Projektmappe Gesichtserkennung

OpenBiometry Plattform Server -Client-Kernel

Inhaltsverzeichnis

1. Infrastruktur

Gesichtsbiometrie wird für eine große Anzahl von Anwendungen eingesetzt, und muß in den verschiedensten Projekten zum Einsatz kommen.

Daher ist es notwendig, ein modulares System zu haben, von dem Komponenten je nach Bedarf zusammengesetzt werden können, und zwischen den Komponenten standardisierte Schnittstellen verwendet werden.

Am Anfang haben wir uns die verschiedenen Konzepte der Schnittstellen und Integrationen angesehen, und festgestellt, daß eine einfache TCP Schnittstelle zwischen dem Biometrie Server und beliebigen Anwendungen die beste und universellste Abstraktionsform ist.
Über diese Schnittstelle hat die Anwendung die Möglichkeit mit einfachen Befehlen die Biometrie zu steuern:

EPhilipp
lernt die Person "Philipp" in das System ein.

VPhilipp
verifiziert die Person "Philipp", und liefert entweder
V100
bei erfolgreicher Verifikation, oder
V010
wenn die Person nicht mit dem Template übereinstimmt, oder
V000
wenn keine Person gefunden wurde.
...

Durch diese logische Trennung, haben wir eine Anwendungs-, Hersteller- und Plattform unabhängige Umgebung, die durch TCP/IP Netzwerktransparent, und durch SSH verschlüsselt und Authentifiziert werden kann.
Es ist damit möglich, den Server auf Linux laufen zu lassen, und als Client zum Beispiel eine Zeitabrechnungssoftware auf einem anderen Betriebssystem zu verwenden.

Nachdem sich diese Schnittstelle herauskristallisiert hat, war klar, daß folgende Dinge notwendig sind:

- OpenBio Server, der die verschiedenen Kernel in eine Umgebung einbettet, die Datenbankverwaltung macht, die Ressourcenverwaltung der Kameras, und dann die gesamte Logik über die TCP Schnittstelle zur Verfügung stellt. Der OpenBio Server ist stark an den jeweiligen Biometrie Kernel gebunden, und steht daher in verschiedenen Ausführungen zur Verfügung, wahrscheinlich für jeden Kernel einen.

- OpenBio Clients für die verschiedenen Anwendungszwecke. Als einfachste Umgebung kann ein Telnet Programm verwendet werden, gute Administrationsoberflächen für die Standardapplikationen, und Einbindungen in verschiedene Software von Drittanbietern sind die weiteren Ziele

- OpenBio Kernel

Die Server sind auf die Kernel abgestimmt, und Standardmodule, die man Plug&Play verwenden kann. Es wird Standard Clients geben, mit einfachen Oberflächen zum Bedienen, einloggen,

Administrieren, und spezielle Clients (auch von Drittanbietern), die dann spezielle Dinge wie Arbeitszeiterfassung, ... machen.
Auch eine Umsetzung der Schnittstelle in Bibliotheken und Module für verschiedene Programmiersprachen ist möglich.

Als Alternative ist bei Spezialanforderungen natürlich eine Vollintegration ohne modularem Konzept auch möglich.

Wünsche, die an das Gesamtsystem herangetragen wurden:

Stabilität:
• Dauerbetrieb, laufende Erkennung

Sicherheit:
• Verschlüsselung und Authentifizierung der Datenübertragungen
• Verschlüsseltes speichern von Templates
• Keine Sicherheitslücken

Datenschutz:
• Anonymisierung, Beschränkter Zugriff auf Logfiles

Performance:
• Verarbeitungszeiten < 1 Sekunde

Flexibilität:
• Integrierbarkeit in verschiedenste Anwendungen

Embedded System, damit man nicht für jede Tür einen PC braucht
Parallele Verarbeitung von mehreren Kameras von einer Blackbox
Anbindung an Relais und Taster für Türöffner, Schleusen und Drehkreuze

2. OpenBio Server

Angefangen hat die OpenBio Server Serie mit dem TCPServer genannten Server, der den BioID Kernel zur Verfügung gestellt hat.
Nachdem es viele Probleme mit dem Kernel gegeben hat, und es für die Zutrittslösungen in Richtung Embedded Blackbox weiterging, und bereits eine Login-Lösung unter Linux entwickelt wurde, fiel die Entscheidung, einen OpenBio Server auf Linux zu entwickeln.
Derzeit sind 2 OpenBio Server unter Linux verfügbar: Die neue OpenBiometry Engine und die alte Cognitec Engine. Eine BioID Implementierung ist angedacht.

