Mini-Projet : Empreintes digitales

24 novembre 2025

Plan

- Administration
  - Informations générales/Point de départ
  - Groupes et soumissions
  - Portail de soumission
- Projet
  - But et algorithme à implémenter
  - Code fourni
  - Etapes du projet
    - Étape 1 : Squelettisation
    - Étape 2 : Localisation et orientation des minuties
    - Étape 3 : Comparaison d’empreintes

Échéances

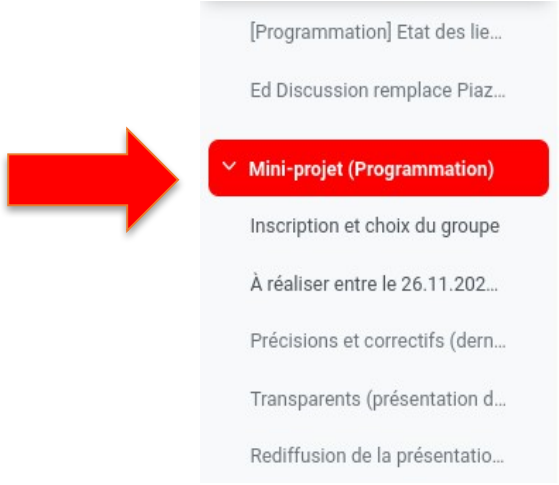
- Publication du matériel : Lundi 24 Novembre
- Séance TP du 25.11 entièrement dédiée au lancement du mini-projet
- Dernier mardi du semestre (16.12) dédié à la finalisation
- Ouverture du portail de soumission : **Lundi 8.12**
- **Délai de soumission : Mercredi, 17.12, 13h.**

Ressources pour le projet

- Toutes les informations et ressources sont sur Moodle
  - Archive Zip avec le matériel fourni et le fichier à compléter
  - Description (document .pdf) expliquant le matériel fourni et les tâches à réaliser
  - Annexe du .pdf : rappel de l'algorithme à implémenter
- Avant de commencer lisez la documentation fournie complètement et avec soin !

Soumission

- **Deadline: Mercredi 17.12, 13:00**
- Groupes d'**au plus 2 étudiants**
- Les instructions pour la soumission seront sur Moodle



Contenu de la soumission

- UN SEUL FICHIER
  - fingerprint.cpp

Veillez à ne pas faire dépendre votre code de nouveaux ajouts dans les autres fichiers fournis !

Portail de soumission

- Sera ouvert le **8.12** sur Moodle
  - **Pas de garantie de réponse rapide en cas de problèmes sur des soumissions faites le week-end**
- Fermeture le **17.12 (délai strict)**
- Vous pouvez soumettre autant de fois que vous voulez
- **Tâche 0**: Soumettez dès l'ouverture du portail même avec une version très incomplète. Le but est de vous familiariser avec le protocole de rendu et de **vérifier que votre code compile bien sur les machines de correction**

Soumission – Plagiat

- Le projet est noté
- Les discussions et échanges d'idée entre groupes est permis et recommandé
- **L'échange de code est strictement interdit!**
- **Le plagiat de quelque nature que ce soit sera contrôlé** et considéré comme **tentative de tricherie.**
- En guise de sanction la note "NA" sera attribuée:
  Art. 18 "**Fraude de l'ordonnance sur la discipline**"
  https://www.admin.ch/opc/fr/classified-compilation/20041650/index.html
- **Vous devez en tout temps être capable d'expliquer le code que vous avez produit.**

Notation

Le projet comporte 3 parties obligatoires:

- Étape 1 : Squelettisation : **40 points**
- Étape 2 : Localisation et orientation des minuties : **30 points**
- Étape 3 : Comparaison d’empreintes : **30 points**

- Fixez vous des **objectifs raisonnables** en fonction de votre niveau!

