INSTITUTO POLITÉCNICO DE BEJA

Escola Superior de Tecnologia e Gestão

Mestrado em Engenharia de Segurança Informática

Sistema de Autenticação Biométrica Facial

Metodologia de Análise de Componentes Principais Modificada

Joel Alexandre Sequeira Silvestre

Beja

2015

INSTITUTO POLITÉCNICO DE BEJA

Escola Superior de Tecnologia e Gestão

Mestrado em Engenharia de Segurança Informática

Sistema de Autenticação Biométrica Facial

Metodologia de Análise de Componentes Principais Modificada

Dissertação de Mestrado

Elaborado pelo aluno:

Joel Alexandre Sequeira Silvestre

Dissertação orientada por:

Doutor José Jasnau Caeiro

Beja

2015

Agradecimentos

A realização desta dissertação marca a minha conclusão no Mestrado de Engenharia de Segurança Informática e contou com apoios e incentivos muito importantes, que me deram força e motivação ao longo desta caminhada. Gostaria de deixar alguns agradecimentos a todos aqueles que contribuíram de uma forma mais presente para a sua concretização e que sem eles, não teria sido possível.

Um especial agradecimento ao meu orientador e companheiro nesta minha caminhada, o Doutor José Jasnau Caeiro, não só pelo apoio técnico e por todas as suas contribuições no meu trabalho, como também a nível psicológico e motivacional. Foi um gosto tê-lo como orientador.

Aos meu circulo de amigos e colegas mais próximos, que sempre me apoiaram durante esta fase e que me deram especial força em momentos mais difíceis.

À Maria Teresa Couto, pelo companheirismo, apoio e constante motivação. Partilhou este percurso comigo e é com apreço e carinho que lhe deixo um especial obrigado.

Por último, manifesto um sentido e profundo reconhecimento à minha família, os meus pais e meu irmão, pelo apoio incondicional não só ao logo desta dissertação mas ao logo de todos os anos de vida académica até aqui.

Resumo

A segurança em sistemas de informação através de autenticação biométrica, baseada em características físicas intrínsecas ao ser humano, tem vindo, a ser apontada como a solução futura para problemas de autenticação. Há poucos anos, a autenticação biométrica era utilizada apenas em grandes sistemas de segurança governamentais mas, hoje em dia, devido ao aperfeiçoamento da tecnologia e à redução de custos dos dispositivos, até mesmo num *smartphone* se disponibiliza um sistema de autenticação biométrico. Esta dissertação aborda a autenticação biométrica como forma de melhorar a segurança em sistemas de informação, em particular a autenticação através de reconhecimento facial.

Foi desenvolvido um sistema de segurança com base na autenticação biométrica em ambiente web, utilizando o algoritmo *Transform-Invariant PCA*, que deriva do tradicional *Principal Component Analysis* (PCA), conjugado com um alinhamento nas faces proporcionado pelo *Simultaneous Inverse Compositional* (SIC). O sistema consiste numa plataforma web em que a parte de servidor é responsável pelo processamento biométrico e a de cliente por recolher características e apresentar o resultado da autenticação.

Neste trabalho, serão apresentados e detalhados os métodos e mecanismos utilizados no sistema de autenticação por reconhecimento facial implementado, examinando e discutindo os seus benefícios e limitações. A presente dissertação apresentará ainda uma explicação do que é um sistema biométrico, a sua arquitetura, os seus tipos, o que deve respeitar, vantagens, desvantagens e fará compreender de que modo está relacionado com a autenticação e de que forma pode melhorar a segurança em sistemas de informação.

Abstract

The security in information systems through biometric authentication, based on physical characteristics intrinsic of the human being, has come, progressively, being pointed as the future solution for authentication problems. A few years ago, biometric authentication was only used in big governmental security systems but, nowadays, thanks to the developments of technology and cost reduction of electronic devices, even on a smartphone, for example, one can have a biometrical authentication system. This dissertation addresses biometrical authentication as a way to improve the security in information systems, in particular the authentication through facial recognition.

A security system was developed based on biometrical authentication in a web environment, using the algorithm Transform-Invariant PCA, the is a derivation of the traditional Principal Component Analysis (PCA), conjugated with an alignment of faces, which is provided by the Simultaneous Inverse Compositional (SIC). The system consist of a web platform with two parts: the server, responsible for biometrical processing, and the client, which collects characteristics and presents the final result of the authentication.

In this document, will be presented and described the methods and mechanisms applied in the facial recognition authentication system used, examining and discussing its benefits and limitations. The present dissertation will also give an explanation of what is a biometrical system, his architecture, types, what has to respect, advantages, disadvantages, how it is related to authentication and how it can improve the security in information systems.