Der Server besteht aus folgenden Komponenten:

Kameratreiber
Ressourcenverwaltung
Gesichtsfinder
Vorverarbeitung
Feature Extrahierung (Eigenfaces, Lorenz, Graph)
Vergleich
Auswertung
Datenbank
Steuerung
TCP Server

Bezüglich Konfiguration ist das Konzept, daß der OpenBio Server selber keine Einstellungen speichert.
Einstellungen des Servers werden vom Client gespeichert, und bei jedem Verbindungsaufbau an den Server geschickt. Dies hat den Vorteil, daß der Server keine Einstellungen speichern muß, und dadurch Stateless wird. Außerdem löst es das Problem von verschiedenen Clients, die abwechselnd denselben Server ansteuern.

Der OpenBio Server muß als root mittels "/usr/bin/OpenBioServer" gestartet werden, dies passiert als Service : /etc/rc.d/openbio [start|stop]
Das Datenbankverzeichnis ist /var/OpenBio/db/ und kann mittels SHFS/Samba in einem Netzwerk verteilt werden. Das Datenbankverzeichnis ist nur für den Systemadministrator zugreifbar, normale Anwender haben aus Sicherheitsgründen keinen direkten Zugriff darauf.
Der OpenBio Server stellt einen TCP Server auf dem TCP Port 1072 zur Verfügung.
Der genaue Funktionsumfang kann mit dem Kommando H für Hilfe/Help abgefragt werden:

```
sourcerer@linux1:~> telnet servername 1072
Trying 192.168.0.190...
Connected to servername.
Escape character is '^]'.
h
```

OpenBio TCP Server Linux Version 1.5
Hilfe:
E<user> Enroll user
V<user> Verify User
I Identify User
A<user> Automatic Verify/Enroll
H Show this help page
S Status
L<user> Delete User
R0.xx Setzen des Schwellwertes für Identifikation und Verifikation

Beispielsitzung:

EPhilipp
E100
VPhilipp
V100

Noch offene Diskussionspunkte

Mit welchem User soll OpenBio Server gestartet werden?
Der OpenBio Server wird entweder mit root Rechten oder mit einem speziellen Benutzer "openbio" angelegt. Man kann sich aber nicht als Benutzer "openbio" anmelden.

* root
So ist es derzeit eingestellt, daß der Server als root läuft, und dementsprechend auch die Rechte hat,
auf /var/OpenBio/db zu schreiben, den Kameratreiber anzusteuern, ...
Das Laufen als User root hat den Vorteil, daß es nicht von normalen Usern direkt beeinflußt werden kann.
Nachteil: Buffer-Overflows und ähnliche Probleme könnten zur Katastrophe werden.

* Als normaler User
Ist derzeit nicht vorgesehen, könnte aber daraufhin geändert werden.
Vorteil: Server läuft nicht mit root Rechten, und kann direkt von dem Benutzer gestartet werden
Nachteil: Der Server kann vom Eigentümer manipuliert werden
Es ist die Frage, ob der Server dann in jedem Fall die notwendigen Rechte hat, auf die Kamera zuzugreifen, ...

* Als spezieller User
Eine Mischung aus root und normalem User.
Keine Probleme mit Buffer-Overflows, Sicherheit gegen normale Benutzer.
Die Frage der Zugriffsrechte auf die Hardware ist noch offen.

* Privilege Separation
Der Server wird mit root Rechten gestartet, gibt die aber schnell wieder ab.

Daemon oder normales Programm?

Derzeit ist der OpenBio Server als Daemon konzipiert.

* Hat die volle Kontrolle über das System
* Läuft auch, wenn noch keine Kamera im System hängt, versucht bei jeder Anfrage die Kamera zu bekommen.

Das Einsatzszenario in Mittleren Unternehmen ist folgendes: Auf der Embedded Box läuft der Server als Daemon mit root Rechten, und greift so als einzigster direkt auf die Datenbank.
Parallel dazu wird ein normaler user mit automatischen Login eingeloggt, dort über Autostart die grafische Admin Oberfläche gestartet, und mit einem Paßwort versehenen Bildschirmschoner geschützt. Die Admin Oberfläche kommuniziert über TCP mit dem OpenBio Server.

Dadurch hat der normale User nur die Möglichkeiten, die ihm die jeweilige Oberfläche bieten, und kann die Datenbank nicht manipulieren.

Sollen die beiden Programme (OpenBio PAM und OpenBio Server) getrennt werden?
Soll OpenBio PAM auf den OpenBio Server zugreifen, oder die Biometrie voll integriert haben?

Sollen TCP Verbindungen, die nicht von Localhost kommen akzeptiert werden?
Derzeit werden alle Verbindungen akzeptiert.

Nachteil derzeit: Sicherheitsproblem durch Denial of Service, unverschlüsselte Verbindungen über das Netzwerk

Mögliche Lösung:
Nur Verbindungen von Localhost erlauben, und die Verbindungen, die von außen notwendig sind durch SSH tunneln:
Dadurch authentifizieren sich die Clients automatisch, und die Verbindungen sind verschlüsselt.

ssh -L 1072:server:1072 user@server

Die Idee ist es, unter Linux mehrere wirklich parallele Biometrie Services gleichzeitig laufen zu lassen:

Wie sollen parallele OpenBio Server gestartet werden?
Als ein Server, der über Forking die Kontrolle hat, und die Datenbank zum Beispiel über IPC Mechanismen abgleicht, ...?
(Datenbankabgleich ist derzeit nicht notwendig, weil der OpenBio Server die Daten nicht cacht, sondern das caching der Linux-Speicherverwaltung überläßt.)

