BIOMETRISCHE MERKMALE

Zeigt her eure Hände

Mit dem FPR-Framework können Sie selbst mit Fingerabdrücken experimentieren.

War der Fingerabdruck vor einigen Jahren noch etwas Besonderes, so verlassen sich heute nicht nur Zugangskontrollen, Mechanismen der Cybersicherheit oder die Kriminalistik auf ihn, sondern fast jedes moderne Smartphone. Anwender nutzen ihn heute ganz selbstverständlich.

Es ist inzwischen aber auch so einfach, egal bei welcher Art von Anwendung, ob Handys zu entsperren sind, der Zugang zum Firmengebäude kontrolliert oder ob die Zeit der Anwesenheit bestimmt werden soll. Der Fingerabdrucksensor macht das Leben einfacher und bequemer – einfach Finger auflegen und schon werden Daten erfasst und das Handy oder die Zugangskontrolle aktiviert.

Bei Fingerabdrucksensoren stellen kapazitive und optische Scanner die gängigsten Geräte zum Erfassen dar. Beide machen ein Bild der Fingerkuppe, um dieses beim Abgleichen zurate zu ziehen, wenn ein neuer Finger auf den Sensor gelegt wird. Die Technik in der Sensor-Hardware unterscheidet sich jedoch stark je nach Hersteller und Anwendungsbereich.

Die Grundidee ist aber immer dieselbe: Es werden die wichtigsten Unterscheidungsmerkmale von Fingerabdrücken, die Singularitäten und Minutien, aufgenommen und extrahiert. Die Vorlage, die aus diesem Fingerabdruck entsteht, kann gespeichert und zu einem späteren Zeitpunkt mit einem weiteren Fingerabdruck verglichen werden.

Neben der vielfältigen Hardware gibt es auch eine große Anzahl von Fingerabdruckalgorithmen. Bild 1 zeigt eine nur kleine Auswahl der möglichen Algorithmen. Darunter finden sich Implementierungen in fast jeder aktuellen Programmiersprache wie zum Beispiel Python, Java, C, C++ und natürlich auch C#. Inzwischen gibt es darüber hinaus auch eine Vielzahl von neuronalen Netzen und Machine-Learning-Modellen zum Bestimmen von Fingerabdrücken.

1. 123 ID (USA) [CVT]
2. Acter (Switzerland) [acterBIOLIB]
3. ActivCard (Canada)
4. Aldebaran Systems (USA) [participated to FVC2002]
5. Antheus Technology (Brasil, USA) [Agora]
6. APRO Technology (Japan)
7. AST (Spain) [ASTAS]
8. Astro Datensysteme (Germany) [BioTouch]
9. Av@Ion Systems / Semantic system (Switzerland) [Ultramatch] [participated to FpEVT 2003, NIST]
10. Beijing BaiXinTong Electronic Tech (China) [participated to FVC2006]
11. Beijing Smackbio Technology (China) [Smackfinger] [participated to FVC2006]
12. Beijing HanWang Technology (China) [participated to FVC2004]
13. Bergdata (CDVI) (Germany, France) [bdfis]
14. BeyondLSI (Japan)
15. BioCert (USA?) [BioCert]
16. BIO-Key (USA) [VST]
17. Biolink (USA) [U-Match]
18. Biometrix (Austria) [BioCheck]
19. Bionopoly (USA)
20. Bioscrypt (Canada, USA) [Bioscrypt Core, V-Pass...]
21. Bromba (Germany) (spin-off from Siemens, end 2003)
22. CASIA Institute of Automation, Chinese Academy of Sciences (China) [participated to FVC2004]
23. Casio (Japan) [VeriPat]
24. CBA-Japan (Japan) [FCHIP2]
25. The Chinese University of Hong-Kong (Hong-Kong) [for smartcard (2004)]
26. Cogent (USA) [Bioswipe] is 3M (2010)
27. ComnetiX (Canada)
28. Cottonwood (CCS) (seems no more available 03/2004)
29. Count Me In (USA) [LightningID] (Digital Persona?)
30. CrossMatch (USA) [ID 500...]
31. Daimin (Korea)
32. Dalian University of Technology (China) [FVC2006]
33. Datamicro (Russia) [participated to FVC2002, FVC2004]
34. DDS Digital Development Systems (Japan) [UBF]
35. Dermalog (Germany)
36. Beijing Fingerpass / labs: Digital Fingerpass Corporation (China)
37. DTK Digital Tech Korea
38. Ekey (Austria) [TOCAxxx]
39. Fidelica (USA) 27 feb 2004: announced the availability of the FBA-4001 matching algorithm.
40. Fingerpin AG (Switzerland) (May 2004: access problem to website)
41. FIST (Korea) [Finguard]
42. Fujitsu (Japan)
43. Futronic Technology (China) [participated to FVC2004]
44. Genologic (Germany)
45. Gevarius (Russia) [participated to FVC2004]
46. Golden Finger Systems (USA?) [participated to FpEVT 2003, NIST]
47. Griaule (Brasil) [Pequi]