Conteúdo

Lista de Figuras

Capítulo 1

Introdução

Nos últimos anos, a evolução tecnológica, tanto em *software* como *hardware*, tem permitido que qualquer pessoa tenha a possibilidade, em qualquer lugar, através de um computador, telemóvel ou *tablet*, de aceder facilmente à sua conta nas redes sociais, ao seu *email*, efetuar compras e pagamentos *online*, entre outros. Esta evolução deve-se ao facto de a Internet ter tido uma expansão global e nos permitir estar ligados em rede a todo o mundo a cada instante. Esta evolução trouxe também o aumento em massa da informação no mundo digital, exigindo cada vez mais meios de segurança e proteção. Um dos maiores problemas a nível de segurança é a identidade. Conseguir provar que a pessoa que está a aceder à informação é aquela que realmente diz ser, é um problema.

Hoje em dia os mecanismos mais utilizados na verificação de identidade são *passwords* ou cartões magnéticos, no entanto, pretende-se que as tecnologias biométricas venham elevar o grau de certeza e substituir os mecanismos mais convencionais. Apesar de se reconhecer que uma *password* por si só não é suficiente, as preocupações sobre a confiabilidade da tecnologia biométrica permanecem[19]. Ken Munro sita que «a tecnologia biométrica é a desejada mas continua a limitar-se à segunda escolha na autenticação»[19]. Ainda assim, a biometria começa a ter grandes avanços não só enquanto tecnologia mas também na sua presença na sociedade. Justin Hughes diz que, «a biometria está a atravessar a curva clássica de tecnologia: é cada vez mais rápida e os sistemas estão cada vez mais baratos».[19]. Em 2013, a Apple introduziu pela primeira vez um *scanner* de impressão digital, no modelo IPhone 5[17], no entanto, também em 2013, investigadores de segurança do CCC[1]

[1] A CCC é uma associação europeia de *hackers*. `https://www.ccc.de/en/`

(Chaos Computer Club) criaram um dedo falso que alegam facilmente poder falsear qualquer sistema biométrico, de impressão digital. O que indica que a biometria necessita de aperfeiçoamento, para que se possa tornar a primeira escolha em sistema de autenticação.

As empresas começam a pensar cada vez mais na segurança, querendo inovar e apostar nas tecnologias biométricas. Começam a ganhar interesse das instituições alguns eventos como a «Biometrics»[2], uma conferência internacional que abordar a atualidade das tecnologias biométricas, apresenta novos avanços, e permite com isto que as organizações escolham a tecnologia biométrica mais indicada para as suas instituições.

Existem cada vez mais entidades como aeroportos, empresas no sector financeiro, saúde, judicial, entre outros, que precisam de assegurar que somente pessoas devidamente autorizadas têm acesso a determinada informação, serviço ou locais restritos. O FBI tem-se mostrado um dos pioneiros na biometria, melhorando a ação forense e consequentemente a judicial[5]. No setor bancário, na Turquia por exemplo, em 2012, foram introduzidas tecnologias biométricas em mais de 3.000 ATM's, permitindo aos consumidores levantar dinheiro sem a necessidade de qualquer verificação adicional[17]. Nos aeroportos, tanto em Portugal como em toda a Europa, já existem terminais onde é feito o *check out* através da validação facial. Qualquer situação em que seja necessária uma confirmação de identidade tem um propósito associado, a segurança.

Este trabalho incide na autenticação em sistemas informáticos através de tecnologias de biometria de reconhecimento facial. A técnica *Transform-Invariant PCA*[8] (TIPCA), concebida recentemente, foi estudada e aplicada nesta dissertação. O que diferencia esta técnica das restantes é o facto de efetuar um alinhamento das imagens de forma a tornar mais evidentes os componentes principais encontrados pelo tradicional algoritmo PCA. Perante os resultados deste trabalho, perceberemos qual a importância da extração das principais características das faces humanas, o impacto de alinhar previamente as caras presentes nas imagens recolhidas e ainda a diferença entre utilizar linguagens de alto e baixo nível para processar uma elevada quantidade de dados. O alinhamento das faces humanas é o ponto diferenciador deste

[2]Conferência internacional, realizada em Londres, anualmente, e que conta com a presença de grandes referências na área e grandes empresas todos os anos. `http://www.biometricsandidentity.com`.

algoritmo, seguindo a tendência do DeepFace, sistema de reconhecimento facial do Facebook[28].

1.1 Objetivos e Motivação

O principal objetivo desta dissertação é apresentar uma técnica de reconhecimento facial e demonstrar de que forma pode ser possível melhorar outras já existentes. O algoritmo *Transform-Invariant PCA*, que é responsável pela parte lógica do sistema, foi implementado com este objetivo. Outro dos objetivos foi dar corpo ao algoritmo e desenvolver um sistema de autenticação biométrico para a web, com recurso ao reconhecimento facial. Este é composto por um servidor, onde é processada a informação biométrica, e pelo cliente, que terá como função apresentar resultados e recolher características biométricas. A autenticação biométrica e as suas vantagens e desvantagens são apresentadas como uma alternativa que se mostra viável, no reforço da segurança nos sistemas de informação.