Aide pendant les TPs/ en semaine

- Les assistants n’ont pas pour tâche de résoudre vos erreurs d’exécution
- Ils peuvent vous aider sur les fautes de compilation
- Et vous aider à développer des stratégies pour débusquer les sources des erreurs par vous même
- Il y a une **équipe très limitée en appui**: **privilégiez le forum Ed pendant la semaine**

Plan

- Administration
  - Informations générales/Point de départ
  - Groupes et soumissions
  - Portail de soumission
- Projet
  - But et algorithme à implémenter
  - Code fourni
  - Etapes du projet
    - Les structures de données
    - Étape 1 : Squelettisation
    - Étape 2 : Localisation et orientation des minuties
    - Étape 3 : Comparaison d’empreintes

But du projet

- Écrire une programme capable de détecter si deux empreintes digitales proviennent du même doigt

Structure d’une empreinte digitale

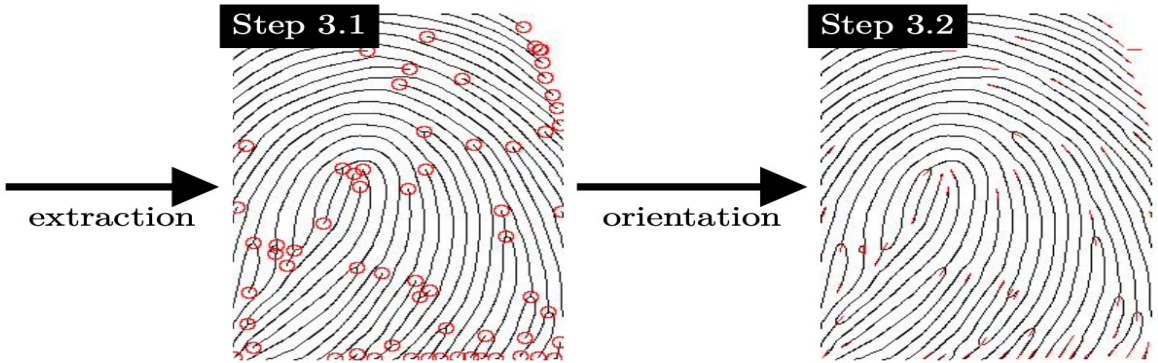
- Lignes de crête
- Un point où une ligne de crête change est appelé **minutie**
- Les minuties sont utilisées pour déterminer **le caractère unique** d'une empreinte digitale.
- Il en existe différents types, par exemple : **terminaison de crête, bifurcation de crête**, **étang**,...
- Les empreintes digitales de haute qualité contiennent entre 25 et 80 points caractéristiques.

Comment comparer deux empreintes ?

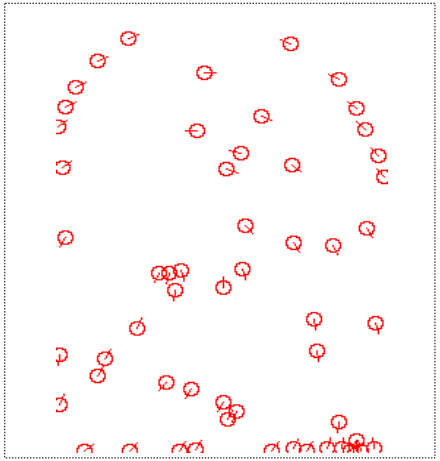
- **Extraire la liste des minuties** de chaque empreinte digitale
- Une minutie est représentée par son **emplacement** et son **orientation**.
- **Comparer les deux listes** de minuties et compter le nombre de minuties qui correspondent (c'est-à-dire dont l'emplacement et l'orientation sont similaires*).

