Milestone 1

PG-CSI

Paderborn Center for Parallel Computing
Universität Paderborn

29. November 2007

Allgemeine Ziele der PG-CSI

- Entwicklung eines Matching System für Biometrische Merkmale.
- Mit einer hohen Erkennungsgenauigkeit ...
- .. und einer state-of-the-art Performance.
- Zum erreichen dieser Ziele sollen ...
  - Arminius, der Parallelrechner der Universität
  - und FPGAs

  ... um die Nutzen dieser Technologie für die Biometrie untersuchen zu können, zum Einsatz kommen.

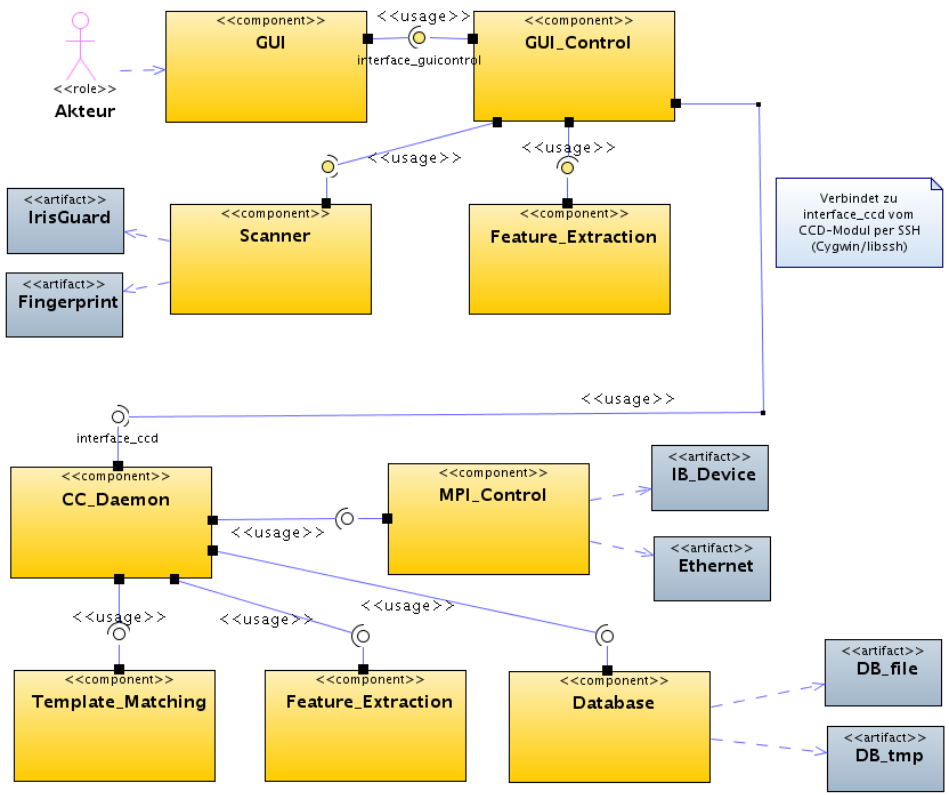
Ziele des ersten Milestones : Prototyping (soll Zustand)

- Algorithmen Prototypen zur Evaluierung.
- Feature-Extraction Algorithmen sind implementiert.
- Matching Engine ist im Grobentwurf fertig.
- Clusterparallelisierung ist implementiert.
- Architektur:
  - Teile arbeiten ansatzweise zusammen
  - Aufrufe zumindest manuell möglich

