictures.
Demonstrating the challenges of the re-identification algorithm:

- Low camera resolutions.
- Different illumination.
- Different pose.
- Different geometry.

This project is based on the paper "Color Invariants for Person Re-Identification"
By Mr. Igor Kviatkovsky, Dr. Amit Adam and Prof. Ehud Rivlin from the Technion CS department.
And a prior Matlab project by Mr. Igor Kaviatkovsky.
The images of the algorithm-description section in this document are taken from the above paper.

Algorithm Description

The algorithm that was implemented in this work is a color based algorithm. As previously mentioned, this kind of algorithm faces a few challenges: illumination, pose, geometry and low resolution. Those attributes are prone to change between cameras thus comparing the colors value of images is not effective. To overcome those challenges a color invariant was invented, this invariant means that under the above circumstances the color data doesn't change much.

When the invariants are extracted from the images they can be evaluated.
The evaluation results in a distance value which indicates the similarity of the images. The lower the distance value the closer the images.
To evaluate the distance a threshold value was chosen by experimentation. When the distance is less than the threshold the images are labeled as containing the same person, otherwise the persons are different.

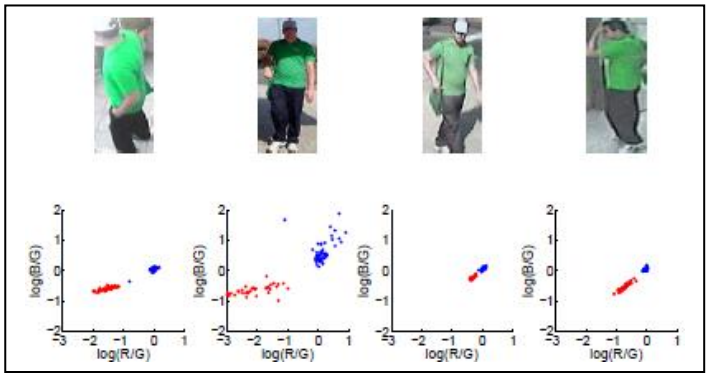
Log-Chromaticity colorspace

On the first stage of the algorithm an image and its silhouette are received from an outer source. The silhouette is a binary matrix of the same dimensions as the image. Its purpose is to filter out the irrelevant pixels of the image. Such pixels could be the background behind the person.

The image is then split into two rectangular windows. An upper window which approximately captures the 'shirt' related pixels of the person, and a lower window which captures the 'pants' pixels.
Those windows location are predefined and remains the same for every image.
At each window, which referred as ROI (area of interest), a log-chromaticity colorspace is calculated. Every pixel at the ROI is transferred into a coordinate of the form: $(\ln\frac{R}{G}, \ln\frac{B}{G})$ which results in an upper and a lower coordinates lists which referred as upper and lower clouds.
The upper cloud is marked with red color and the lower cloud is marked with blue color.

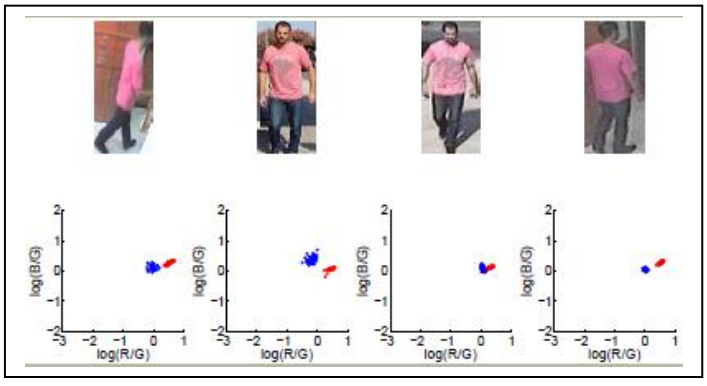
On the figure from the right a several images of the same person were taken.

Under each image the upper (red) and lower (blue) clouds of coordinates are drawn.



Notice that the locations of the clouds in all images are similar. The blue cloud is located at the upper-right side of the red cloud. This fact would be valuable on the next section.