** Similaires signifie que la différence entre elles est inférieure à un seuil donné.*

Extraction des minuties

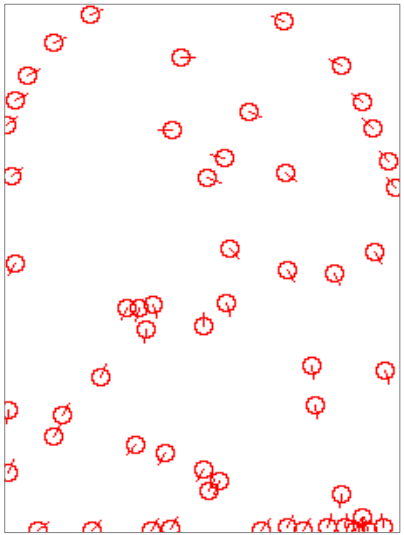


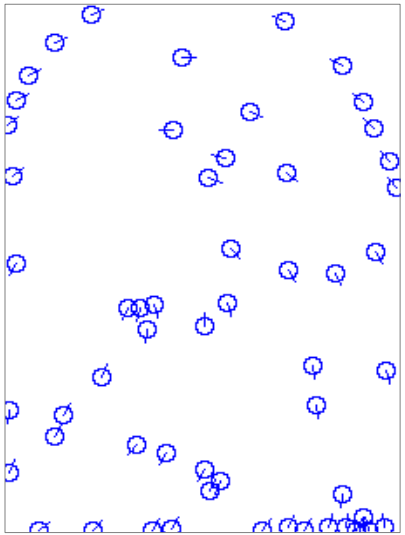
Résultat: liste de minuties



Mise en correspondance

Input : deux listes L_1 et L_2 de minuties

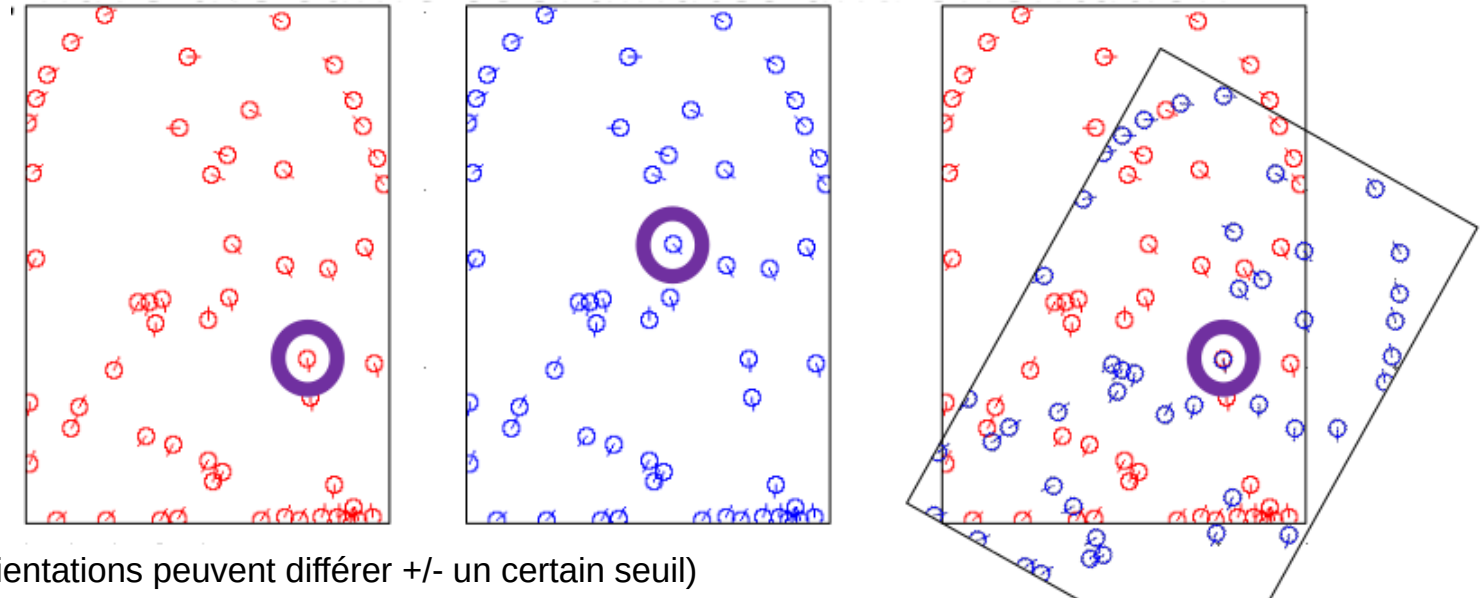




Algorithme **«Brute-force»**:
Pour chaque paire de minuties (m_1,m_2) de L_1xL_2, on teste si m_1 «correspond» à m_2