Agenda dieses Vortrages

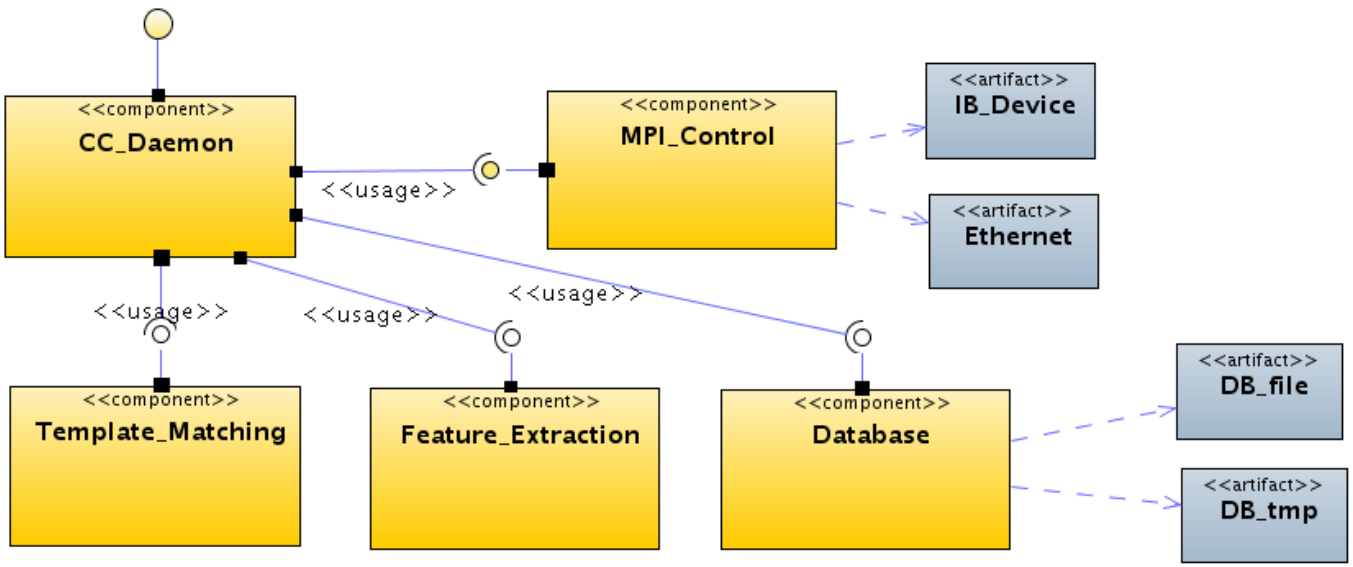
Die Komponenten

Gesamtübersicht zu den Komponenten

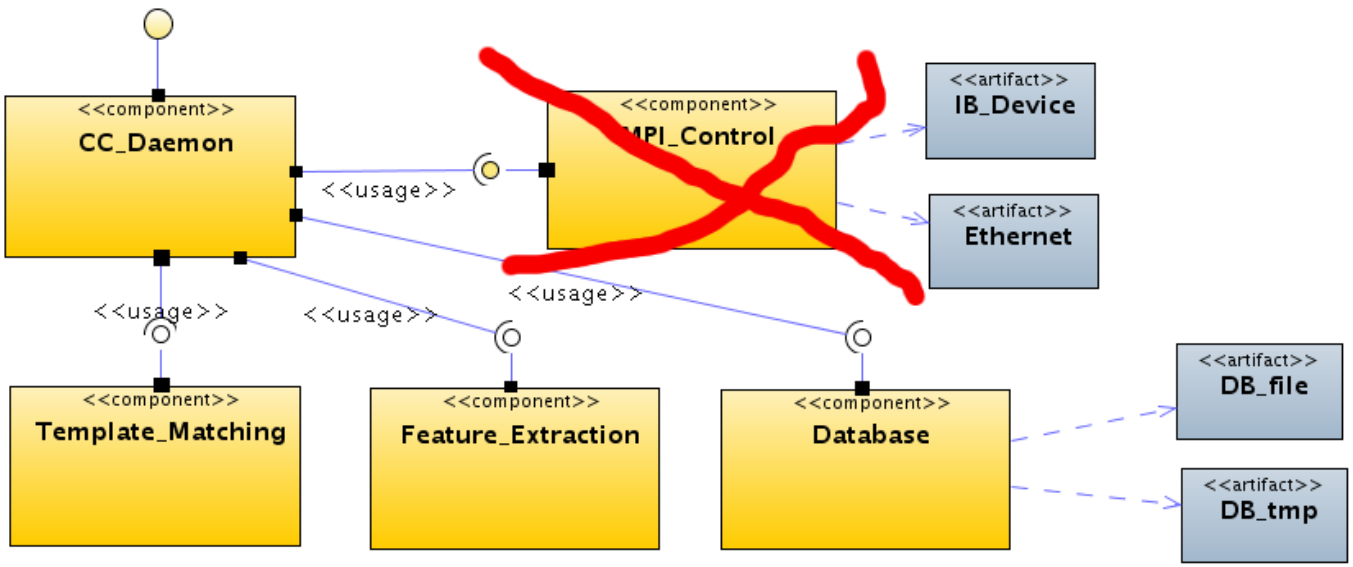


Cluster

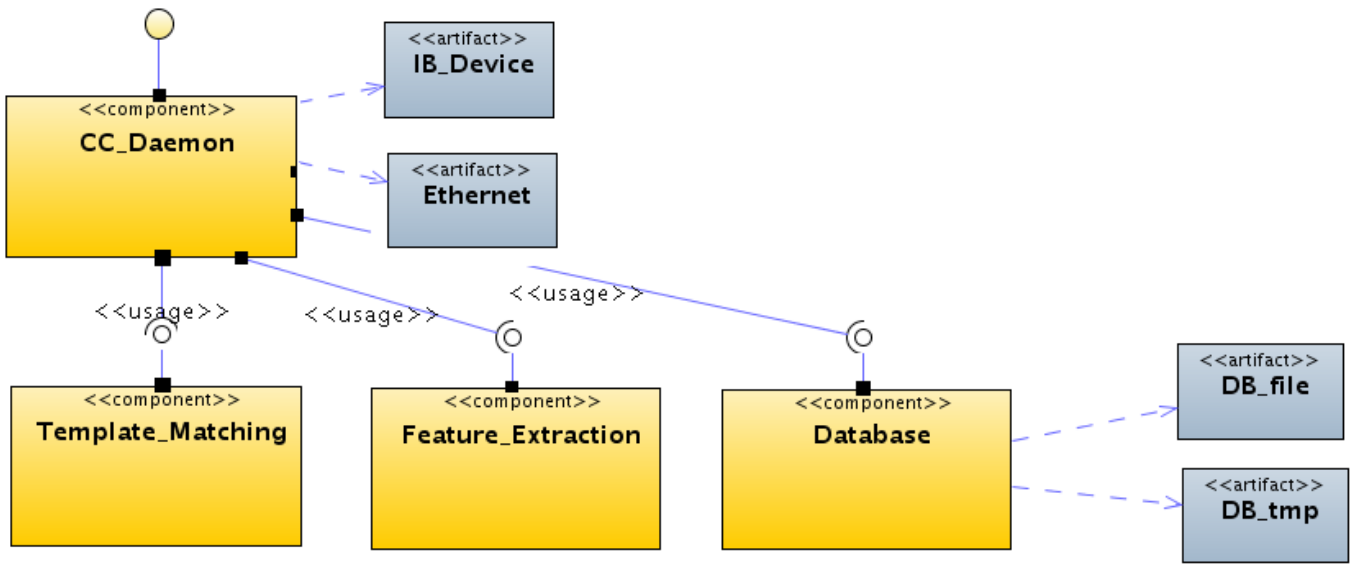
Übersicht der Cluster-Software



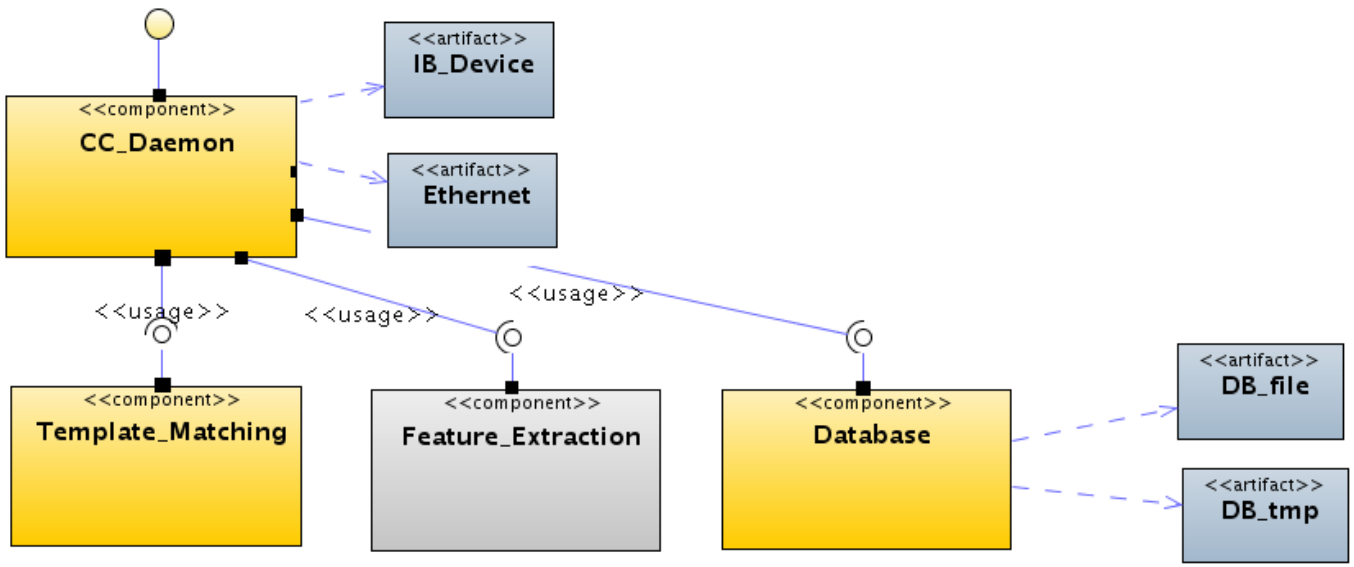
Übersicht der Cluster-Software



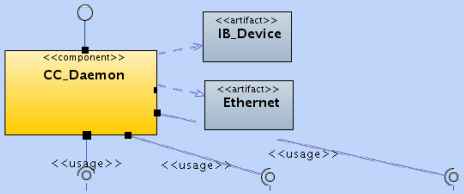
Übersicht der Cluster-Software



Übersicht der Cluster-Software



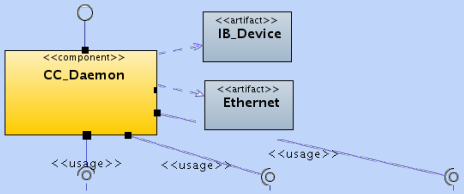
Cluster Controller - CCD (Christoph Konersmann)



Aufgaben

- Stellt Kommunikationsschnittstelle zur Außenwelt zur Verfügung
- Koordiniert und Überträgt Jobs an Knoten
- Verwaltet Knoten-Konfigurationen
- Verwaltet Log-Output
- Koordiniert MPI-Kommunikation

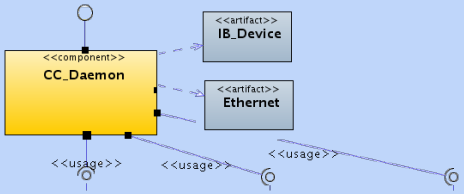
Cluster Controller - CCD (Christoph Konersmann)