A motivação na escolha do tema deve-se ao gosto pela biometria e pela área de visão por computador, com o intuito de estudar um pouco mais que os algoritmos tradicionais de reconhecimento de padrões e adquirir conhecimentos numa área que se prevê que venha a fazer parte importante do futuro tecnológico na área da segurança informática.

1.2 Enquadramento

Nos últimos 30 anos, desde o aparecimento da Internet, o mundo tem assistido a inúmeras transformações, não só tecnológicas mas também sociais. Atualmente, existem cada vez mais plataformas aplicacionais online, que facilitam o dia a dia das pessoas, possibilitando efetuar transações, compras, etc. Esta evolução requer inerentemente maior segurança nestes sistemas, que deve começar por um processo de verificação/validação da identidade de um indivíduo, ao qual se dá o nome de autenticação. Uma autenticação segura e robusta é o componente principal da segurança de um sistema fidedigno.

A identidade é um conjunto de dados que permite reconhecer uma pessoa e distingui-la das demais. Existem três formas de a provar (Vielhauer 2006, pág.3)[33]:

Conhecimento: Consiste em algo que a pessoa sabe ou tem memorizado, como uma *password*, uma frase ou um código PIN.

Posse: Baseia-se na propriedade especifica de um artigo ou prova, como uma chave ou cartão magnético.

Biometria Características específicas, fisiológicas ou comportamentais de um indivíduo.

Cada uma destas abordagens pode aferir, de alguma forma, a identidade. No entanto, os mecanismos, tanto de conhecimento como de posse, são facilmente destronáveis uma vez que códigos PIN, *passwords*, cartões ou chaves podem ser perdidos, esquecidos ou roubados, podendo assim ser forjada uma verificação de identidade, pelo que nenhum destes métodos é totalmente confiável.

Posto isto, existe cada vez mais a necessidade de validar a identidade de uma forma conclusiva e com uma probabilidade de erro quase nula ou mesmo inexistente. A certeza em sistemas de autenticação, contudo, e seja qual for o método utilizado, não é fácil de obter. Hoje em dia a biometria é o método que consegue oferecer mais garantias. De entre todas as formas de sistemas biométricos, existem alguns que se destacam pelas suas baixas taxas de ocorrência de falsos positivos, como é o caso da retina, íris e ADN. Estas opções exigem grandes recursos tecnológicos e são demasiado intrusivas, razão pela qual não são tão utilizadas.

Os sistemas de controlo e validação de identidade de pessoas têm vindo a evoluir no sentido de evitar erros de validação, trocas e violação de identidade, evitando consequentemente o acesso indevido a informação e/ou locais privados. Depois de muitos anos de pesquisa e desenvolvimento em tecnologias biométricas, o reconhecimento facial[22] tem sido um dos mais utilizados. Este permite uma identificação à distância, não obriga a uma interação direta entre a pessoa e o equipamento, e a sua recolha de dados é feita apenas através de uma câmara incorporada num dispositivo. Apesar de ser uma tarefa que exige algum esforço computacional, os dispositivos da atualidade conseguem facilmente dar resposta às necessidades, refletindo-se em custos muito reduzidos. Hoje, os cartões de identificação que são emitidos aos empregados para acessos físicos e a informação, bem como os cartões que são usados para transações financeiras, muitas vezes incluem informação biométrica. Um exemplo disso é o cartão de cidadão português, que já contém tanto a impressão

digital como a fotografia da face da pessoa, para posterior identificação por reconhecimento facial. Ambos os setores, público e privado, estão à procura de métodos fiáveis, precisos e práticos para a verificação automática da identidade. Existe uma aposta cada vez mais forte nas tecnologias biométricas. Por exemplo, «desde 2001, aeroportos em todo o mundo têm aumentado o número de instalações de sistemas de segurança com reconhecimento biométrico»[24], permitindo aos viajantes transitar para a zona de embarque de uma forma mais rápida. Grandes organizações governamentais têm feito também uma aposta muito grande na biometria, como é o caso do FBI que está empenhado em utilizar biometria e sistemas de gestão de identidade em toda a parte, dando origem em 2007 ao «*Biometrics Center of Excellence* (BCOE) como meio de coordenar pesquisas, desenvolvimento e aplicação de sistemas biométricos»[5], que são posteriormente utilizados em todo o USG (*United States Government*). Até mesmo para efetuar um simples *login* num computador pessoal já é utilizado muitas vezes o reconhecimento facial ou impressão digital, em vez da tradicional *password*. Concluindo, a evolução na área biométrica está em contínuo crescimento e encontra-se cada vez mais presente nos nossos dias.