Oder starten der einzelnen Server, als mehrfaches Starten des Servers?

Wie sollen die einzelnen parallelen Server die Ressourcenverwaltung machen?
(Kameras können im laufenden Betrieb an, ab und umgesteckt werden)

Als grafische Oberfläche für den Monitor (Videobild mit Gesichtsfinder) wird derzeit die Highgui Bibliothek des OpenCV Projektes verwendet. Es besteht auch die Möglichkeit, das Kamerabild als .jpg Datei abzuspeichern.

Bekannte Probleme der OpenBio Engine:

Die Auswertungsfunktion des Gesichtsvergleichs dürfte noch fehlerhaft sein.

Bekannte Probleme der Cognitec Engine:

Man darf keine 0x0 Pixel großen Bilder von der Kamera in den Kernel hineinschicken.

Bei direkter Einbindung als Bibliothek in ein PAM Modul hängt sich der Biometrie Kernel reproduzierbar auf, ein Stack-Überlauf wird angenommen.

Es können mit dem Cognitec Kernel maximal 10 Templates gleichzeitig zur Identifkation hergenommen werden.

3. OpenBio Client

Die einfachste Variante, die Schnittstelle zu bedienen, ist einen Telnet Client zu verwenden, und auf den Port 1072 zu verbinden.

Unter Linux steht inzwischen ein OpenBio Client für KDE3 zur Verfügung, und ein Automatik-Client.

4. OpenBio Kernel

Die Entwicklung des OpenBio Kernels hat im September 2002 begonnen

Die anfängliche Idee ist, die Gesichtserkennung in folgenden Schritten zu machen:

Augenfindung
Bewegungserkennung
Gesichtsfindung
Qualitätscheck
Freistellung des Gesichts
Feature Extrahierung
Vergleich

Das derzeitig (Stand Juli 2003) verwendet Konzept der OpenBiometry Engine läuft nach folgendem Konzept:

Gesichtsfinder (Haarlike, OpenCV)
Freistellung des Gesichtes, Kontrastspreizung, Auto-Gamma
Feature Extrahierung (Eigenfaces, libface)
Vergleich (Eigenfaces, libface)

Geplant ist vor die Freistellung noch eine Augenerkennung einzubauen, und dann bei der Vorverarbeitung die automatische Rotation mitnutzen, und auch die

Der Gesichtsfinder von OpenCV hat sich als extrem robuster und performanter Gesichtsfinder herausgestellt

Der Gesichtsfinder vergleicht die Gesamthelligkeit von Rechtecksbereichen im Gesicht miteinander.
Die Stirn ist zum Beispiel heller als der Augenbereich, ...

Der Gesichtsfinder liefert dann die Koordinaten aller gefundenen Gesichter.
Aus den gefundenen Gesichtern wird das größte ausgewählt. Dies hat den Grund, daß wenn jemand einer anderen Person über die Schulter schaut, daß die Person, die der Kamera am nächsten ist zum Vergleich hergenommen wird.

Die Vorverarbeitung schneidet dann das gefundene Gesicht aus, skaliert es auf eine vorgegebene Größe, macht eine Kontrastspreizung und dann eine automatische Gammakorrektor, um eine durchschnittliche Helligkeit des Bildes zu erreichen.

Das extrahierte Gesicht wird dann in einen multidimensionalen Vektor umgerechnet (Eigenspace), und dann mit wird im multidimensionalen Raum der Abstand zum Referenzvektor verglichen.

5. Begriffserklärung

PAM	Pluggable Authentication Modules, Loginsystem von Linux und Solaris
Cognitec	Gesichtserkennungs Hersteller
E	Einlernen
V	Vergleichen
I	Identifizieren
TCP/IP	Transmission Control Protocoll/Internet Protocoll, das Basisprotokoll des Internets
Server	System, das einen Dienst zur Verfügung stellt. Webserver, Mailserver, ...
Daemon	Eine Software, die im Hintergrund läuft
Denial of Service	Angriff, der das Anbieten oder Verwenden eines Dienstes blockiert.
Buffer Overflow	Angriff, der durch das überlaufen eines Puffers gemacht wird.
OpenCV	Open Computer Vision Library, eine Bildverarbeitungsbibliothek, die von Intel angefangen wurde.
BioID	Gesichtserkennungs System der Firma Humanscan
Eigenface	System, bei dem ein Bild in einen multidimensionalen Vektor umgerechnet wird, und diese Vektoren dann verglichen werden können.