On figure from the right is an example which demonstrate how a different person has a different cloud location at his log-polar histogram.



Parts Shape Context Representation (PartsSC)

Shape Context descriptor of an image is a 2D logpolor histogram.
It counts the number of located coordinates in a $\log(r)$ radius with θ from a reference point.

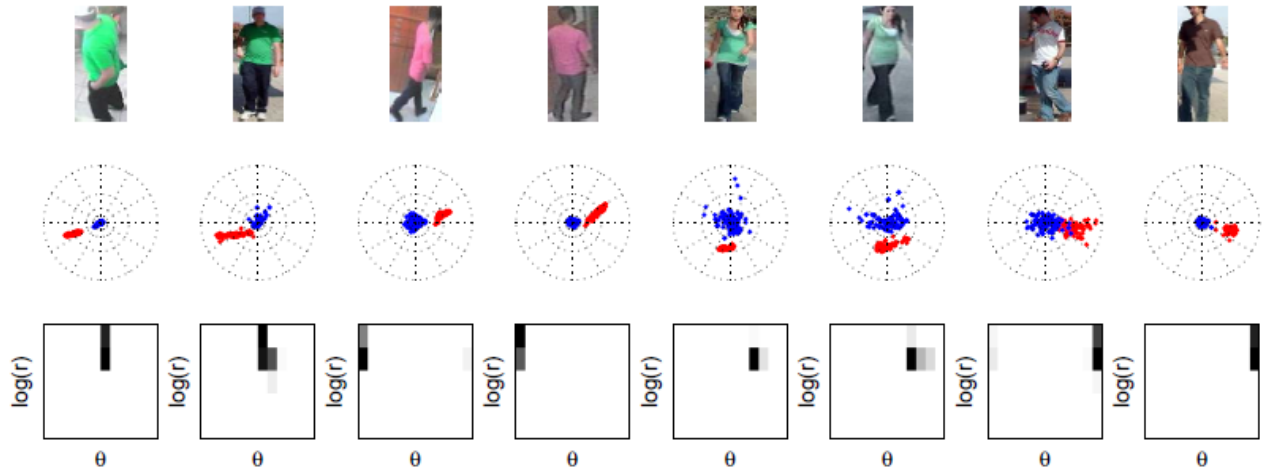
The input for this descriptor computation is two lists of coordinates which were extracted in the log-chromaticity colorspace extraction phase.
The extracted coordinates are denoted as $O = \{x_1, ..., x_N\}$.

The first list is a list of coordinates that were extracted from the image's upper part and referred as data coordinates. It is denoted as $O_U = \{x_i \mid l_i = 1\}$ where $l_i = 1$ indicates the upper origin of the coordinate.
The second list is a list of coordinates that were extracted from the image's lower part and referred as reference coordinates and denoted as $O_L = \{x_i \mid l_i = 0\}$.
Not all colorspace coordinates are used. Only a few samples from the upper part and the same amount from the lower part required.

A shape context descriptor is a set of data coordinates O and a single reference point x which marked as $sc(x, O)$. In order to achieve scale invariance all radial distances are divided by the mean radial distance from the reference coordinate.

When a set of shape context descriptor are computed for each reference point a parts based descriptor can be constructed. Its formal annotation is $PartsSC(O_{L,} O_U) = \{sc(x, O_U) \mid x \in O_L\}$.
It is a matrix of NxM where N is the number of reference (center) coordinates and M is the total number of bins. Each row represents a histogram of a specific shape context that was flattened into a single row.

This PartsSC descriptor or signature encodes the upper coordinate's positions relatively to the lower coordinates.



The above figure illustrates the $sc(x,O)$ structure.

Under every image there is a logpolar axis system which centers around one of the blue coordinates (coordinate taken from a lower part of the image). The red coordinates are counted and the histogram under the axis system is built. Notice how all of the red coordinates of the left-most image are located at $\theta=[180^o,200^o]$ and the histogram shows that too. The more coordinates at a specific radius there are, the more darkener the bin in the 2D histogram.

Another observation is that the two images which belong to the same dressed person have a similar shape of the clouds at the axis system and a similar logpolar histogram. Notice how the left two images have this similarity and the other two pairs on the right of it have it too.