1.3 Estrutura do Documento

Este trabalho é composto por um primeiro capítulo introdutório, com um enquadramento geral na área da segurança biométrica em sistemas de informação. É dada uma perspetiva atual do tema e são introduzidos conceitos essenciais ao seu enquadramento. São também propostos e descritos os objetivos que se pretendem com a elaboração deste trabalho e é apresentada a estrutura do documento.

O segundo capítulo tem como objetivo introduzir o leitor no tema. Contém uma explicação mais detalhada, dando algumas noções e esclarecimentos de biometria e apresentando os processos que constituem um sistema biométrico, as técnicas existentes, e a relação entre a biometria e sistemas de informação. Mostra de que forma os sistemas biométricos podem falhar e quais os seus tipos de falhas. Aponta as vantagens e desvantagens da utilização de sistemas biométricos e esclarece a diferença entre verificação e identificação biométrica.

O terceiro capítulo aborda o tema principal que é a autenticação por reconhecimento facial, dando a conhecer o que já existe na área, apresentando a arquitetura do sistema e descrevendo as fases de cada etapa.

O quarto capítulo expõe aprofundadamente os algoritmos utilizados, fundamentando matematicamente o seu funcionamento. Apresenta passo a passo a descrição de cada um dos algoritmos e as demonstrações efetuadas para a percepção dos mesmos. Além do conteúdo matemático são também descritos os algoritmos, como surgiram e a sua origem.

O quinto capítulo apresenta a estrutura do sistema implementado e as suas componentes. Descreve todas as tecnologias e ferramentas utilizadas na elaboração deste trabalho, demonstrando como funcionam e interagem os diferentes componentes. Este capítulo tem um grande foco na descrição do funcionamento do *Web Browser* e *Web Server*, contendo diagramas de sequência nos diferentes casos de uso.

O sexto capítulo evidencia os resultados experimentais do funcionamento em função do protocolo experimental estabelecido. É feita ainda uma análise desses resultados, e elaboram-se alguns dos gráficos para o seu melhor entendimento. Foram feitos testes de aceitação, de redução de dimensionalidade, de alinhamento de imagem, entre outros.

O sétimo capítulo apresenta as conclusões de todo o trabalho, referindo quais são as vantagens e desvantagens deste sistema, enunciando também algumas propostas e ideias para trabalhos futuros finalizando a dissertação.

Capítulo 2

Sistemas Biométricos

2.1 Fundamentos Biométricos

≪O termo biometria deriva do grego e tem dois significados distintos: *bios* (vida) e *metron* (medida)≫[31]. Na autenticação, refere-se à utilização de características próprias de um indivíduo para proceder à verificação ou identificação da sua identidade. Uma característica biométrica só é classificada como tal se satisfizer alguns requisitos básicos[25]:

Universalidade: Todos os indivíduos registados no sistema devem apresentar as mesmas características físicas e biológicas, tais como, dedos, íris, face e DNA, isto porque podem existir pessoas que, devido a deficiência ou acidente, não contenham estas características;

Unicidade: Uma característica biométrica deve ser única para cada indivíduo, ou seja, a possibilidade de pessoas distintas possuírem características idênticas, deve ser nula ou desprezível. A altura não é uma boa característica, por exemplo, já que várias pessoas possuem a mesma altura e não existe forma de ser diferenciada;

Permanência A característica deve ser imutável. Na prática, existem alterações ocasionadas pelo envelhecimento, pela mudança das condições de saúde ou mesmo devido a acidentes que alterem o estado físico da pessoa. No entanto, salvo exceções, muitas características permanecem praticamente inalteradas ao longo da vida do ser humano;

Coletável: A característica tem que ser mensurável por meio de um dispositivo. Na prática, todas as características biométricas utilizadas conseguem de alguma forma atender a este requisito;

Aceitação: A recolha da característica deve ser tolerada pelo indivíduo em questão. Na prática, existem preocupações com privacidade e questões culturais que diminuem a aceitação da recolha. Por exemplo, o reconhecimento facial em países como o Afeganistão, onde muitas muçulmanas adotam a burca, não é uma opção viável;

Evasão/Facilidade de Recolha: A recolha deve ser o menos intrusiva possível e de fácil aquisição.

Na realidade, nenhuma característica biométrica consegue atender de uma forma inequívoca a todos os requisitos, o que impede o conceito de característica biométrica ideal. Ao contrário de sistemas baseados no conhecimento e posse, «sistemas biométricos não dão uma resposta de sim ou não mas sim uma percentagem de similaridade»[32]. Em biometria, ao contrário dos sistemas de segurança convencionais, em que se pode saber ou não uma *password*, ter ou não um cartão magnético, o processo de decisão não é assim tão direto, pois existe sempre um erro associado. A biometria tem uma taxa de similaridade que dará origem a uma decisão segundo o limite de aceitação definido no sistema[1]. Uma característica biométrica dá origem a uma tecnologia, caracterizando tecnologias biométricas como «métodos automatizados para identificar uma pessoa ou verificar a sua identidade com base em características fisiológicas ou comportamentais do ser humano»[22].