Bereits implementiert

- Schnittstelle und Struktur zur Frontend-Cluster-Kommunikation
- Konfiguration über Kommandozeile und Konfigurationsdatei
- MPI-Methoden zur Jobverteilung und Ergebnissammlung
- Option zum Benchmarking der Jobs
- Matching-Module eingefügt: Minutiae, Clustering
- Datenbank-Modul eingefügt

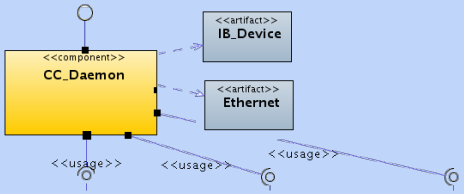
Cluster Controller - CCD (Christoph Konersmann)



Werdegang / Probleme

- Kommunikation Fronten-Cluster:
  - Versuch, „Json“ als Protokoll zu implementieren
  - Versuch, „XML-RPC“ als Ersatz zu „Json“ zu implementieren
  - Statt „Json“ und „XML-RPC“ nun Kommunikation durch einfaches Netzwerksocket und systemunabhängige Struktur
  - Berechnen von Strukturgrößen und Verschicken von Binärdaten zwischen Windows/Linux und 32bit/64bit nicht trivial!

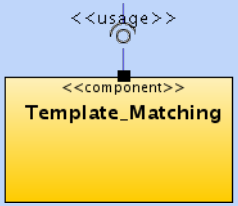
Cluster Controller - CCD (Christoph Konersmann)



Weitere Planung

- Integration von Methoden zur Programmierung und Nutzung von FPGA-Systemen.
- Bei positivem Matching vorzeitiges Beenden der Clusterberechnungen.
- Implementierung von nicht blockierenden MPI-Methoden.

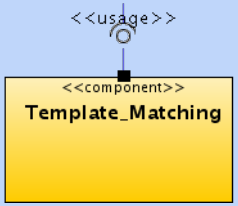
Matching - Iris (Christoph Scholz, Nils Timm)



Aufgaben

- Matching der Iriscodes durch Berechnung der Hamming-Distanz
- Ausschluss verdeckter Irisbereiche durch Verwendung von Maskenbits

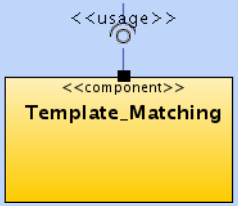
Matching - Iris (Christoph Scholz, Nils Timm)



Bereits implementiert

- Software-Matcher zum 1:1 Vergleich von Iriscodes
- Vergleich der Iriscode-Paare in 8 verschiedenen Rotationen

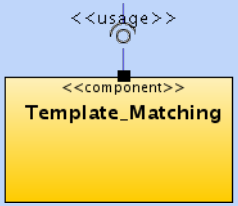
Matching - Iris (Christoph Scholz, Nils Timm)



Werdegang / Probleme

- Rotationsinvarianz noch nicht optimal, geeigneter Drehungswinkel muss noch bestimmt werden

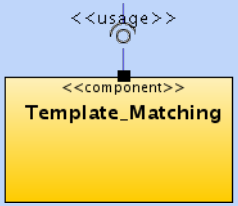
Matching - Iris (Christoph Scholz, Nils Timm)



Weitere Planung

- Festlegung der Thresholds für 1:1 und 1:N Vergleiche
- Integration der Maskenbits
- Umsetzung des Matchings auf FPGA's

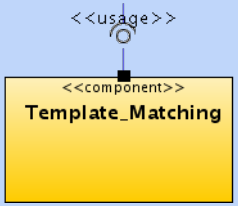
Matching - Finger/Minutien (Elmar Weber)



Aufgaben

- Implementierung des Fingerprintmatchings über Minutien

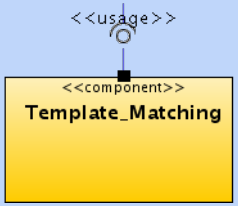
Matching - Finger/Minutien (Elmar Weber)



Bereits implementiert

- Lauffähige Implementierung des Algorithmus' in C
- Komplette Abdeckung durch Unit Tests
- Simulierte Version des Algorithmus in VHDL

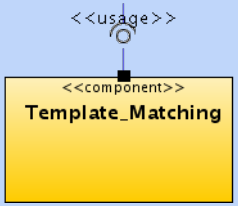
Matching - Finger/Minutien (Elmar Weber)



Werdegang / Probleme

- Konkrete Toleranzwerte des Papers unbekannt → versucht gute Toleranzwerte zu finden anhand der Standardauflösung
- Aufgrund fehlender Testdaten keine Tests möglich → Generator für Templates geschrieben, der basierend auf einem Pseudozufallszahlengenerator reproduzierbare, aber zufällige Templates generiert (Daten für andere Templates werden mittlerweile auch mitgeneriert)
- FPGA Implementierung: Anbindung per DMA an den Host

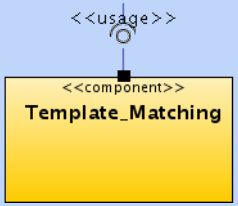
Matching - Finger/Minutien (Elmar Weber)

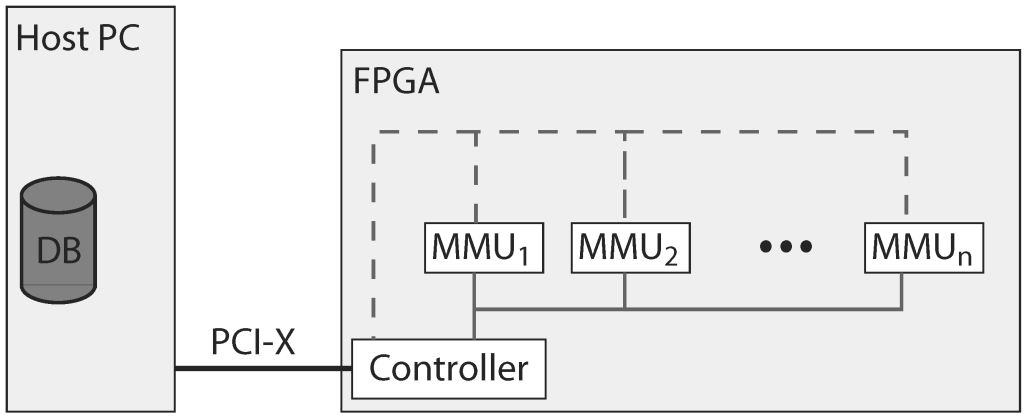


Weitere Planung

- Parameter Prototyping des Algorithmus durch genetischen Algorithmus (mit Hilfe des Clusters und evtl. FPGA)
- Anbindung der FPGA Version
- Optimierung der Software Version

Matching - Finger/Minutien (Elmar Weber)



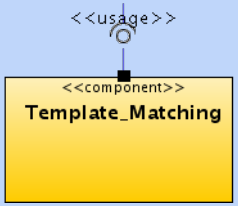


MMU — Minutia Matching Unit

– – – Global OR bus: asserts if any MMU matches

——— Data bus: programs the MMUs, forwards minutiae data

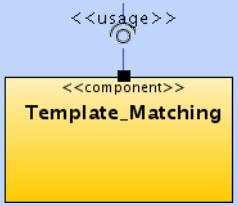
Matching - Finger/Clustering (Robert Meiche)



Aufgaben

- Klassifizierung der Daten um die Vergleiche zu reduzieren
  - Offline database clustering:
    - Daten werden Clustern zugeordnet
    - Cluster werden widerum in Gruppen eingeteilt