Mise en correspondance de minuties

- Une minutie m_1 correspond à une minutie m_2 (elles “match”), *si elles se superposent* (c-àd. qu’elles sont proches à un certain seuil près) **lorsque m_1 et m_2 sont alignés**
- Pour aligner m_1 et m_2, on transform L_2 de sorte à ce que m_1 et m_2 aient les **mêmes coordonnées et orientation***



(*les orientations peuvent différer +/- un certain seuil)

Plan

- Administration
  - Informations générales/Point de départ
  - Groupes et soumissions
  - Portail de soumission
- Projet
  - But et algorithme à implémenter
  - Code fourni
  - Etapes du projet
    - Les structures de données
    - Étape 1 : Squelettisation
    - Étape 2 : Localisation et orientation de minuties
    - Étape 3 : Comparaisons d’empreintes

Fichiers fournis

- fingerprints.zip contient :
  - **Cmakelists.txt**: fichier pour compiler avec QtCreator
  - **Makefile.geany**: fichier pour compiler en ligne de commande ou avec Geany
  - Fichiers .hpp or .cpp (voir transparents suivants)

- Vos contributions seront dans un **seul fichier**
  mais vous travaillerez dans un environnement “Projet”

  - Plusieurs fichiers sont impliqués lors de la compilation
  - Conformez vous aux directives d’installation décrites dans le descriptif général du projet.

Code fourni (1/2)

Ne pas modifier !

- **`utils.hpp, utils.cpp,stb_image.h, stb_image_write.h, helper.h, helper.cpp :`** utilitaires de traitement d'images et de test
- **`test_helper.hpp, tests_helper.cpp :`**
  - differentes fonctions de conversion en string
  - `pour faciliter les affichages nécessaires aux tests`

Code fourni (2/2)

- **`fingerprint.hpp`** :
  - Contient les **types** et les **prototypes** des fonctions à coder
    Ne pas modifier !
- **`fingerprint.cpp` : C'est ici que vous codez ! Ce fichier est noté !**
  - contient les corps vides des fonctions à coder
  - si nécessaire vous pouvez ajouter vos propres types et prototypes
- **`main.cpp`** : Fichier non noté !
  - tests fournis pour commencer à tester votre code.
  - **Ces tests ne sont pas exhaustifs, vous devez les compléter.**

Plan

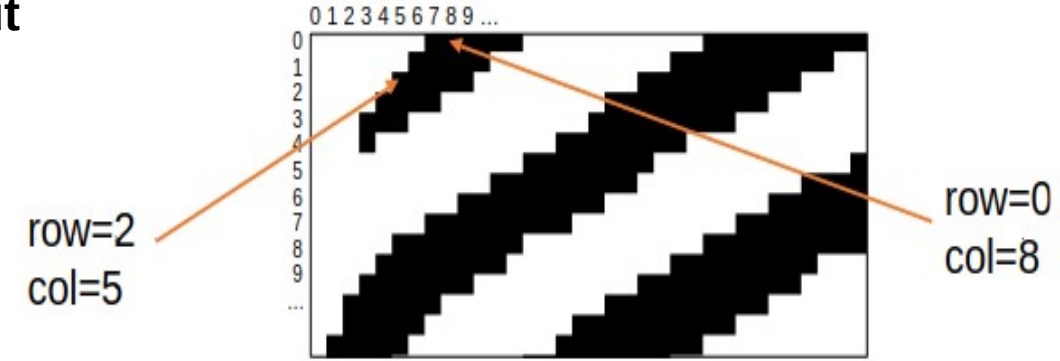
- Administration
  - Informations générales/Point de départ
  - Groupes et soumissions
  - Portail de soumission
- Projet
  - But et algorithme à implémenter
  - Code fourni
  - Etapes du projet
    - Étape 1 : Squelettisation
    - Étape 2 : Localisation et orientation de minuties
    - Étape 3 : Comparaison d’empreintes

Représentation des données

Une image d'empreinte est un ensemble de **pixels** (points d'image)