A figura (2.3) representa um organograma que divide o sistema biométrico em dois tipos de características possíveis, físicas ou comportamentais, apresentando vários exemplos de tecnologias em cada um deles.

[1]Na secção 3.2.4 deste trabalho, esta relação entre a decisão e a taxa de similaridade é apresentada em detalhe.

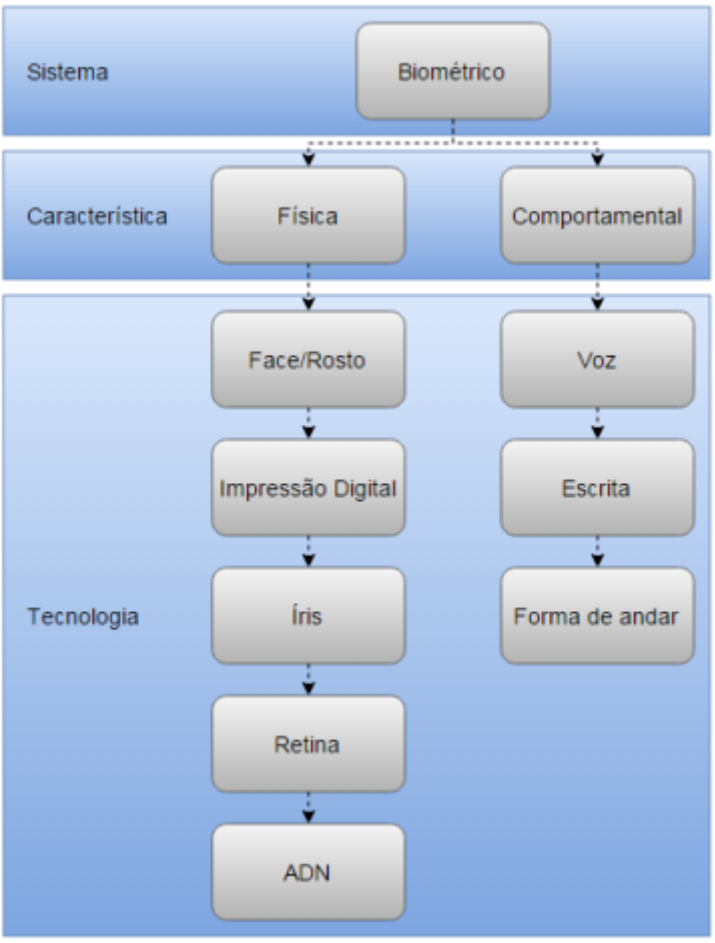


FIGURA 2.1: Diagrama com técnicas biométricas.

A melhor característica biométrica para um sistema de autenticação é aquela que melhor se adaptar às necessidades pois, como se pode constatar através da tabela seguinte, cada tecnologia tem vantagens e desvantagens. Tendo em conta os requisitos básicos descritos anteriormente e as tecnologias biométricas apresentadas na figura anterior, foi construída uma tabela que indica os pontos fortes e fracos das tecnologias existentes, classificando os requisitos em três níveis: alto, médio e baixo.

	Universalidade	Unicidade	Permanência	Coletável	Aceitação	Evasão
Face	Alto	Baixo	Médio	Alto	Alto	Baixo
Impressão D.	Médio	Alto	Alto	Médio	Médio	Alto
Íris	Alto	Alto	Alto	Médio	Baixo	Alto
Retiana	Alto	Alto	Médio	Baixo	Baixo	Alto
ADN	Alto	Alto	Alto	Baixo	Baixo	Baixo
Voz	Médio	Baixo	Baixo	Médio	Alto	Baixo
Escrita	Baixo	Baixo	Baixo	Alto	Alto	Baixo
Caminhar	Médio	Baixo	Baixo	Alto	Alto	Médio

TABELA 2.1: Classificação do nível de cada tecnologia perante os requisitos base.[13]

Na tecnologia com base em características da face, são apontados apenas dois pontos fracos, a unicidade e a permanência. A unicidade deve-se ao facto de o rosto do ser humano ser muito parecido entre indivíduos e difícil de distinguir. Na nossa sociedade conseguimos rapidamente distinguir uma pessoa, no entanto, caso nos encontremos noutra sociedade, como por exemplo a chinesa, teremos muita dificuldade em efetuar o reconhecimento, isto precisamente porque a unicidade entre faces humanas é baixa. O facto de a permanência ser também um ponto fraco explica-se devido ao envelhecimento, por exemplo, sendo classificado num nível médio pois são alterações que não são imediatas mas sim graduais. Quanto aos pontos positivos aponta-se o baixo nível de evasão, consequentemente um alto nível de aceitação e também um nível elevado para a universalidade, pois embora possam existir exceções todos os utilizadores têm um rosto constituído de igual modo.