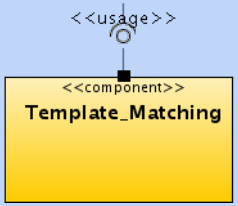
Matching - Finger/Clustering (Robert Meiche)



Aufgaben

- Klassifizierung der Daten um die Vergleiche zu reduzieren
  - Online query processing (Suche) :
    - Phase1: Abgleichen des queryprints mit den Repräsentanten der Cluster
    - Phase2: Abgleichen des queryprints mit den Repräsentanten der Gruppen der gefundenen Cluster
    - Phase3: Abschließende Suche innerhalb der gefundenen Gruppen

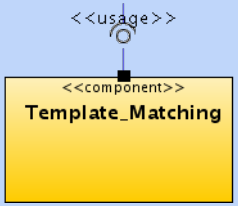
Matching - Finger/Clustering (Robert Meiche)



Bereits implementiert

- Offline Database-Clustering läuft zufriedenstellend mit der Testdatenbank
  - Ermittlung der Repräsentanten und Aufteilung der Daten in die entsprechenden Cluster funktioniert sehr gut
- Online query processing ist implementiert
  - Die erste Phase läuft sehr effektiv. Der abgleich mit den Repräsentanten der Cluster bei den Testdaten ergab, dass der entsprechende Cluster immer als erstes Element der Ergebnisliste zurückgegeben wurde.

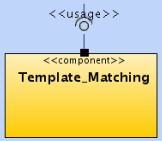
Matching - Finger/Clustering (Robert Meiche)



Werdegang / Probleme

- Die zweite Phase der Suche läuft noch nicht. Es werden nur Gruppen zurückgegeben, in denen der Suchabdruck nicht enthalten ist → dritte Phase konnte noch nicht richtig getestet werden.

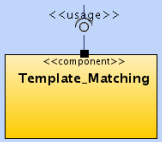
Matching - Finger/Clustering (Robert Meiche)



Weitere Planung

- Die Unterteilung der Cluster in Gruppen muss eventuell abgeändert werden
- Evtl. andere Verteilungsfunktion beim Verteilen der Daten auf die Gruppen der einzelnen Cluster
- Feineinstellung der entsprechenden Suchparameter
  - Evtl. Alternativlösung, falls die zweite Suchphase nicht funktionieren sollte
    $\rightarrow$ Kombinieren mit der Minutien-Suche :
    Mit Hilfe der Clusterrepräsentanten die Ergebnismenge reduzieren und danach die Minutien-Suche anwenden

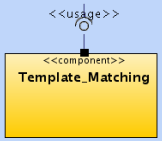
Matching - Finger/Filterbank (René C. Zorn, Rosbeh Etemadi)



Aufgaben

- Fingerprint Matching
  - Implementierung des filterbank-basierten Fingerprintmatchings über Feature Vectors
- Feature Extraction
  - Implementierung eines Verfahrens zur Erstellung von Filterbank-Templates
  - Implementierung eines Gabor Filters

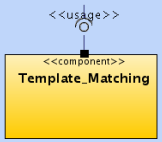
Matching - Finger/Filterbank (René C. Zorn, Rosbeh Etemadi)



Bereits implementiert

- Fingerprint Matching
  - Matching von Filterbank-Templates
- Feature Extraction
  - Erstellung von Filterbank-Templates aus Fingerabdruckbildern

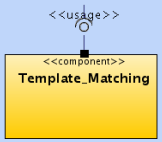
Matching - Finger/Filterbank (René C. Zorn, Rosbeh Etemadi)



Werdegang / Probleme

- Tests anhand von Bitmap-Dateien um die korrekte Erstellung von Feature Vertors zu prüfen
- Zuerst wurden Feature Vectors aus kartesischen Pixelkoordinaten berechnet
- Später dann Feature Vectors aus polaren Pixelkoordinaten
- Probleme gab es mit der eigenen Implementierung des Gabor Filters

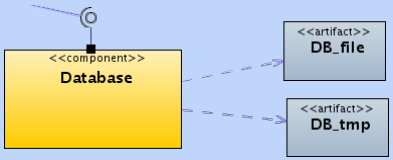
Matching - Finger/Filterbank (René C. Zorn, Rosbeh Etemadi)



Weitere Planung

- Funktionsfähige Implementierung des Gabor Filters (vorerst mit Hilfe externer API)
- Optimale Parameter für das Matching und den Gabor Filter bestimmen
- Laufzeitmessungungen durchführen und dokumentieren
- Umsetzung des Matching-Algorithmus auf FPGAs

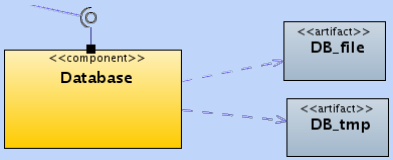
Database (Dominic Eschweiler)



Aufgaben

- Konsistentes Speichern der Templates
- Performantes Auslesen (Bootstrapping) unter optimalen Bedingungen für das Cluster-Dateisystem
- Unterstützung des Verteilungsprozesses
- Effiziente Benutzung des Speicherplatzes

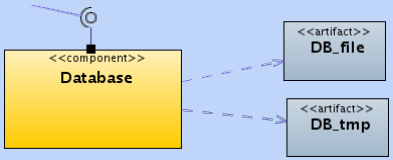
Database (Dominic Eschweiler)



Bereits implementiert

- Daten Ablegen und gepuffertes Lesen ist vollständig implementiert
- Abstraktionsschichten sorgen für den Erhalt der Konsistenz
- Um platzsparend zu arbeiten, werden die Templates enthaltenden Structs in mehrere große Dateien gestreamt.
- das csi_dbtool kann verwendet werden um :
  - Datenbanken neu anzulegen
  - Bestehende Datenbanken für eine andere Anzahl an Headnodes zu optimieren.

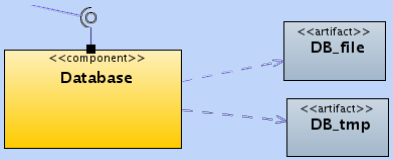
Database (Dominic Eschweiler)



Bereits implementiert

This is the CSI DB-Utility in Version : 1.0
-i input dir
-o output dir
-c <CREATE> <CONVERT>
-n headnodes
-t <ir> <fp> <db>
–help

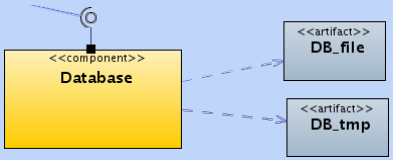
Database (Dominic Eschweiler)



Werdegang / Probleme

- Anfänglich lange Verzögerung wegen eines Buffer-Overflows, der sich aber an einer ganz anderen Stelle zeigte.
- Um die Software den immer neuen Anforderungen anzupassen, ist die API in vielen Schichten implementiert worden.
- Da aber die vielschichtige Implementierung mit einer hohen Anzahl an Routinen das Programmieren dieser sehr vereinfacht.