Eine Auswahl von verschiedenen Algorithmen (Bild 1)

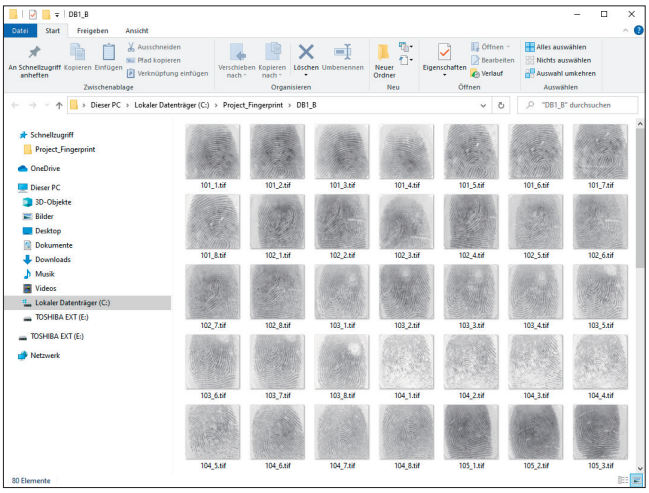
Extractor und Matcher

Aber auch bei der Vielzahl von Erkennungsmöglichkeiten für den Fingerabdruck erfordert das Erkennen zwei Hauptteile, den sogenannten Feature Extractor und den Fingerprint Matcher.

Der Feature Extractor importiert das Fingerabdruckbild als Rohbild und codiert es in ein entsprechendes Bildformat. Der Zweck des Matchers besteht darin, zwei Fingerabdruckvorlagen zu vergleichen. Er verwendet nicht das ursprüngliche Bild des Fingerabdrucks, sondern die resultierenden Vorlagen. Daraus erzeugt er dann eine Ähnlichkeitsbewertung, die angibt, ob zwei Fingerabdruckvorlagen von demselben Finger stammen oder nicht.

Das FPR-Framework

Die Suche nach Veröffentlichungen zur Fingerabdruckerkennung und -verifizierung fördert auch eine Reihe von Open-Source-



Der Inhalt der Datenbank DB1_B (Bild 2)

Algorithmen zutage, aber noch mehr Angebote von kommerziellen Anbietern, die Module oder Bibliotheken anbieten.

Für diesen Beitrag wird das FPR-Framework (FPR: Fingerprint Recognition) von Miguel Angel Medina Pérez und seinen Mitarbeitern verwendet. Pérez stellt dieses Framework, das vollständig in .NET und C# implementiert ist, zu Forschungszwecken zur Verfügung [1].

Das Framework verfügt sowohl über einen Feature Extractor, der es ermöglicht, ein Fingerabdruckbild aus einem entsprechenden Bildformat zu laden, als auch über eine große Auswahl von verschiedenen Fingerprint-Algorithmen, die dann als Matcher eingesetzt werden können.

Die Kernfunktionen des von Pérez vorgestellten Frameworks dienen dem Verifizieren von Fingerabdrücken. Hierzu können die Datenbanken A von FVC2002 und FVC2004 sowie die Datenbanken B von FVC2000, FVC2002 und FVC2004 eingesetzt werden.

FVC steht für Fingerprint Verification Competition und ist ein internationaler Wettbewerb für Fingerabdruck-Verifizierungsalgorithmen. Organisiert wurde er das erste Mal 2000 und danach noch dreimal in einem Abstand von zwei Jahren (FVC2002, FVC2004 und FVC2006). Das heißt, wenn Sie also weder Fingerabdruckleser noch Fingerabdruckbilder zur Hand haben, können Sie die Muster-Fingerabdrücke aus den Beispielsdatenbanken FVC2000 [2], FVC2002 [3] und FVC2004 [4] herunterladen. Jede Datenbank ist 110 Finger breit und acht Abdrücke pro Finger tief. Insgesamt stehen so 880 Fingerabdrücke zur Verfügung.