Vantagens e Desvantagens em sistemas biométricos

As vantagens mais notórias da utilização de biometria em vez dos tradicionais métodos de autenticação são:

- Características biométricas são inerentes ao ser humano, não podem ser esquecidas, perdidas ou simplesmente fornecidas a outra pessoa;
- A probabilidade de existirem duas pessoas com as mesmas características biométricas, sejam elas quais forem, é praticamente nula;

- São muito difíceis de copiar.

Quanto às desvantagens podem ser apontadas algumas como:

- Alterações graduais ou repentinas nas características biométricas do ser humano, devido a envelhecimento ou acidente, por exemplo.
- No caso de cópia de características biométricas, estas não devem voltar a ser utilizadas pela pessoa legítima uma vez que não podem ser alteradas.

2.2 Segurança Biométrica

A segurança em qualquer sistema de informação está diretamente relacionada com três conceitos fundamentais[34]:

Confidencialidade: Garante que a informação somente é revelada com autorização apropriada;

Integridade: Garante que a informação somente pode ser alterada com autorização apropriada;

Disponibilidade Garante que a informação seja acessível aos legítimos titulares, quando requerida;

Ultimamente têm sido apontados mais dois novos conceitos, o não-repúdio, que garante que a transação chegou ao seu destino sem a possibilidade de renúncia do recetor, e a **autenticação**, que garante que cada entidade é aquela que alega ser.

Num sistema biométrico, existem duas frentes de segurança que devem ser tomadas em conta, uma delas relacionada com a proteção dos próprios dados biométricos, armazenados em base de dados, outra, relacionada com os dados que são transmitidos entre as diferentes componentes do sistema.

A segurança em sistemas biométricos pode ser avaliada segundo dois aspetos importantes:

Desempenho do sistema: Representado segundo as taxas de erro FAR e FRR, abordadas na secção 2.5;

Robustez do sistema: Define-se quanto à vulnerabilidade a possíveis ataques ao sistema;

«Um sistema biométrico é vulnerável a diferentes tipos de ataques, podendo comprometer a segurança proporcionada, resultando assim numa falha»[15]. Vulnerabilidades são definidas por qualquer fraqueza de um sistema, podendo ser aproveitadas para violar a segurança. Elas surgem normalmente a partir de fragilidades na arquitetura, na implementação ou na comunicação. Atualmente, com o avanço da tecnologia e estudos científicos cada vez mais aprofundados, conseguiu-se descobrir vulnerabilidades destes sistemas. Algumas delas em sistemas de reconhecimento biométricos, representadas segundo o modelo *fish-bone*[14].