When two PartsSC signatures are available the next step is to measure the distance between them.
The two signatures of the form $S=PartsSC=\{sc_i\}$ are composed of N members. For finding the distance between every member from S_1 to every member from S_2 a cost matrix Chi-Squared is computed. The computation results in a cost matrix C.

The formal definition of distance between the two signatures is
d(S_1,S_2) = The minimal cost matching all elements in S_1 with all elements in S_2 using the cost matrix C.

After the cost matrix is computed, the distance itself is computed by calculating the minimal cost using the EMD code.

Histogram Representation

Another method of representing an image is the histogram method.
This method's input is the pre-calculated log colorspace coordinates, and its output is the histogram representation.

For every lower/upper part the histogram is calculated separately.
Every coordinate from the input coordinates list are quantified to match the bins and the number of bins are counted.
Finally the histogram is normalized by division by its sum of bins values.
The histogram of a part (upper or lower) is NxN matrix's where N is the number of bins at each axis (x and y).

When both part histograms are calculated every one of it is flattened into a single row vector. Then, those two row vectors are combined together and form a 2xM matrix (M = #bins x #bins). This matrix is the Histogram representation of the image.

The intersection between two histograms, which represents the sum of minimal bins values between each two corresponding bins are denoted as:

$$H(h_1, h_2) = \sum_{k=1}^{n} \min(h_1(k),\ h_2(k))$$

k – Index of the compared bin.

H – Sum of the intersections between bin's value at h_1 and at h_2

The distance between two histogram representations are calculated as follows:

$$d_{Hist}(h_1, h_2) = 1 - H(h_1, h_2)$$

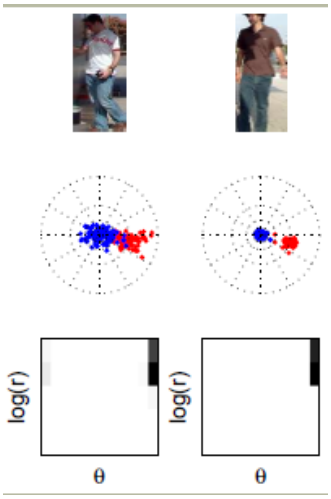
The bins are normalized thus the not intersected values are sums up to 1 and form the distance between the histograms.

The total distance between images using the histogram method is the sum of distances between the upper and the lower parts $D = d_U + d_L$

Covariance Representations

As presented at the PartsSC section, PartsSC is a way to construct a relative representation of an image. Therefore some failures can occur.

The figure from the right illustrates such case. Although the shape contexts of the images are relatively similar, the images are of different persons.



This failure occurs due to the relativeness of the partsSC representation. And of the colorspace proportional method of computing the coordinates

Therefore, an additional image information encoding is required.
The covariance representation advantage is that it encodes the absolute values of the image colors.

The process of extracting the covariance representation has two phases:
It takes as input the image and its silhouette and does the following:

1. Extracting covariance data of an image:

   a. A mask matrix of the same dimensions as the image itself is built.
   This matrix cells values are 1, 2, 3 … k and 0. Those values represent the significant colors that were found by applying the classical MeanShift algorithm on the image. The irrelevant pixels are marked with 0 and those pixels will be ignored

   b. Getting the mask from the previous stage the maximum value is found.
   This cell value is a value that appears the most on the mask matrix, its possible range is 1, 2 … k.

   c. Having found the most appearing cells value a covariance data matrix of the image can be produced.
   This matrix is produced by taken into account only the cell which holds the most appearing value. The cell's R, G, B data is taken along with the spatial information of the Row-Index location of the cell.
   This cell data for every relevant cell is combined into a matrix whose rows are of the form: $z_i = [R(x_i, y_i), G(x_i, y_i), B(x_i, y_i), y_i]$.
   The z_i's RGB values are normalized by its R+G+B.

2. This phase computes the covariance matrix from the covariance data that were produced in the previous phase: $C_R = \frac{1}{n-1}\sum_{k=1}^{n}(z_k - \mu)(z_k - \mu)^T$
   It results in a 4 x 4 covariance matrix.