Alternativ können Nutzer im FPR-Framework auch ihre eigenen nutzerdefinierten Auswertungsprotokolle implementieren und mit den eigenen Datenbanken arbeiten. Sie können aber auch die NIST-Spezialdatenbank [5] und die Beispieldatenbank für Fingerabdrücke von Neurotech [6] verwenden. In Bild 2 ist ein Ausschnitt der extrahierten Datenbank FVC2000 DB1_B zu sehen.

Download und Installation

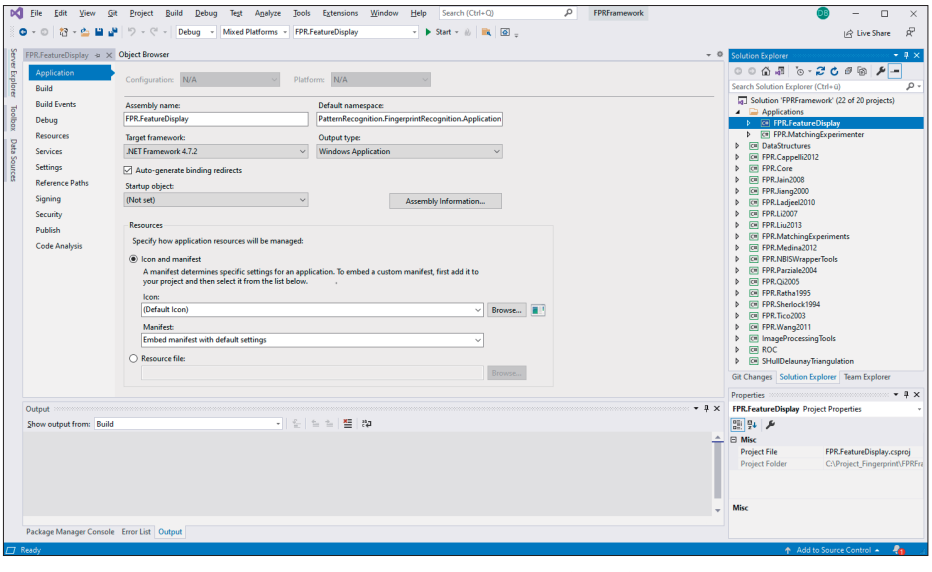
Für den Download des Frameworks [1] genügen als Systemanforderung .NET Framework 4 und Visual Studio ab Version 2015. Im Artikel wird Visual Studio 2019 in der Community Edition mit .NET Framework 4.7.2 verwendet.

Nachdem Sie das Framework heruntergeladen und die ZIP-Datei entpackt haben, können Sie die notwendigen Projekteinstellungen vornehmen.

Das *FPRFramework*-Projekt lässt sich über die Projektmappe *...\FPRFramework\FPRFramework.sln* in Visual Studio 2019 aufrufen. Da die Dateien für SourceAFIS (ein Open-Source-Algorithmus zum Identifizieren von Fingerabdrücken) beim Download nicht zur Verfügung stehen, müssen Sie das Projekt *FPR.SourceAFISWrapper* aus der *FPRFramework*-Projektmappe entfernen.

Die nächsten Schritte:

- Wechseln Sie in den Ordner *Applications* im Projektmappen-Explorer und markieren Sie das Projekt *FPR.FeatureDisplay*.
- Wählen Sie im Kontextmenü den Eintrag *Eigenschaften* aus und stellen Sie das Ziel-Framework auf *.NET Framework 4.7.2*. Bild 3 zeigt die Einstellung für die Projekteigenschaften.
- Bestätigen Sie die Änderung des Zielframeworks mit *Ja* und wiederholen Sie diesen Schritt auch für das Projekt *FPR.MatchingExperimenter*.
- Legen Sie jetzt das Projekt *FPR.FeatureDisplay* über das Kontextmenü als Startprojekt fest, indem Sie den Punkt *Als Startprojekt festlegen* auswählen.