Les empreintes sont stockées dans des tableaux de booléens à deux dimensions

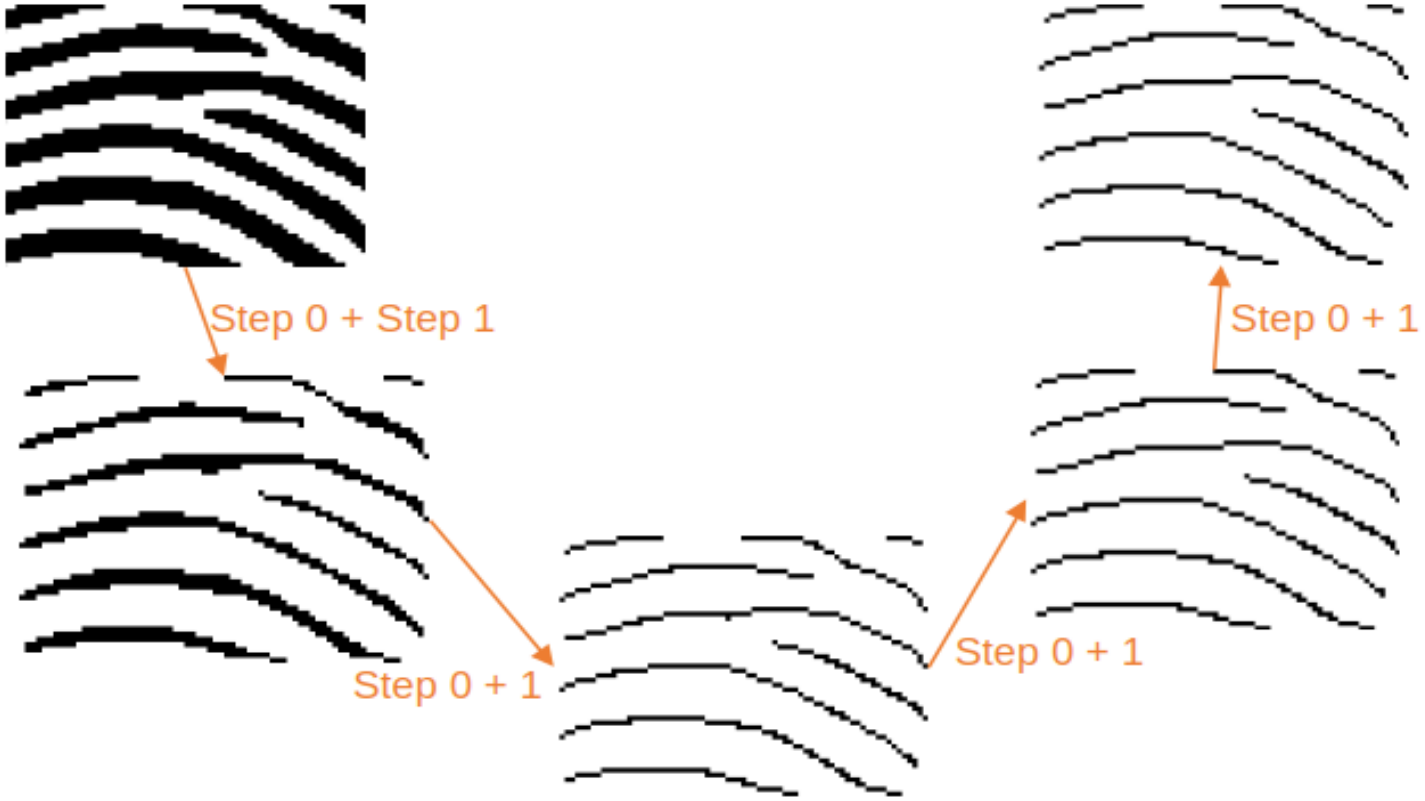
- true correspond à un pixel **noir** (existant)
- false correspond à **blanc** (non-existant)
- image[0][0] (ligne 0, colonne 0) est le **coin supérieur gauche** !
- image [0] est la **ligne du haut**