The distance between two covariance matrixes is defined as:

$$d_{Cov}(C_1, C_2) = \sqrt{\sum_{i=1}^{d} \ln^2 \lambda_i(C_1, C_2)}$$

$\lambda_i -$ a generalized eigenvalue that was computed from $C_1 v_i = \lambda_i C_2 v_i$

Thresholds

In order to decide the right threshold for each representation several calculations were made. A statistical analysis was made on the VIPeR database. In this database there are 632 pairs of images from 2 cameras. For every image from one camera there is a corresponding image at the second camera.

For every pair of correct matching images the distance of PartsSC, Histogram and Covariance were measured and stored in local array. Then for this distance data it was calculated the minimum, maximum, average and some cumulative distribution of the correct distances.

The results for Parts SC, Histogram and Covariance:

```
=== Parts SC ===
minimum = 0.121036
maximum = 0.960732
average = 0.541328
Table:
0.1 >= 0 % (0)
0.2 >= 1.11 % (7)
0.3 >= 4.91 % (31)
0.4 >= 17.88 % (113)
0.5 >= 39.4 % (249)
0.6 >= 67.72 % (428)
0.7 >= 84.49 % (534)
0.8 >= 95.25 % (602)
0.9 >= 99.68 % (630)
1 >= 100 % (632)
```

- The minimum value of the distances between correct matching images that were record with the PartsSC representation = 0.121
- The maximum value = 0.96
- The average = 0.54
- The Table displays a cumulative measurement of how many distances were record under some [0.1, .. 1] threshold.
  For example: 0.5 >= 17.88 % (113). Means that 17.88% of the distances are under 0.5, and their number is 113.

```
=== Histogram ===
minimum = 0.0656996
maximum = 0.960129
average = 0.457901
Table:
0.1 >= 0.16 % (1)
0.2 >= 5.7 % (36)
0.3 >= 24.21 % (153)
0.4 >= 44.15 % (279)
0.5 >= 65.66 % (415)
0.6 >= 80.06 % (506)
0.7 >= 85.28 % (539)
0.8 >= 90.82 % (574)
0.9 >= 97.15 % (614)
1 >= 100 % (632)
```

```
=== Covariance ===
minimum = 0.0961872
maximum = 1
average = 0.355654
Table:
0.1 >= 0.32 % (2)
0.2 >= 10.44 % (66)
0.3 >= 37.82 % (239)
0.4 >= 71.99 % (455)
0.5 >= 86.39 % (546)
0.6 >= 94.3 % (596)
0.7 >= 97.31 % (615)
0.8 >= 98.1 % (620)
0.9 >= 98.73 % (624)
1 >= 100 % (632)
```

With the help of the above statistics the thresholds can be decided.

Parts SC – the table shows that most of the distances are under the 0.7 and 0.6. The average value is ~=0.5 therefore a value of 0.6 will be sensible enough to capture most of the correct matches.

Histogram – about 80% of the distances are below 0.6 and 65% under 0.5, the average is about 0.45 . Therefore a value of 0.5 is chosen

Covariance – Most of the distance are under the 0.5 ~ 0.4 and the average is 0.35 therefore a value of 0.45 is chosen.

Summary

Under the following threshold, the compared pair of images considered as matched.

PartsSC-Th = 0.6
Histogram-Th = 0.5
Covariance-Th = 0.45

Average-Th = 0.51

Android Application description

The previous section described the algorithm which is used by the android application. This algorithm is fully implemented with C++ and uses the openCV library.

On this section I will describe the user interface and the functionalities that the application.

User Interface & functionalities

At the upper left side it is seen the speed of the camera in FPS its current resolution
At the upper middle are the camera's name and the device IP and Port

In the middle of the screen are the two green boxes and a red one. The red box is used to focus on the captured person and the two green boxes show the captured pictures.