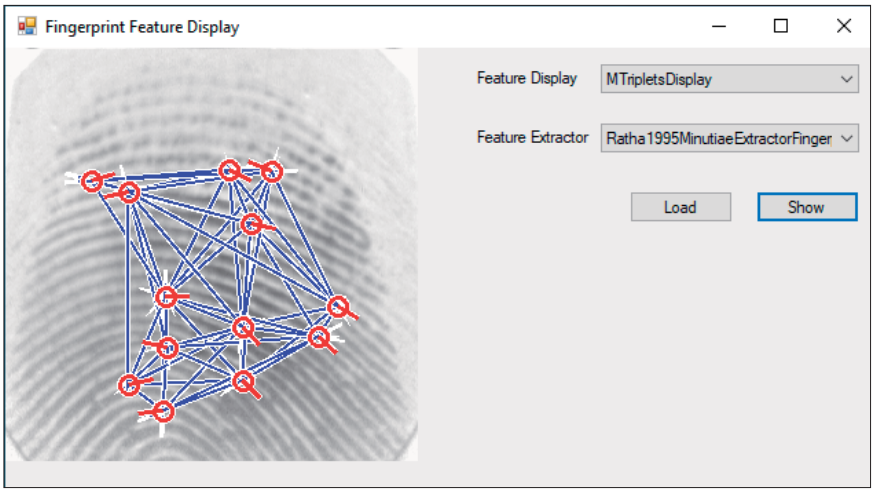


Die Einstellung des Beispielprojekts in Visual Studio 2019 (Bild 3)

Nach dem Build-Vorgang ist das FPR-Framework für eigene Forschungen einsatzbereit. Bild 4 zeigt die Anwendung Fingerprint Feature Display im Einsatz. ▶

Das Beispielprogramm

Als Beispielanwendung können Sie neben dem Fingerprint Feature Display auch das Projekt *FPR.MatchingExperimenter* ausführen (Bild 5). Über das Textfeld *Resources* legen Sie den Pfad zur gewünschten FVC-Datenbank fest. Achten Sie darauf, dass Sie im Kombinationsfeld *Experiment* die richtige Datenbank eingestellt haben. Über die Comboboxen *Minutia Extractor*, *Orientation Image Extractor* und *Skeleton Image Extractor* wählen Sie die Algorithmen aus, die verwendet werden sollen, um die grundlegenden Merkmale von Minutien, Orientierungsbild und Skelettbild zu berechnen.



Die Beispielanwendung Fingerprint Feature Display (Bild 4)

Das Kombinationslistenfeld *Matcher* wählt den Verifizierungsalgorithmus für den visuellen Abgleich des Fingerabdrucks aus. Das Feld *Feature Provider* bestimmt den Algorithmus, der die Merkmale für den ausgewählten Matcher speichert und abruft. Bild 6 zeigt die Anwendung in Aktion.

Das steckt drin

Mit dem Framework mit seinen beiden Applikationen lassen sich schon sehr gut Fingerabdrücke ermitteln. Bild 7 zeigt die komplette Projektstruktur des FPR-Frameworks in Visual Studio. Das wichtigste Ziel, das mit dem Framework verfolgt wird, ist die Möglichkeit, verschiedene Algorithmen zum Verifizieren von Fingerabdrücken zu prüfen und zu bewerten und zu einer Analyse der realen Bilder der Fingerabdrücke zu kommen.

Neben den Verifizierungsalgorithmen unterstützt das Framework auch Minutien-Matching-Algorithmen. Die Matcher, die für die Implementierung des LPIDBMinutiaeDescriptorsEvaluation-Verfahrens bereitstehen, sind [1]:

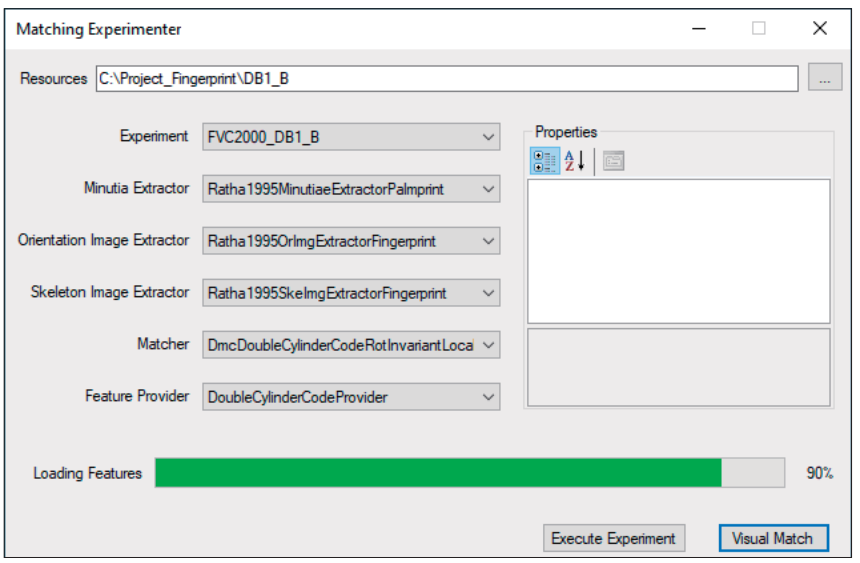
- DMCDoubleCylinderCodeRotInvariantLocalMatching (Cappelli et al., 2012)
- ILMDLocalMatcher (Laadjel et al., 2010)
- LiuDescriptorLocalMatcher (Liu et al., 2013)
- M3g11_5_LocalMatching (Medina-Pérez et al., 2012)
- MinutiaCodeLocalMatcher (Jain und Feng et al., 2009)
- MinutiaNPtLi2007LocalMatcher (Li et al., 2007)
- MinutiaNPletRao2016LocalMatcher (Rao et al., 2016)
- MinutiaNPletTan2009LocalMatcher (Rao et al., 2009)
- ModifiedRadialTriangulationLocalMatcher (Wang et al., 2012)

Die Beispiele aus der Datenbank mit den Applikationen Feature Display und Matching Experimenter demonstrieren, wie einfach das Framework zu bedienen ist und wie selbsterklärend der Nutzercode implementiert wurde. Die einfache und strukturierte Umsetzung im Framework erlaubt es darüber hinaus, beliebige Komponenten mit minimalem Aufwand zu ersetzen oder zu ändern.

Arbeitsweise

Die Applikation Fingerprint Feature Display ermöglicht es, Merkmalseigenschaften und Merkmalsextraktion für die Erzeugung eines Orientierungsbildes auszuwählen sowie einen Feature Extractor für die Codierung in die ausgewählte Vorlage festzulegen.

Die Individualität des Fingerabdrucks wird durch anatomische Merkmale und durch die gegenseitige Orientierung festgelegt. In der Praxis betrachtet man die Grate, die auf der Oberfläche zu erkennen sind, um die Fingerabdrücke zu unterscheiden. Die Unterscheidungsmerkmale lassen sich hierbei wie folgt einteilen:



Der Matching Experimenter im Einsatz (Bild 5)

- Singularitäten
- Kern
- Minutien

Unter Singularitäten versteht man die globale Struktur eines Abdrucks, die durch die Ausrichtung der Papillarleisten festgelegt ist. Das heißt, es handelt sich um Merkmale, die durch die umliegenden Grate eines Punktes bestimmt sind. Diese spezifischen Anordnungen werden dabei folgendermaßen beschrieben:

- Loop (Schleife): Klassifiziert das halbkreisartige Umlaufen eines Punktes.
- Whorl (Wirbel): Der Fingerabdruck weist ein strudelartiges Muster auf.
- Delta (Gabelung): Beschreibt ein dreieckförmiges Zusammenlaufen der Grate.

Als Kern (englisch: Core) bezeichnet man eine Singularität in der Nähe des Zentrums eines Fingerabdruckbildes, an deren Anordnung sich die Grate des gesamten Fingerabdrucks ausrichten.

