# Programming Exercise 3
# Fingerprint Matching (8 Pts)

Deadline: 12th February 2015 - 11:59 p.m.

Methods of User Authentication

Research Group: Mobile Security

Winter Semester 2015/16

Prof. Dr. Markus Dürmuth, Maximilian Golla, Jan Rimkus

Task: Write a program that implements a simple minutiae-based fingerprint matching system using the generalised hough transform.

- We will provide a C framework for this task
- We compile with the latest `gcc` (e. g., 4.8.4) running on Ubuntu 14.04
- Compile string: `gcc -O2 simple_matcher.c -lm -o simple_matcher`
- The framework includes all necessary files (fingerprints, method skeletons)
- Only submit your code, i. e., please do not submit the `*.xyt` files
- Naming: `Fingerprint_<Team-xx>.c`
  (Example: Team 1, `Fingerprint_Team-01.c`)

Just upload your submission to the Moodle course. Please be aware that you can only submit once!

Fingerprints can be compared by matching their minutiae. To simplify this exercise, we will provide fingerprint scans (images) and their corresponding minutiae (text) via Moodle.

**Hints for Grading 3.1** Get Familiar With the `*.xyt` Files, 0 Points

The provided `*.xyt` files were generated using a fingerprint minutiae detector called MINDTCT, which is part of the „NIST Biometric Image Software“ (NBIS)[1], and the fingerprint scans from the „Fingerprint Verification Competition“ (FVC)[2] 2004. A file includes one minutia per line and is structured as follows:

```
x coordinate, y coordinate, angle theta in degree, quality factor [1-99]
```

Note: We do not process the quality factor in our algorithm, just ignore it.

**Hints for Grading 3.2** Read the Content of the `*.xyt` Files, 1 Point

Before we can start, we need to read the minutiae from disk. Please use the method skeleton `loadMinutiae()` for this purpose. Your task is to parse the files and store their values in the provided struct. Furthermore you need to implement some kind of integrity check for the case that a file contains invalid data. As every `*.xyt` file (fingerprint scan) contains a different number of minutiae (rows) you should check that only `MAX_MINUTIAE` are considered, discard the rest. You can find more on this topic in the actual method skeleton and in the programming tutorial slides.

[1]`http://www.nist.gov/itl/iad/ig/nbis.cfm` – NIST Biometric Image Software, as of December 18, 2015.

[2]`http://bias.csr.unibo.it/fvc2004/default.asp` – Fingerprint Verification Competition, as of December 18, 2015.

**Hints for Grading 3.3** Calculate the Score, 3 Points

Now that the minutiae are accessible via the struct we can compare our probe fingerprint with one from the gallery of available fingerprints. Please use the method skeleton `getScore()` for this purpose. The scoring algorithm is very simple and was described in the lecture. An example can also be found in the lecture notes. If you are curious you can find more information in [MMJP09, pp. 177]. The algorithm requires empirical values which are provided as constants, e. g., `threshold_d` (displacement), `threshold_r` (rotation).

**Hints for Grading 3.4** Apply the Generalized Hough Transform to Improve Results, 4 Points

To improve the matching, the Generalized Hough Transform (GHT) [MMJP09, pp. 184] can be applied before calculating the score. A higher score is achieved by a better alignment transformation, using displacement and rotation. The GHT algorithm was described in the lecture. However, you should read the provided tutorial slides as they include a hint. Please use the method skeleton `alignment()` for this purpose.

Here is a small example how the fingerprint matcher output may look like:

```
user@pc:~$ ./simple_matcher -p images/101_1.xyt -g images
Looking for images in dir:  images
109_4.xyt, full:  images/109_4.xyt
101_7.xyt, full:  images/101_7.xyt
101_6.xyt, full:  images/101_6.xyt
...
Node:  0x868760, Path:  images/104_4.xyt, Score:  16
Node:  0x86da40, Path:  images/102_2.xyt, Score:  17
Node:  0x878660, Path:  images/110_8.xyt, Score:  17
Node:  0x866780, Path:  images/104_8.xyt, Score:  19
Node:  0x870080, Path:  images/101_1.xyt, Score:  76
```

Using the Generalized Hough Transform (alignment function) the results should improve:

```
user@pc:~$ ./simple_matcher -p images/101_1.xyt -g images -h
Looking for images in dir:  images
109_4.xyt, full:  images/109_4.xyt
101_7.xyt, full:  images/101_7.xyt
101_6.xyt, full:  images/101_6.xyt
...
Node:  0x17409e0, Path:  images/101_4.xyt, Score:  32
Node:  0x174d5e0, Path:  images/101_3.xyt, Score:  40
Node:  0x1731140, Path:  images/101_2.xyt, Score:  44
Node:  0x1746fe0, Path:  images/101_5.xyt, Score:  44
Node:  0x173d080, Path:  images/101_1.xyt, Score:  76
```

## Bibliography

[MMJP09] Davide Maltoni, Dario Maio, Anil K. Jain, and Salil Prabhakar. *Handbook of Fingerprint Recognition*. Springer Science & Business Media, Berlin Heidelberg, 2 edition, 2009.

A very brief extract (snippet.pdf) of this book can be found in Moodle, please use the password “**sicheres-passwort**” to read it. Please note, you are **not allowed to redistribute or print** it.