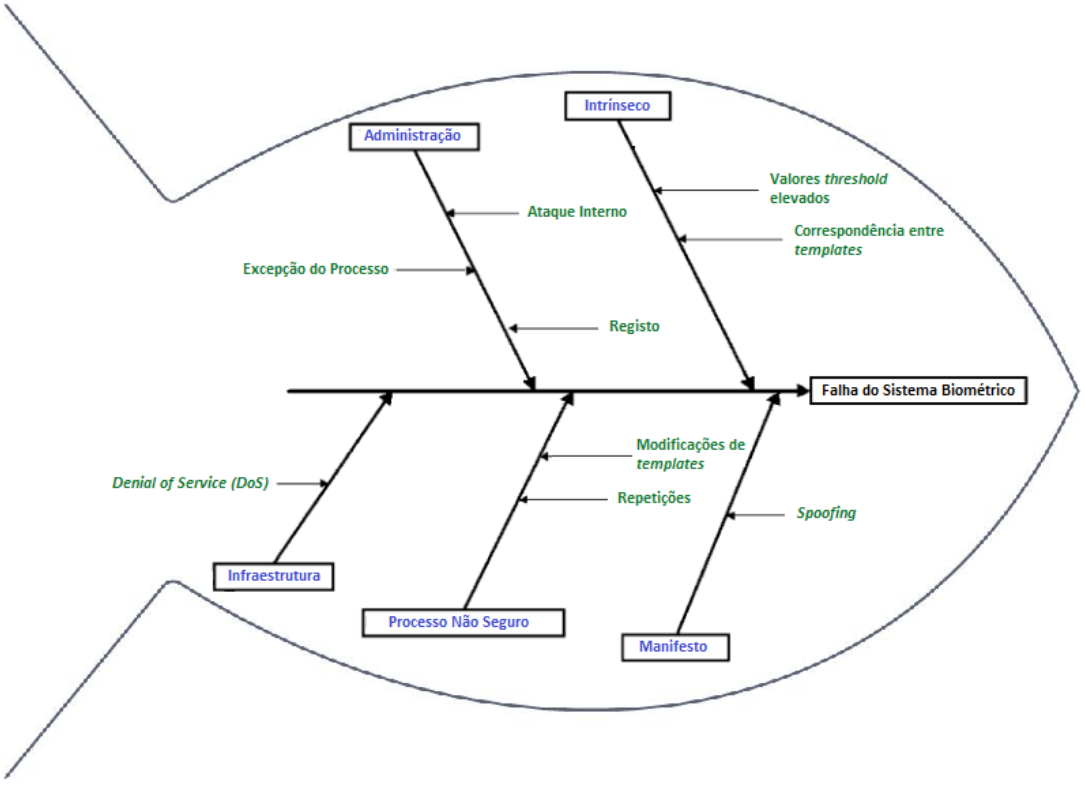


FIGURA 2.2: Modelo de vulnerabilidades numa representação em espinha *fish-bone*.

As causas dum possível fracasso num sistema biométrico, representadas no modelo da figura 2.2, podem ser organizadas em duas principais categorias[14]:

Falha intrínseca: Acontece devido às limitações das tecnologias na aquisição de características, como por exemplo *webcams* com fraca qualidade de imagem;

Falha devido a ataques: Acontece quando há ataques por parte dos impostores e estes tentam corromper o sistema para benefício próprio;

2.2.1 Falha intrínseca

«Esta falha ocorre quando são tomadas decisões incorretas por parte do sistema»[15]. Um utilizador válido pode ser recusado pelo sistema devido a diferenças entre *templates* armazenados e o *template* extraído na hora da autenticação. As diferenças podem emergir perante uma má interação entre o utilizador e o sistema biométrico ou por insuficiências do próprio sistema. Devem ser definidas algumas regras[2] para tentar minimizar estas situações. Um utilizador inválido pode ser também aceite no sistema, isto ocorre devido a demasiadas semelhanças dos *templates* biométricos, pondo em causa as propriedades de unicidade e singularidade das características biométricas, já referidas neste trabalho (2.1). Ambas as situações consistem numa falha intrínseca do sistema.

2.2.2 Falha devido a ataques

Estas falhas surgem devido a ataques intencionais, cujo sucesso depende principalmente da robustez do sistema, recursos computacionais e nível de conhecimento do impostor. Estas falhas são classificadas em três tipos, segundo o modelo *fish-bone*: Ataque ao sistema administrativo, ataque a infraestruturas não seguras e ataque a características biométricas[15].

Ataque ao sistema administrativo - Este ataque é considerado como uma ataque interno sendo que explora as vulnerabilidades existentes em áreas de administração, reservadas a utilizadores privilegiados.

Ataque a infraestruturas não seguras - Hardware, software, protocolos de transmissão de informação entre os vários componentes do sistema, tudo isto representa a infraestrutura de um sistema biométrico. Estes ataques são classificados por alterações, substituições e interceção de comunicações à infraestrutura do sistema. Exemplos destes ataques existem são: Intercetar características biométricas na fase de registo que podem ser posteriormente utilizadas para pedidos de autenticação maliciosos; *Templates* armazenados em base de dados podem ser substituídos, permitindo que o impostor se autentique com as suas

[2]Esta regras são definidas na secção 3.2.1.

próprias caraterísticas; Intercetar as comunicações entre os vários componentes do sistema, substituindo o *template* que procura autenticar-se; Modificar o método de classificação, alterando o limiar da aceitação.

Ataque a características biométricas - Este ataque consiste na aquisição física das características de um indivíduo. Existem técnicas biométricas que são mais vulneráveis a este tipo de ataques como é o caso da impressão digital, em que ao tocar com o dedo em algo, é possível recolher uma amostra e reproduzir um dedo artificial exatamente com as mesmas características do indivíduo plagiado.

2.3 Processos de Reconhecimento

Independentemente da tecnologia biométrica utilizada, os processos descritos adiante, bem como as fases de cada processo, são idênticos. Estes sistemas têm dois processos distintos, que são: o registo e a verificação ou identificação[3]. Em todos estes processos, existem fases que são similares a cada um deles, como a aquisição das características, o pré processamento e a extração de características. Descreve-se o que é feito em cada uma destas fases:

- A **aquisição de caraterísticas** é feita através de dispositivos próprios que dependendo da tecnologia utilizada. A amostra biométrica capturada é transformada num perfil biométrico a que se dá o nome de *template*.
- O **pré processamento** é a fase em que é feita uma normalização e eliminação de ruído do *template* e tem uma importância fundamental nos resultados do sistema.
- A **extração de características** é a etapa responsável por encontrar caraterísticas diferenciadoras e que permitirão uma classificação/decisão.