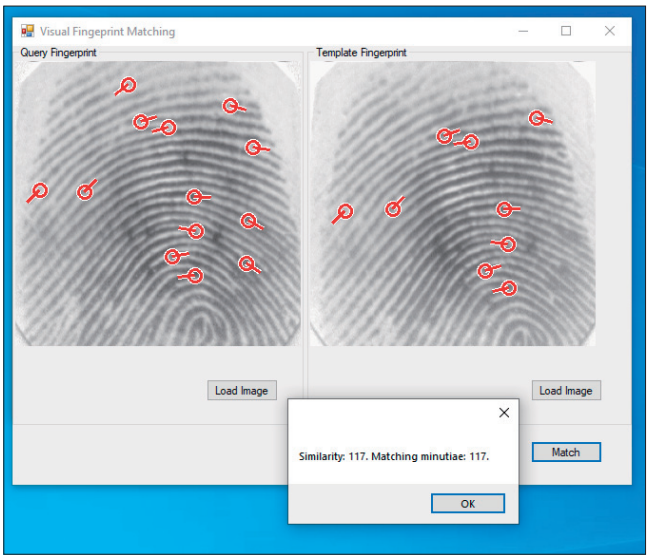
Die Minutien (von lateinisch minuzien für Kleinigkeiten) sind die charakteristischen Merkmale, die durch die Beschaffenheit oder Anordnung von Graten einmalig sind, die feinen Endungen und Verzweigungen der Fingerlinien. Insgesamt sind mehr als 150 unterschiedliche Minutien bekannt. Für die Authentifizierung eines Fingerabdrucks werden mehrere Minutien mit Referenzdaten verglichen. Daher lassen sich in der Praxis sehr einfach verschiedene Merkmale des Fingerabdruckes unterscheiden. Wie zum Beispiel:

- Grundmuster
- Grobe Merkmale, Schleifen, Bögen und Windungen
- Feinere Merkmale in Form von Minutien
- Porenstruktur

Ein System für das Erkennen von Fingerabdrücken erlaubt es, die Aufgaben grob in folgende Schritte aufzuteilen:

- Einlesen des Fingerabdrucks
- Bildverbesserung mit Filtern
- Skelettieren (Thinning) der Linien. Das heißt, die Linien werden auf ein Pixel Breite reduziert.
- Klassifikation des Inputs
- Lokalisieren von Linienenden und Gabelungen, also eine Detektion der Minutien
- Abgleichen des Fingerabdrucks mit gespeicherten Abdrücken

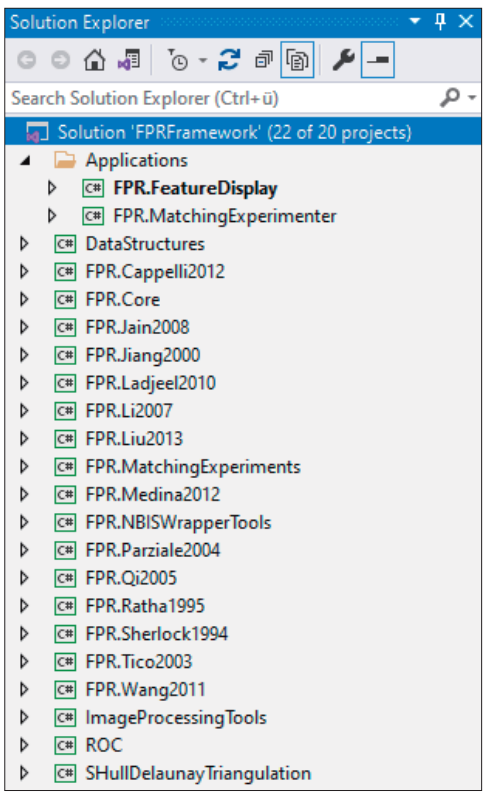
Die wichtige Aufgabe ist der Versuch, die Minutien zu erkennen und damit den richtigen Fingerabdruck in der Datenbank zu finden. Hier unterscheidet man zwischen zwei Stufen: der Minutien-Extraktion und dem Minutien-Matching.



Die Analyse des Fingerprint Matching in der Anwendung (Bild 6)

Beim Extrahieren der Minutien wird zuerst die lokale Orientierung geschätzt, mit deren Hilfe die Linien gefunden werden. Das ermöglicht es, die Minutien sehr einfach erkennen zu können. Es genügen in vielen Fällen bereits 14 Minutien zur sicheren Identifikation.

Beim Minutien-Matching versucht ein Algorithmus, die Ausrichtung der Minutien zu schätzen. Danach werden die Minutien vom Eingangsbild mit denen der Datenbank verglichen. Hier entscheiden in den meisten Fällen schon vier Kriterien über den Grad der Übereinstimmung zwischen Muster und Vorlage.



Die Projektstruktur von FPRFramework (Bild 7)

Für die Verwendung des Feature Extractors im FPR-Framework müssen die x- und y-Koordinaten als Position und der Winkel für jede Minutie verfügbar sein. Der Ursprung des Koordinatensystems befindet sich in der linken oberen Ecke des Bildes.