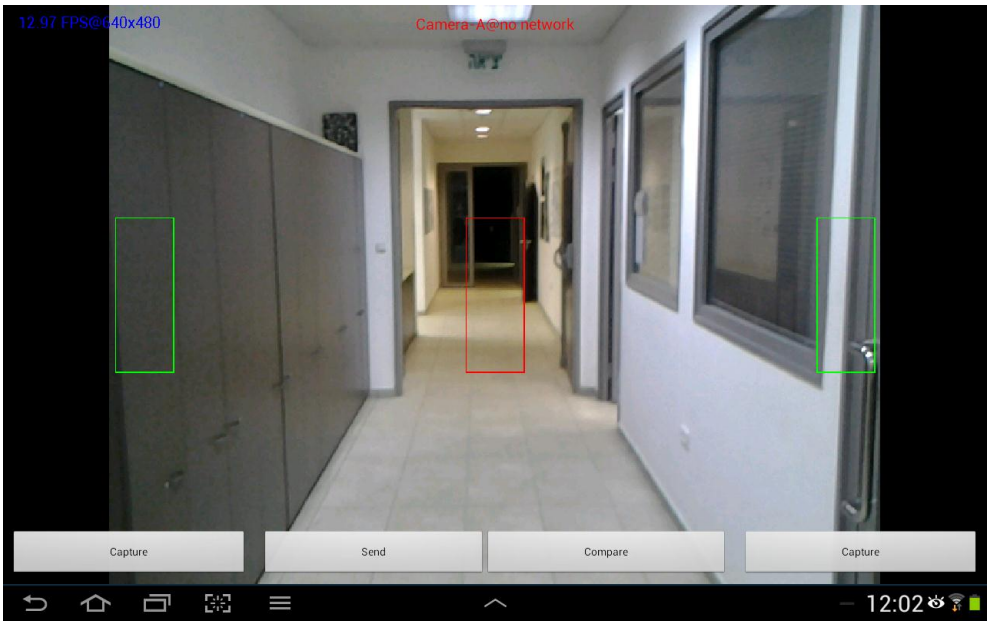
On the bottom there are 4 buttons. The two capture buttons are used to capture a picture and poses it in the corresponding green box.
The compare button is used to show the measured distance results for every method and the average distance of the combination.
The send button is used to send the captured picture to another such application.

Another system button exists, which allows changing the camera resolutions.

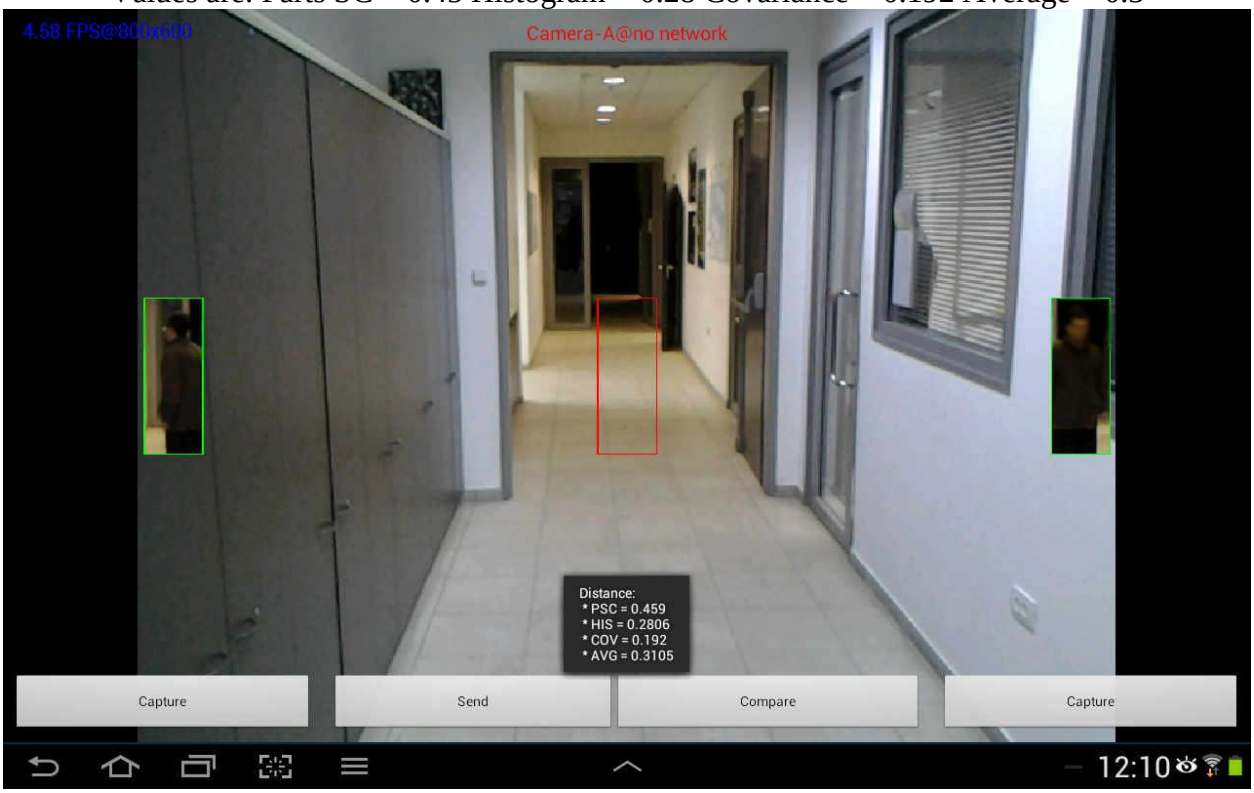
In total the supported functionalities are:

- Capturing 2 images and displaying it/
- Comparing the displayed images and showing the results of the comparison.
- Sending the captured picture representations to a distant device via sockets.
- Changing the camera preview resolution.

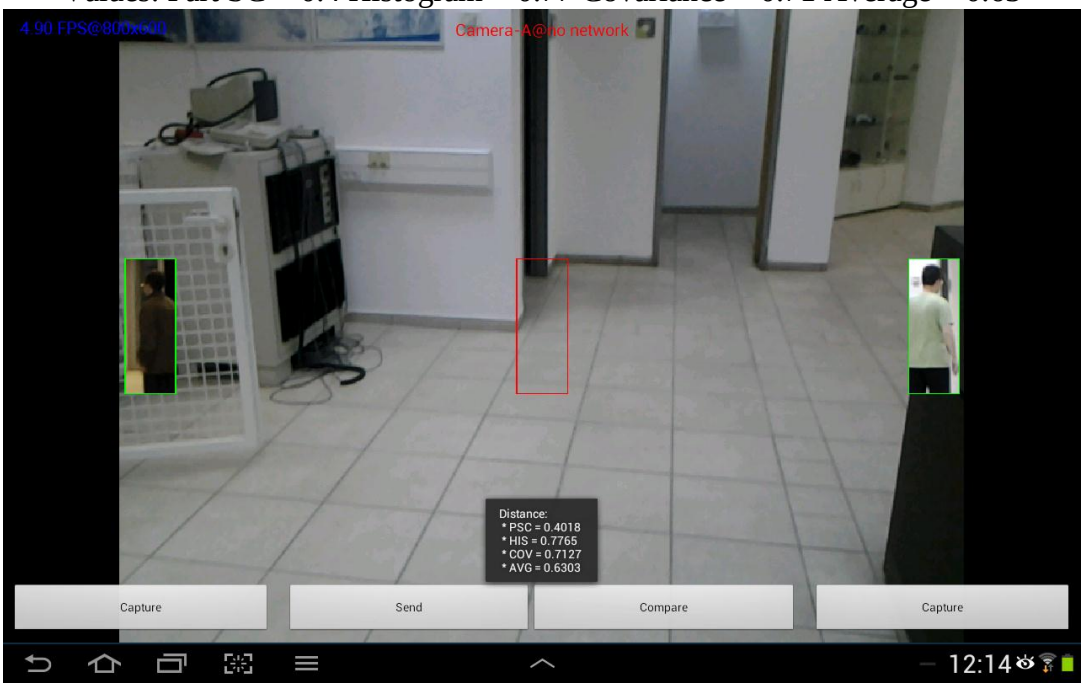
Scene of capturing



Same person capture
Values are: Parts SC = 0.45 Histogram = 0.28 Covariance = 0.192 Average = 0.3



Different clothes
Values: Part SC = 0.4 Histogram = 0.77 Covariance = 0.71 Average = 0.63



Communication

One of the key functionalities of the application is sending the captured picture to a distant device. In order to fulfill this demand the image and its representation are translated into an xml format. And send via socket to a distant device. At the distant device the xml is parsed and a class representing the image is created.