Database (Dominic Eschweiler)

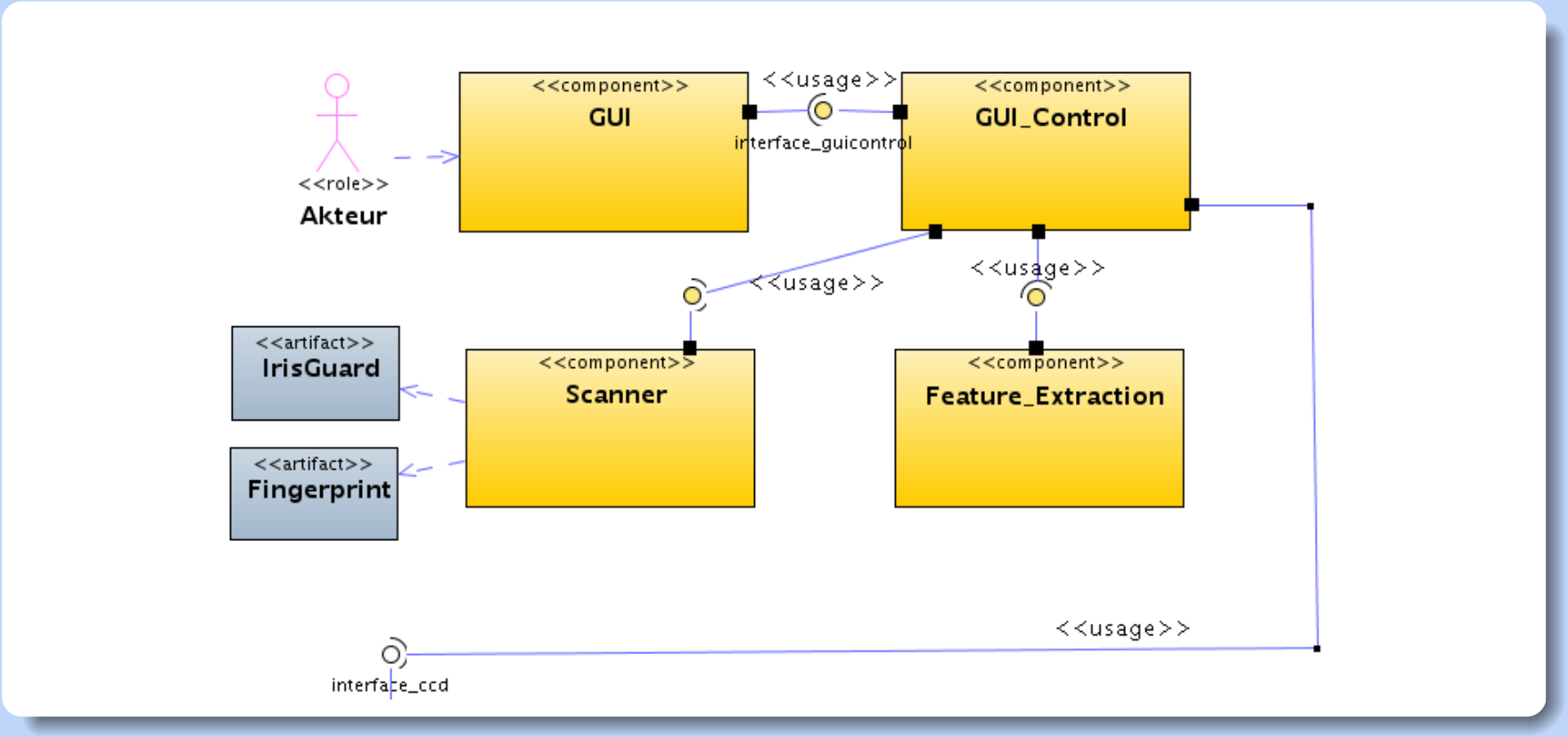


Weitere Planung

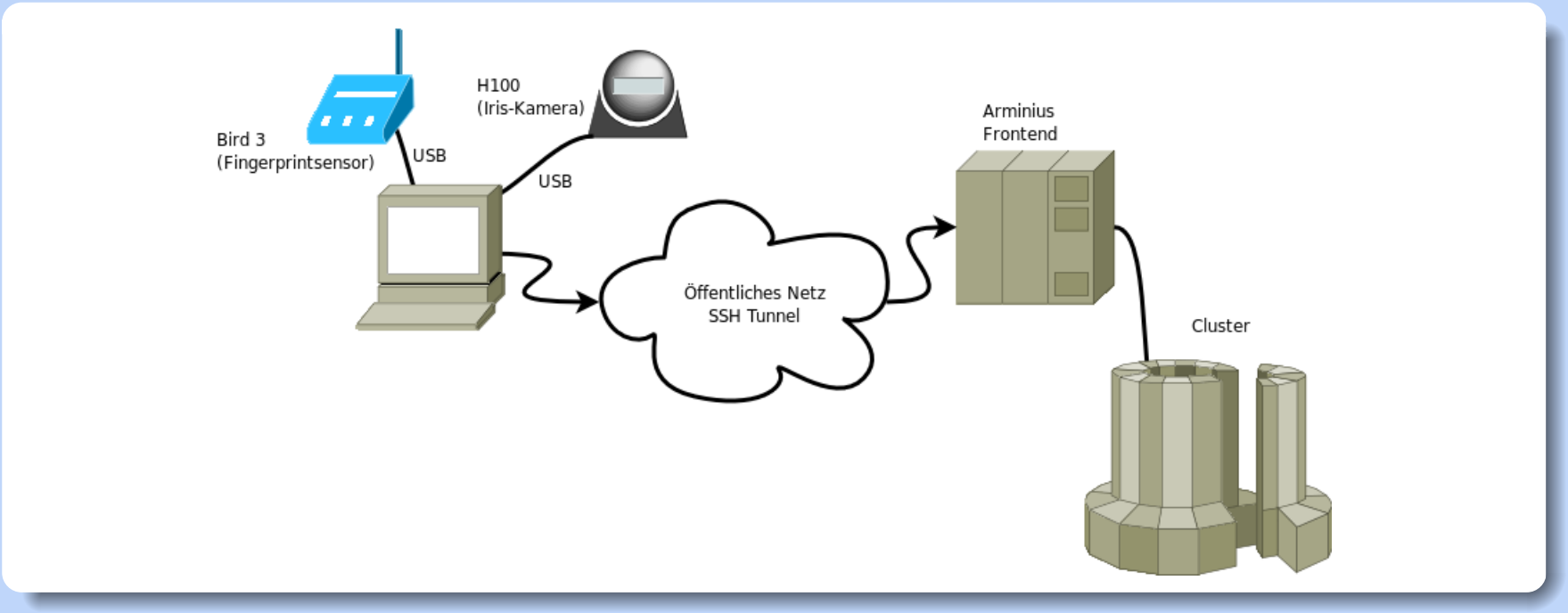
- Grundliegende Funktionen für ein Benchmark zum zweiten Milestone sind implementiert
- Erste Tests lassen vermuten das $\geq$ 50MB lesend erreicht werden können.
- Benchmark-Ergebnisse und das Fein-Tuning werden zum zweiten Milestone verfügbar sein.
- Dominic wechselt zur Unterstützung Pascals zur Feature-Extraction.