Durch diese Vorgabe erhöhen sich die Werte von x von links nach rechts und die Werte von y von oben nach unten.

Die Ausrichtung der Minutien erfolgt in Grad und nimmt im Uhrzeigersinn zu. Die Justierung eines Gratendes wird durch den Winkel zwischen der horizontalen Achse und der Linie dargestellt, die am Ende des Grats beginnt und durch die Mitte des Grats verläuft.

Die Orientierung einer Gabelung ergibt sich durch den Winkel zwischen der horizontalen Achse und der Linie, die am Minutienpunkt ▶

beginnt und durch die Mitte des Tals zwischen den Gabelungsrücken verläuft.

Das Framework erweitern

Um das FPR-Framework um weitere Feature Extractors zu erweitern, können Sie diese einfach per Reflexion dynamisch in das FPR-Framework laden.

Um einen neuen Feature Extractor hinzuzufügen, erben Sie von der generischen Klasse *FeatureExtractor<T>* und implementieren die Methode *ExtractFeatures(Bitmap image)*. Die Klasse können Sie wie dargestellt implementieren:

```
public class DemoFeatureExtractor
    : FeatureExtractor<DemoFeature>
{
  public override DemoFeature ExtractFeatures(
    Bitmap image)
  {
    // Code zum Extrahieren von Merkmalen
  }
}
```

Des Weiteren müssen Sie für jede von *FeatureExtractor* abgeleitete Klasse einen sogenannten Ressourcen-Provider erstellen. Ein solcher Ressourcen-Provider ermöglicht das Laden aus und Speichern in eine Ressourcendatei, die den Fingerabdrücken zugeordnet ist. Im FPR-Framework selbst finden Sie passende Ressourcen-Provider für die Extraktion von Minutien (*MinutiaListProvider*), Orientierungsbild (*OrientationImageProvider*) und Skelettbilder (*SkeletonImageProvider*). Die Implementierung kann dann wie bei dem Beispiel in Listing 1 erfolgen.

Weitere Algorithmen einbinden

Das Einbinden eines neuen Verifikationsalgorithmus für Fingerabdrücke lässt sich mit dem FPR-Framework genauso schnell bewerkstelligen wie das Erstellen eines neuen Feature Extractors, da auch in diesem Fall Reflexion verwendet wird, um alle Algorithmen zur Ausführungszeit dynamisch zu laden.

Durch dieses Vorgehen können Sie jetzt zum Beispiel den Open-Source-Algorithmus mit der Bezeichnung SourceAFIS von Robert Važan (und Mitwirkende) implementieren. Die Quellen für das Open-Source-Projekt können Sie von GitHub herunterladen [7]. Entpacken Sie den Quellcode, kompilieren Sie den C#-Code der Projektmappe *SourceAFIS.sln* und kopieren Sie die erzeugte Datei *SourceAFIS.dll* in den */bin/Release/*-Ordner des FPR-Frameworks.

SourceAFIS für .NET ist eine reine C#-Implementierung des Algorithmus. Der Template-Konstruktor für das Laden eines Bildes lässt sich sehr einfach über das folgende Beispiel implementieren:

Listing 1: Einsatz des Ressourcen-Providers

```
public class DemoFeatureProvider :
    ResourceProvider<DeomFeature>
{
  public MinutiaListProvider minutiaListProvider
  { get; set; }

  public OrientationImageProvider imageProvider
  { get; set; }

  public override string GetSignature()
  {
    return "demo";
  }

  public override bool IsResourcePersistent()
  {
    return true;
  }

  protected override DemoFeature Extract(string
    fingerprint, ResourceRepository repository)
  {
    private DemoFeatureExtractor featureExtractor
      = new DemoFeatureExtractor();
    try
    {
      var mlp = minutiaListProvider.GetResource(
        fingerprint, repository);
      var ip = imageProvider.GetResource(
        fingerprint, repository);
      return featureExtractor.ExtractFeatures(
        mlp, ip);
    }
    catch (Exception e)
    {
      if (minutiaListProvider == null)
        throw new InvalidOperationException
          ("DemoFeature kann nicht extrahiert "
          + "werden: Nicht zugewiesener List "
          + "Provider!", e);
      if (imageProvider == null)
        throw new InvalidOperationException
          ("DemoFeature kann nicht extrahiert "
          + "werden: Nicht zugewiesener Image"
          + " Provider!", e);
      throw;
    }
  }
}
```