Étape 1 : Squelettisation

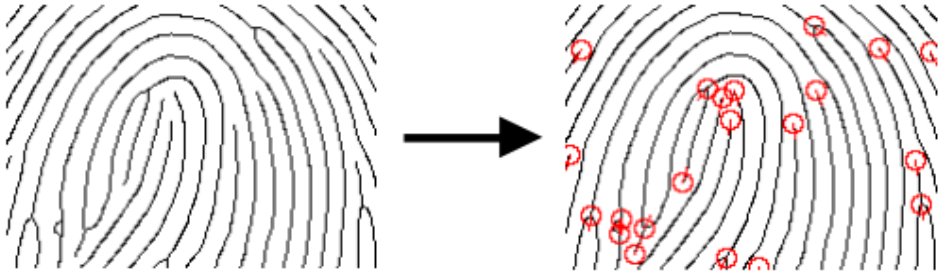
- L’algorithme procède par **itération.**
- Chaque itération comporte deux étapes **(***Step 0* + *Step 1*, explicitées dans l’énoncé).
- À chaque étape, chaque pixel est analysé pour décider s’il doit être supprimé ou pas. **La décision dépend des pixels voisins.**
- Si lors de l’itération courante un pixel disparaît, les deux étapes se répètent.

Étapes de la squelletisation



Étape 2 : extraction des minuties

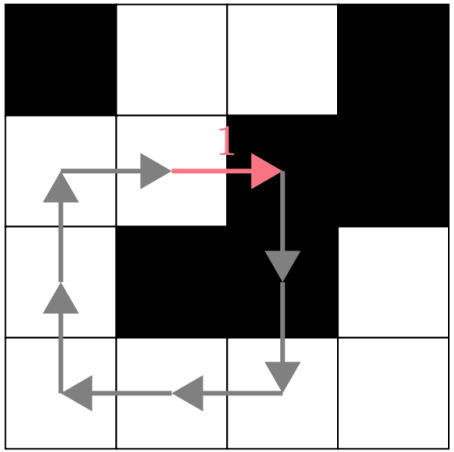
- Entrée : image squelettisée
- Sortie : une liste de minuties, chacune étant donnée par sa ligne, sa colonne et son orientation (stockée sous la forme d'un tableau de à entrées).

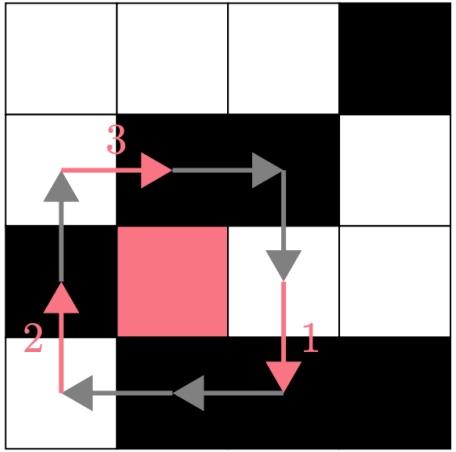


row=11	col=130	o=315
row=23	col=164	o=301
row=24	col=9	o=227
…	…	…

Localisation d’une minutie

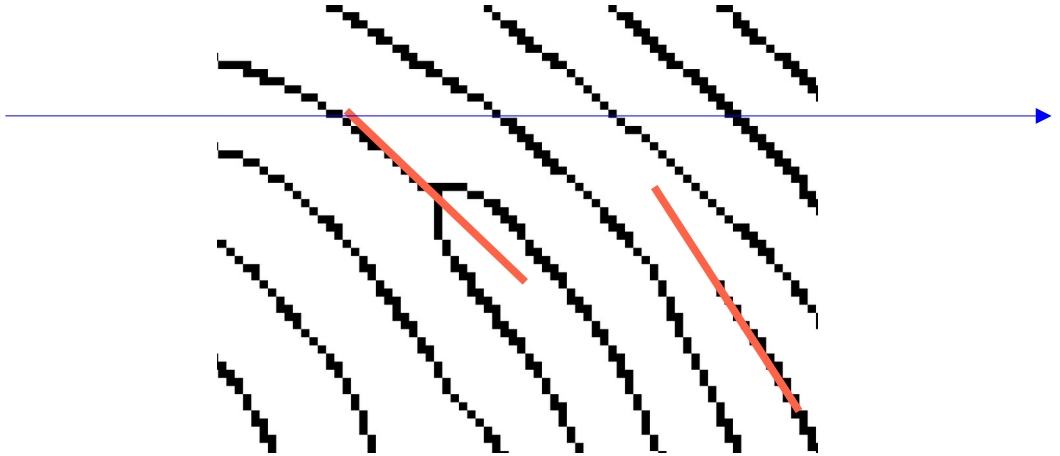
- On ne considère que les pixels **qui ne sont pas dans les bords**
- On considère deux types de minuties: fin de crête ou bifurcation
- on identifie la minutie sur la base du nombre de transitions dans son voisinage:
  - **1 transition blanc vers noir** → fin de crête
  - **3 transitions blanc vers noir** → bifurcation