Frontend

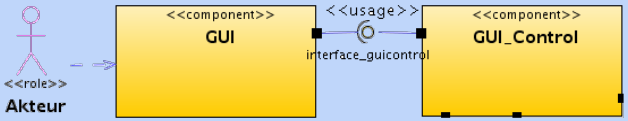
Übersicht des Frontends



Interconnect



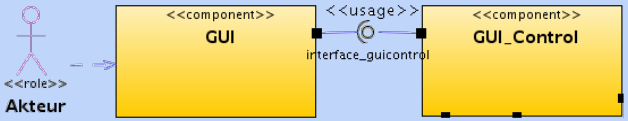
Gui / Gui-Controller (André Ückermann)



Aufgaben

- Erstellen einer grafischen Benutzeroberfläche nach Usability-Gesichtspunkten
- Einbinden aller Module auf Frontend-Ebene
- Kommunikationsschnittstelle zum Cluster

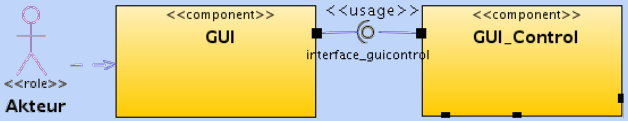
Gui / Gui-Controller (André Ückermann)

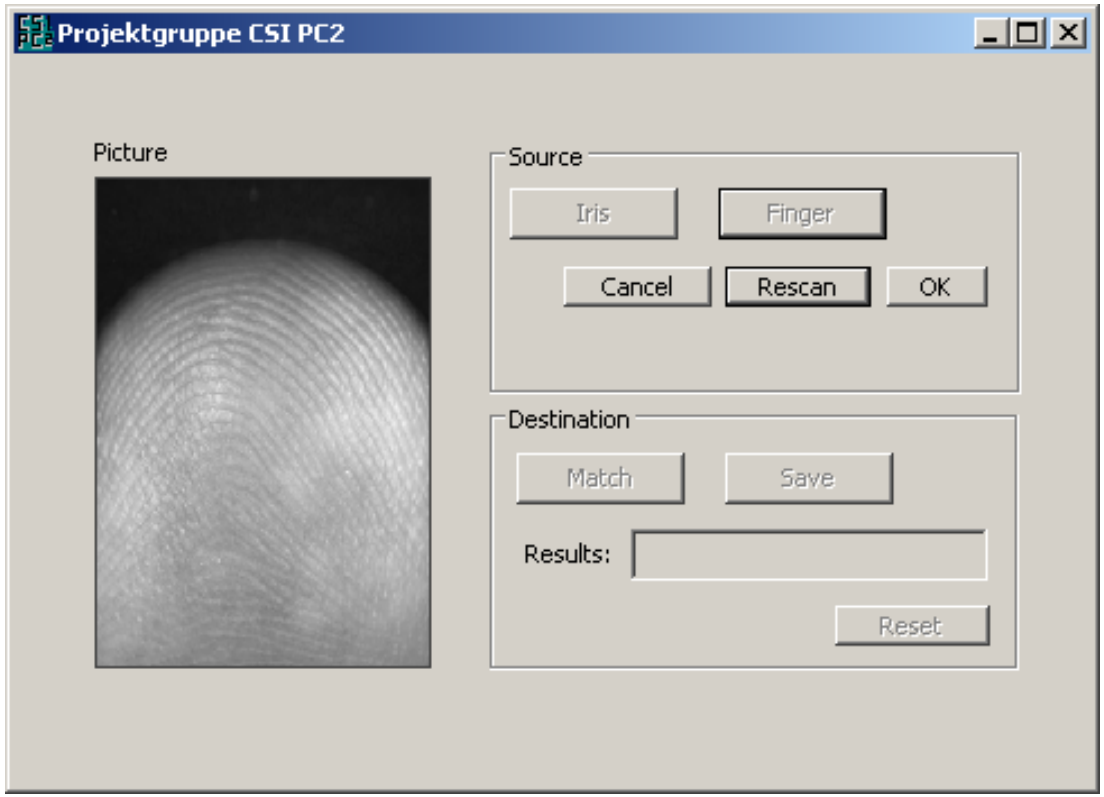


Bereits implementiert

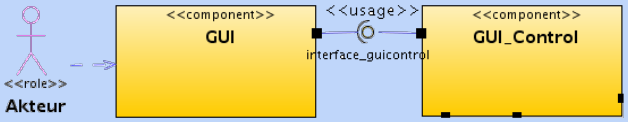
- Grafische Benutzeroberfläche
- Einbinden aller Module
- Kommunikation zum Cluster (Speichern und Matchen)

Gui / Gui-Controller (André Ückermann)





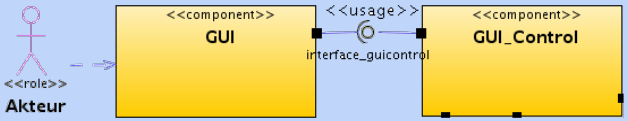
Gui / Gui-Controller (André Ückermann)



Werdegang / Probleme

- Kommunikation Frontend-Cluster
  - Implementierung mit Json und XML-RPC, nicht frei von Fehlern möglich
  - Stattdessen Implementierung mit einfachen Netzwerksockets
  - Probleme mit der Übertragung (Größe der ankommenden Daten)
- Einbinden der Module
  - Modulintegration wegen Verwendung verschiedener Hauptbibliotheken und Strukturunterschieden schwierig.
  - Teilweise Einbindung von Releases und Kommunikation über Dateien.

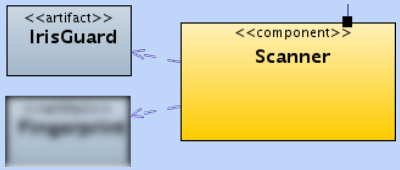
Gui / Gui-Controller (André Ückermann)



Weitere Planung

- Kleine Schönheitskorrekturen.
- Dynamisch einstellbare IP.

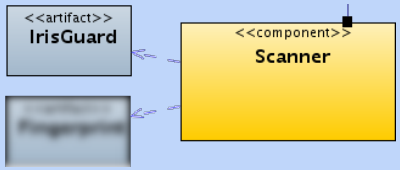
Scanner / Iris-Kamera (Klaus Herbold)



Technische Daten

- IrisGuard IG-H100
- Anbindung Frontend: USB 2.0
- Anbindung Kamera: Chinch mit monochromen NTSC-Composite Video Signal
- Video Grabber liefert Signal an Frontend
- Auflösung: 640 x 480
- Programmier-Schnittstelle: IFA-API
  $\rightarrow$ Programmiersprache: Visual C++

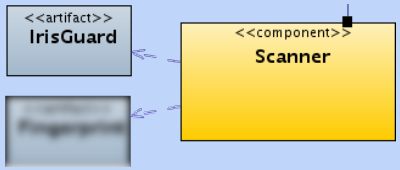
Scanner / Iris-Kamera (Klaus Herbold)



Aufgaben

- Steuerung der Kamera
- Anzeige eines Live-Bildes
- Bild speichern als JPG-/BMP-Datei

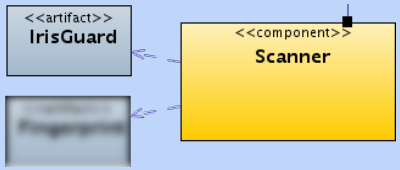
Scanner / Iris-Kamera (Klaus Herbold)



Bereits implementiert

- Zwei verschiedene Libraries:
  - DirectShow für die Vorschau
  - OpenCV zum Speichern

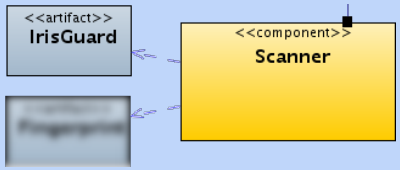
Scanner / Iris-Kamera (Klaus Herbold)



Werdegang / Probleme

- Kamera wurde erst spät geliefert
- IFA-API erst seit letzter Woche da
  - → vorläufige Implementierung noch mit anderen Libraries
- Qualität der Bilder (noch) nicht besonders gut
  - Abbildung blauer Iriden weitaus schlechter als von braunen Iriden
  - Scanlines deutlich sichtbar
  - etwas unscharf
  - → mit neuer API hoffentlich besser

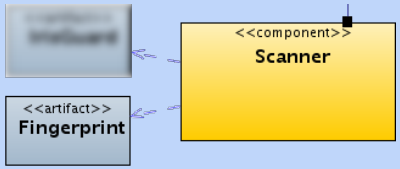
Scanner / Iris-Kamera (Klaus Herbold)