Listing 2: Der SourceAFIS-Ressourcen-Provider

```
public class SourceAFISFeatureProvider :
    ResourceProvider<Person>
{
  private static AfisEngine Afis = new AfisEngine();

  protected override Person Extract(string
      fingerprint, ResourceRepository repository)
  {
    Fingerprint fp = new Fingerprint();
    fp.AsBitmap = imageProvider.GetResource(
      fingerprint, repository);
    Person person = new Person();
    person.Fingerprints.Add(fp);
    Afis.Extract(person);
    return person;
  }

  public override string GetSignature()
  {
    return string.Format("sAFIS");
  }

  public override bool IsResourcePersistent()
  {
    return true;
  }
}
```

```
var options = new FingerprintImageOptions { Dpi = 500 };
var probe = new FingerprintTemplate(
  new FingerprintImage(332, 533, Files.ReadAllBytes(
  "probe.dat"), options));
var candidate = new FingerprintTemplate(
  new FingerprintImage(320, 407, Files.ReadAllBytes(
  "candidate.dat"), options));
double score =
  new FingerprintMatcher(probe).Match(candidate);
bool matches = score >= 40
```

Der Konstruktor akzeptiert ein Graustufenbild zusammen mit der Bildbreite und -höhe. Die Decodierung von Standard-Bildformaten wie PNG und JPEG wird zurzeit nicht unterstützt. Der Fingerprint Matcher wird in dieser Implementierung nur dem Konstruktor übergeben. Die Variable *score* enthält den Ähnlichkeitswert der Fingerabdrücke.

Der Schwellenwert (*matches*) wird mit >= *40* festgelegt, da ab diesem Wert eine Übereinstimmung äußerst wahrscheinlich ist. Die Messung bei der Fingerabdruckerkennung wird durch die falsche Übereinstimmungsrate, kurz FMR (englisch für „false match rate"), gemessen.

FMR ist somit die Häufigkeit, mit der das System nicht übereinstimmende Fingerabdrücke fälschlicherweise als übereinstimmend erkennt. Der angegebene Schwellenwert von 40 entspricht einer FMR von 0,01 Prozent.

Die Implementierung in das FPR-Framework ist in Listing 2 zu sehen. Auch hier benötigt man dank Reflexion nur einen entsprechenden Ressourcen-Provider. Fertig ist der Fingerabdruck-Matching-Algorithmus für den SourceAFIS-Algorithmus.

Fazit

Dieser kleine Workshop sollte Sie in erster Linie animieren, eigene Algorithmen zum Erkennen von Fingerabdrücken zu entwickeln oder diese zu analysieren – und sei es nur zu Lernzwecken. Das vorgestellte FPR-Framework eignet sich sehr gut sowohl für die Forschung und Lehre wie auch für Ausbildung und Studium. Sie können über dieses Framework auch den Einsatz von Reflexion in C# nachvollziehen. Sie können zudem mithilfe des SourceAFIS-Algorithmus die Verwendung eines Algorithmus für die k-nächsten Nachbarn (k nearest neighbour) analysieren, dessen Leistung mit einer Hash-Tabelle vergleichbar ist.

Dieser Workshop kann natürlich nicht alle Besonderheiten oder speziellen Implementierungen berücksichtigen, er konnte lediglich versuchen, Appetit auf das Entwerfen und Entwickeln von Algorithmen mit Fingerabdrücken zu machen. Dazu wünsche ich viel Spaß und Erfolg! ■

[1] FPR-Framework, www.dotnetpro.de/SL2208FingerPrint1
[2] FVC-2000-Datenbank, www.dotnetpro.de/SL2208FingerPrint2
[3] FVC-2002-Datenbank, www.dotnetpro.de/SL2208FingerPrint3
[4] FVC-2004-Datenbank, www.dotnetpro.de/SL2208FingerPrint4
[5] NIST: Special Database Catalog, www.dotnetpro.de/SL2208FingerPrint5
[6] Neurotechnology: Downloads, www.neurotechnology.com/download.html
[7] SourceAFIS fingerprint matcher, https://sourceafis.machinezoo.com

Daniel Basler
arbeitet als Lead Developer und Softwarearchitekt. Seine Schwerpunkte liegen auf Cross-Platform-Apps, Android, JavaScript und Microsoft-Technologien.

dnpCode A2208FingerPrint