The format is of this form:

```
<?xml version="1.0" encoding="UTF-8"?>
<ReidFrame SourceName="camera-A" CapturedDate="21/2/12">

    <image rowsNum="" colsNum="" type="0"> </image>

    <sil rowsNum="" colsNum="" type="0"> </sil>

       <colorspace type="all">
           <upper rowsNum="" colsNum="" type=""> </upper>
           <lower rowsNum="" colsNum="" type=""> </lower>
       </colorspace>

       <colorspace type="samples">
           <upper rowsNum="" colsNum="" type="0"> </upper>
           <lower rowsNum="" colsNum="" type="0"> </lower>
       </colorspace>

       <hist rowsNum="" colsNum="" type=""> </hist>

       <parstssc rowsNum="" colsNum="" type=""> </parstssc>

       <cov rowsNum="" colsNum="" type=""> </cov>
</ReidFrame>
```

The xml contains the following:

- Image matrix
- Silhouette matrix
- Colorspace all coordinates
- Colorspace just the random selected coordinates
- Histogram representation of the image
- PartsSC representation of the image
- Covariance representation of the image

The xml block also contains the camera source name and the date when the picture was captured. All the representations are matrixes, for every matrix its type and the number of rows and columns are attached.

The presented above is the full xml format. In practice it is unnecessary to send all data, and just part of the data is actually sent. Using the full format allows checking that the correct image was passed and debugging via seeing the actual image.

In order to minimize the amount of transferred data the following calculations were made.
Based on those calculations I have chosen which data to send.

Image & Silhouette transfer requires:

- The images that are supported in this project are size of 128 x 48, each pixel has 3 values (actually 4, the fourth is an alpha value which can be ignored) => 128 x 48 x 3 = 18432 numbers.
  The silhouette is with the same dimensions 128 x 48 and holds a
  Binary values => 128 x 48 = 6144 numbers cost of transferring it.
- Conclusion, 18432 + 6144 = 24576 numbers to transfer. It turns out to be the most expensive way to compare two distant images. The image should not be transferred.

PartsSC transfer requires:

- Colorspace – only the sample coordinates required, in total 85 x 2 = 170 coordinates, each coordinate has x and y => total transfer of 170 x 2 = 340 numbers.
- PartsSc Representation – for 85 reference (centers) coordinates and 120 bins which is used in this project => total transfer of 85 x 120 = 10200 numbers
- Conclusion, it is better to transfer the colorspace of partsSC and compute the representation on the distant device.

Histogram transfer requires:

- Colorspace – the full colorspace should be transferred if the computation of the histogram will be done at the distant device. The colorspace consist of 128 x 48 coordinates which eventually result in a matrix of 128 x 48 = 6144 rows and 2 columns. Totally 12288 numbers to send.
- Histogram – consist of 2 rows and 100 columns (bins). In total 200 numbers to send.
- Conclusion, the final Histogram will be sent, sending just 200 numbers.

Covariance transfer requires:

- Covariance data – the number of pixels of the most significant color is unknown, but from practice the numbers are about few hundreds, the number of columns is constant and equals to 4. In average about 500 x 4 = 2000 numbers.
- Covariance Matrix – the method for its computation results in a matrix of just 4 x 4 = 16 number to send.
- Conclusion – the covariance matrix of 16 numbers should be sent.