Weitere Planung

- Die selbe Funktionalität mit IFA-API implementieren
- Qualität der Bilder noch verbessern
- Aufnahmegeschwindigkeit erhöhen

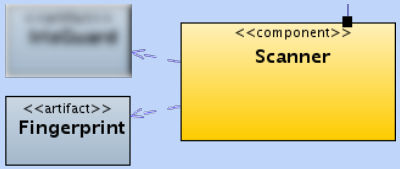
Scanner / Fingerprint-Sensor (Samira Brulic)



Technische Daten

- BiRD 3 Desktop USB
- Anbindung Frontend: USB 2.0
- Auflösung: 256 x 360
- CMOS Sensoroberfläche 16 x 19 mm
- Programmier-Schnittstelle: TST-API
  → Programmiersprache: Visual C++

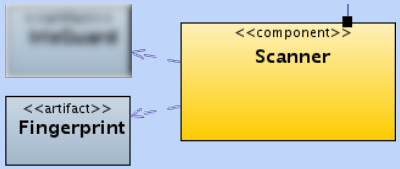
Scanner / Fingerprint-Sensor (Samira Brulic)



Aufgaben

- Steuerung des Scanners
- Asynchrone Bildaufnahme
- Bild speichern als JPG-/BMP-Datei

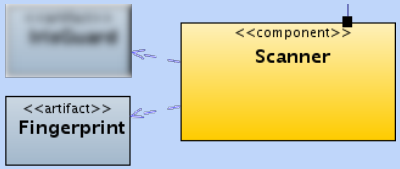
Scanner / Fingerprint-Sensor (Samira Brulic)



Bereits implementiert

- Asynchrone Bildaufzeichnung
- Bildspeicherung als JPG

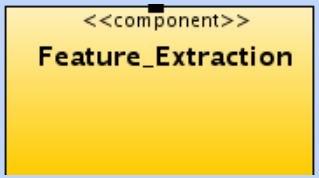
Scanner / Fingerprint-Sensor (Samira Brulic)



Werdegang / Probleme

- Wenn der Treiber im Hintergrund als Service läuft, lässt er sich nicht über eine DLL aufrufen - was aus der API nicht ersichlich war.
- Bilder haben geringe Auflösung und Qualität, was machmal zu Problemen bei der Feature-Extraction führt.

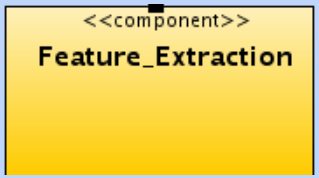
Feature-Extraction-Iris (Christoph Scholz, Nils Timm)



Aufgaben

- Lokalisieren der Iris in einem Bild
- Erzeugung einer normalisierten Darstellung
- Generierung des 2048 Bit Iriscodes und der zugehörigen Maskenbits

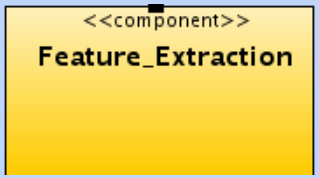
Feature-Extraction-Iris (Christoph Scholz, Nils Timm)



Bereits implementiert

- Auffinden der Pupille und der äußeren Irisränder
- Detektion verdeckter Irisbereiche als Grundlage für Maskenbits
- Transformation des Irisbildes in normalisierte Polarkoordinaten-Darstellung
- Berechnung des Iriscodes mittels Gabor-Wavelets

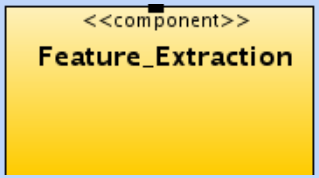
Feature-Extraction-Iris (Christoph Scholz, Nils Timm)



Werdegang / Probleme

- Anfänglich Performance-Probleme beim Lokalisieren der Pupille. Verbesserung der Performance durch Verkleinerung des Suchbereichs
- Bei der Anwendung des Gaborfilters zur Iriscode-Generierung lässt Daugman Parameter bezüglich Skalierung und Frequenz der Wavelets offen. Die Parameter, die einen optimalen Iriscode für das Matching erzeugen, müssen noch bestimmt werden

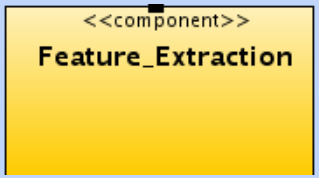
Feature-Extraction-Iris (Christoph Scholz, Nils Timm)



Weitere Planung

- Bestimmung der optimalen Parameter für den Gaborfilter mittels umfangreicher Tests auf dem Cluster
- Weitere Verbesserung der Performance durch Parallelisierung (SSE)

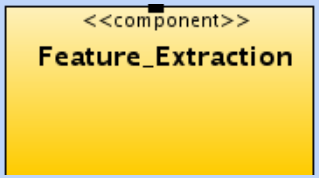
Feature-Extraction-Finger (Pascal Deppe, Samir Brulic)



Aufgaben

- Aufbereitung des Bildes
  - Normalisieren
  - Bestimmen des Orientierungsvektors, der Region Mask und der Rillenfrequenz
- Extrahieren der Merkmale Minutien, Gabor Feature Maps und Orientierungsfeld fürs Clustering

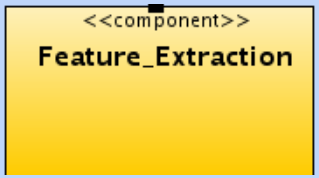
Feature-Extraction-Finger (Pascal Deppe, Samir Brulic)



Bereits implementiert

- Extrahieren der Merkmale (Minutien + Orientierungsfeld) mit mindtct (NIST)
- Berechnung des Core-Points eines Fingerabdrucks
- Aufbereiten der Ausgaben von mindtct zur Weitergabe an das Matching

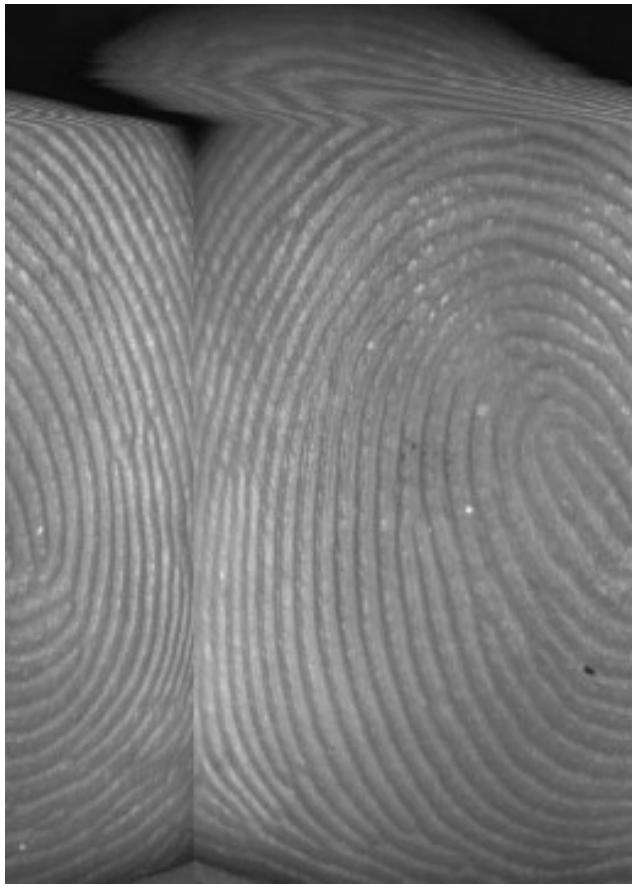
Feature-Extraction-Finger (Pascal Deppe, Samir Brulic)