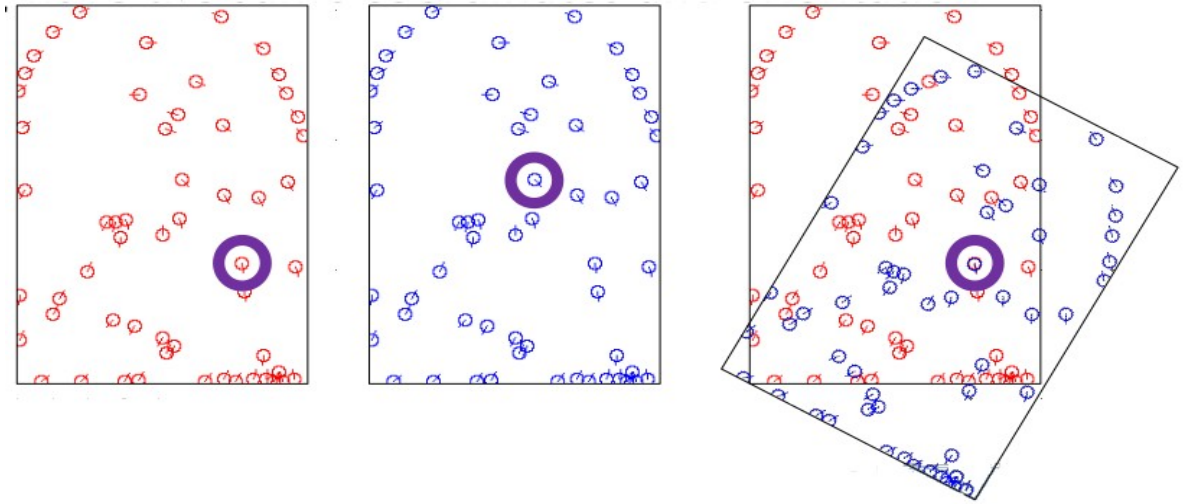


Orientation d’une minutie

Idée: examiner les pixels qui sont
(i) proches et **(ii) connectés** à la minutie et les approximer avec une ligne. L’**angle** que fait cette ligne avec l’axe des x est l’**orientation** de la minutie.



Étape 3 : comparaison



Il faut:

- transformer (rotation et translation) les minuties de la seconde liste et
- contrôler si elles sont «proches»

Proximité de deux minuties

Des seuils de proximités sont pre-définis

- distance maximale tolérée Dmax
- différence d'orientation maximale tolérée Omax

Deux minuties sont proches

- la distance Euclidienne qui les sépare est d'au plus Dmax

$$\sqrt{(\mathtt{row}_1 - \mathtt{row}_2)^2 + (\mathtt{col}_1 - \mathtt{col}_2)^2}$$

- Leurs orientations diffèrent d'au plus Omax

Conseils

- Lisez et comprenez le but de chaque tâche complètement avant de vous lancer dans le codage
- Lisez et comprenez les tests de **main.cpp** pour bien comprendre ce qui est attendu
- N’hésitez pas à commenter les appels à certaines fonctions dans les tests fournis pour tester les étapes encore incomplètes
- Soumettez régulièrement sur le portail de soumission dès son ouverture
- Utilisez le debugger (à combiner avec des instructions d’affichage)
- Sauvegardez régulièrement votre travail
- N’hésitez pas à enrichir les tests fournis dans **main.cpp**

Bon codage !