Summary of sent data:

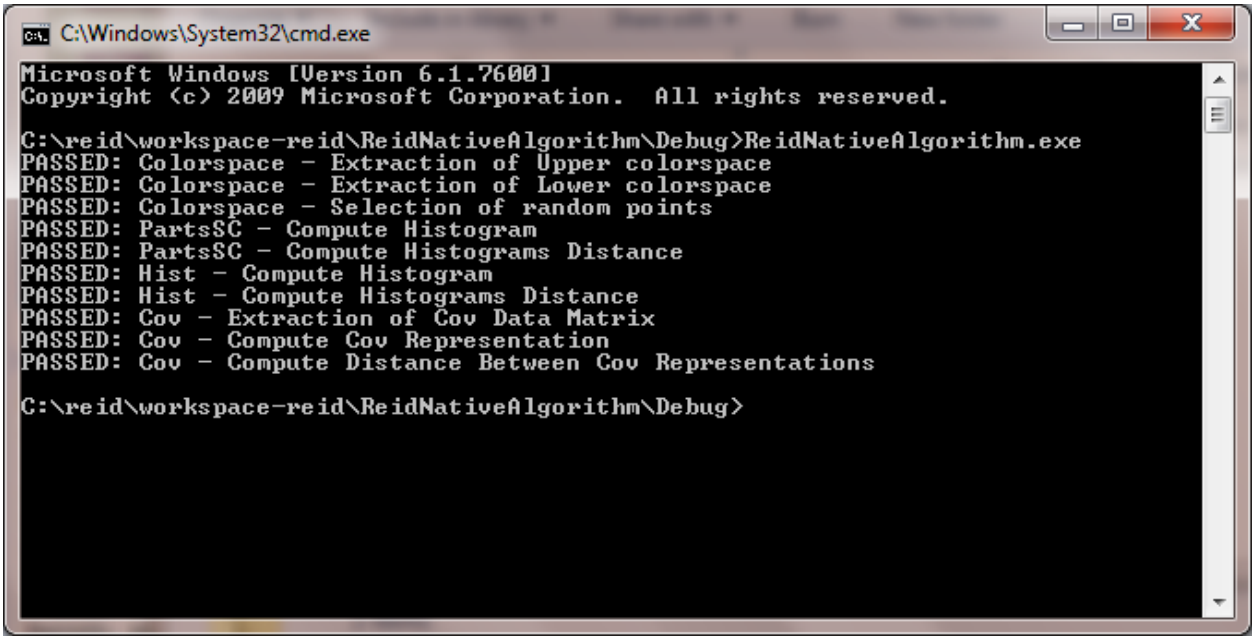
- PartsSC – the colorspace is sent and the representation is computed at the distant device. Sent amount is 340 numbers.
- Histogram – The representation which has 200 numbers is sent.
- Covariance – The representation which has 16 numbers is sent.
- Total data sent => 340 + 200 + 16 = 556 numbers

As the above calculation shows, sending the representations (and the partsSC colorspace) is much cheaper than sending the original image and its silhouette (total sent of 24576 numbers). It is actually 24576 / 556 =~ 44 times less information to send.

** For the above calculations it is assumed (and implemented like this) that the upper ROI and the lower ROI cover the whole image and split the image into two almost similar size parts.
Even if the ROI sections will be smaller, the above calculations will not change in the final conclusions regarding of what data to send.

API Testing

As mentioned in the introduction section, this project is based on a previous project done with matlab. In order to test that the current project's computations results the same output as the matlab I made some tests. Those tests check the interface provided by the c++ code to the android via the JNI layer. The interface receives some input and it checks that for the same input the algorithm produces the same output as matlab produce.



Conclusions

1) Comparing images based on the PartsSC and Histogram representations inhibits incorrectness. Additional representation such as Covariance can help to overcome it.

2) Using the average value for distance evaluation results in a more precise distance between the images.

3) The algorithm produces different image representations that afterwards are compared. Those representations size in bytes are much lesser than the original image size, thus sending it to a distant computer is relatively much cheaper.

References

1. Igor Kviatkovsky, Amit Adam and Ehud Rivlin, "Color Invariant for Person Re-Identification"

2. http://www.cs.technion.ac.il/~kviat/colorReid.htm

3. The VIPeR database http://vision.soe.ucsc.edu/projects

4. http://opencv.org/

5. http://developer.android.com/index.html