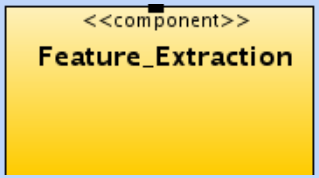
Werdegang / Probleme

- Versuch einer eigenen Implementation auf Basis verschiedener Paper und unterschiedlichen Grafik-Bibliotheken
  - $\rightarrow$ Jedes mal gescheitert, da die Angaben in den Papers oft sehr vage und ungenau.
- Einbindung der Sourcen vom NIST - Probleme:
  - Spezielles Eingabeformat (iHead-Datei)
  - NIST hat Probleme mit den Fingerabdrücken unseres Scanners, da die oberen Bildecken schwarz sind $\rightarrow$ Bild verzieht sich

Feature-Extraction-Finger (Pascal Deppe, Samir Brulic)



Feature-Extraction-Finger (Pascal Deppe, Samir Brulic)



Weitere Planung

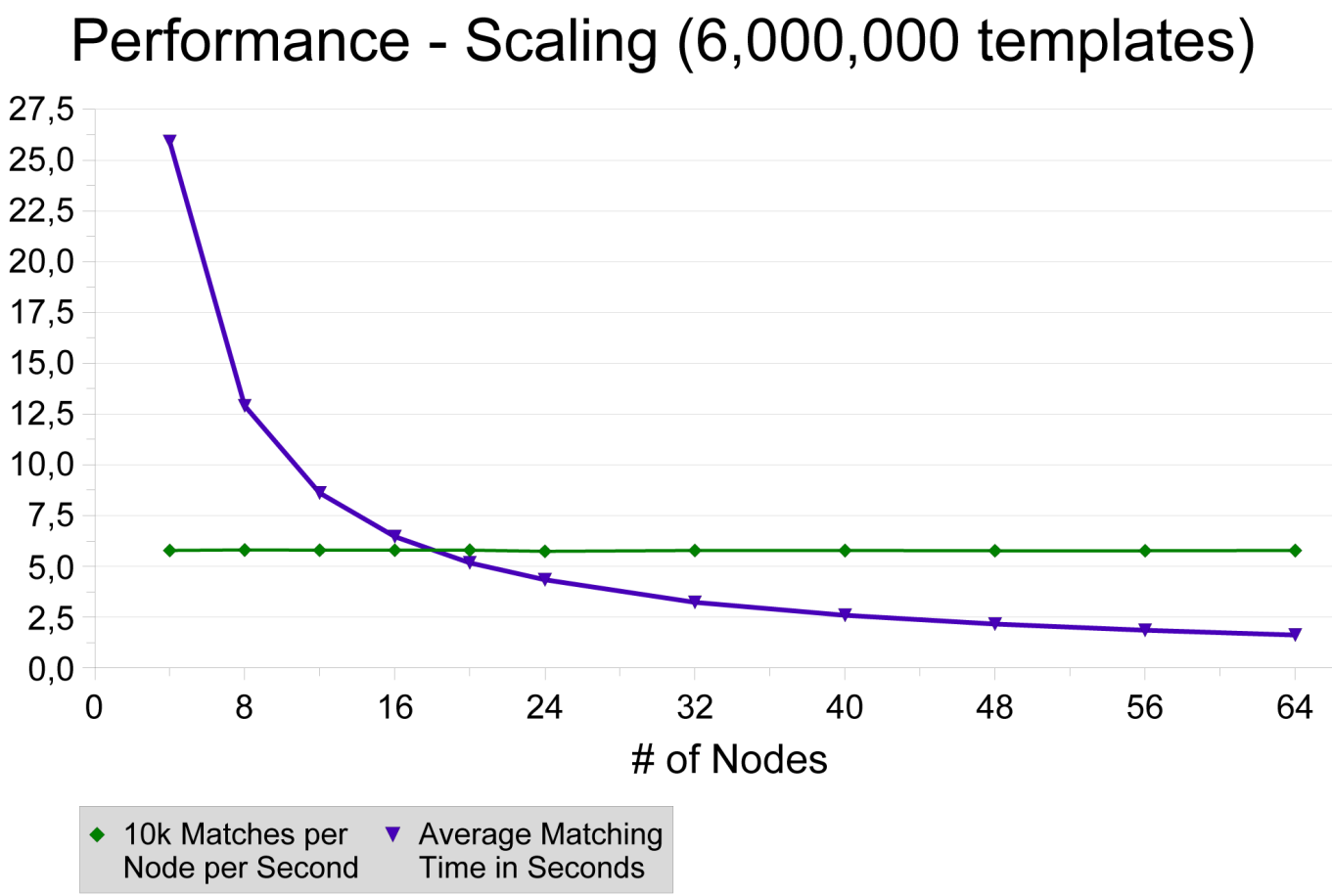
- Sondierung der Verfahren / Paper.
- Implementierung eines Prototypes in C für die Feature-Extraction auf dem Cluster, inklusive einer Optimierung durch MMX/SSE.
- Benchmarking und Audit um die Leistungsfähigkeit der Software-Komponenten zu überprüfen und die Teile zu finden, für die es sich eventuell lohnen würde, diese auf einen FPGA zu implementieren.

Erfolgsmeldungen und noch bestehende Probleme

Erfolgsmeldungen

- Wir konnten uns zu einem hohen Anteil an die Spezifikation halten → gute Planung
- Die Infrastruktur (CCD, GUI / GUI-Controller, Datenbasis) ist fertig implementiert und bietet eine solide Arbeitsgrundlage.
- Die Matching-Verfahren sind insgesamt nahezu oder komplett fertig. Es existiert sogar schon für ein Verfahren eine FPGA-Implementierung → hier sind wir weiter als der Zeitplan es vorsieht!
- Aufgetretene Unwägbarkeiten können aller Voraussicht gut kompensiert werden.

Erfolgsmeldungen



Noch bestehende Probleme

- Zugriff auf die Iris-Kamera noch etwas langsam → lange Lieferzeit, wahrscheinlich nur ein billiger CCD-Sensor verbaut und keine wirklich existierende API.
- Feature-Extraction ist momentan auf die Quellen von NIST angewiesen → unklare Papers, Unterbesetzung der Modul-Maintainer
- Speicherprobleme bei den Bildern der Fingerabdrücke → wenig Platz, NTFS kommt anscheinend schlecht mit **sehr** vielen kleinen Dateien klar

Livedemo

Ausblick auf Milestone 2

Ausblick auf Milestone 2

Weitere Planung bisher (1)

- FPGA Implementierung der Algorithmen
- Matching-Engine $\longrightarrow$ Volle Parallelisierung (done)
- Datenbankzugriff über Headnodes effizient implementiert (done)
- Volles Zusammenarbeiten der Komponenten
- funktionsfähige GUI mit Terminal (done)
- Testdaten beschaffen (im Gange)
- Test der Erkennungsgenauigkeit der Software-Algorithmen

Ausblick auf Milestone 2

Weitere Planung bisher (2)

- Back-to-Back Test der Feature-Extraction Algorithmen mit Referenzimplementierungen

Weitere Planung

- Iris-Kamera ist mit der API von Iris-Guard lauffähig
- Feature-Extraction (Finger) mindestens als Software für den Cluster
- Steigerung der Robustheit und Geschwindigkeit der Software-Matching-Verfahren

**Vielen Dank für Ihre Aufmerksamkeit,
für Fragen steht das Team der PG-CSI noch weiter zur Verfügung**