Estas etapas serão pormenorizadas e enquadradas no âmbito deste trabalho no próximo capítulo, secção 3.2.

Posto isto, os processos que compõem um sistema biométrico são os seguintes:

[3] A diferença entre sistemas baseados em identificação ou verificação é explicada na secção 2.4 deste trabalho.

2.3.1 Registo (Enrollement)

O registo é feito a partir da captura de amostras biométricas de um indivíduo, para que fiquem num formato digital, formando assim os *templates*. Estes são armazenados numa base de dados para posterior comparação no processo de verificação ou identificação. Para além das características, o utilizador tem de fornecer também a sua identificação, visto que o *template* armazenado ficará associado à identificação fornecida no ato do registo.

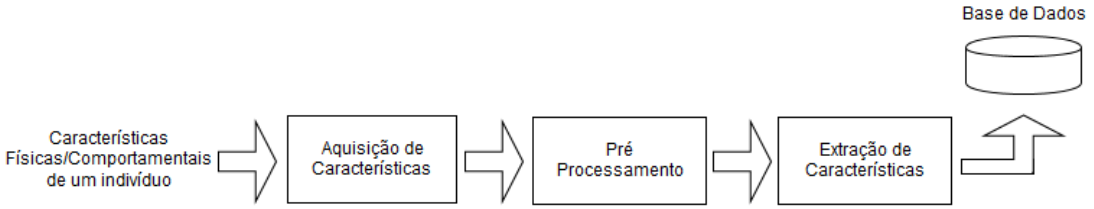


FIGURA 2.3: Processo do Registo em Sistemas Biométricos

2.3.2 Verificação

A verificação biométrica é o processo em que o utilizador apresenta as suas características biométricas juntamente com a sua identificação. Depois de ser construído o seu *template* biométrico, este vai ser comparado com o *template* armazenado em base de dados que corresponde à identidade fornecida, a fim de confirmar ou negar a alegada identidade. «Verificação é referida frequentemente como uma pesquisa um para um»[31], respondendo à questão «o indivíduo é quem alega ser?». O resultado deste processo é uma confirmação ou negação de identidade. A figura 2.4 ilustra a arquitetura desde processo.

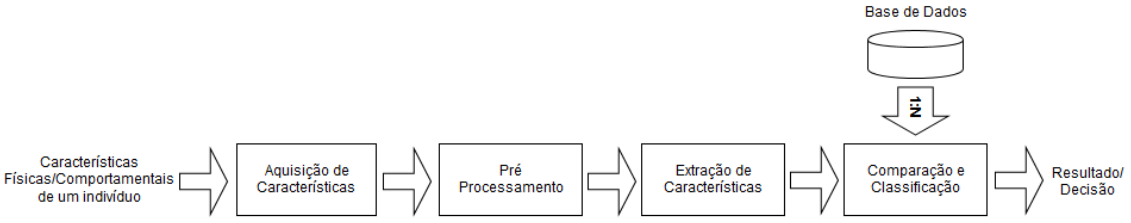


FIGURA 2.4: Processo do Verificação em Sistemas Biométricos

2.3.3 Identificação

Na identificação biométrica, o utilizador fornece apenas as suas características biométricas competindo ao sistema identificar o sua identidade, «contrariamente ao processo de verificação onde é necessária a prévia identificação do utilizador»[31]. Depois de extraído o *template* biométrico, este vai ser comparado com todos os registos na base de dados, a fim de identificar, ou não, o utilizador. «A identificação designa-se como uma pesquisa de um para vários»[31]. O sistema é responsável por responder à pergunta, «quem é o indivíduo?». O seu resultado deve ser a identidade do utilizador. A sua arquitetura é idêntica à da verificação, varia apenas no modo de comparação e no resultado alcançado, como mostra a figura 2.5.

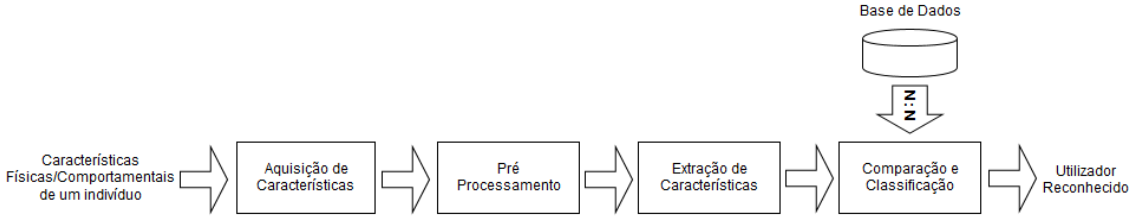


FIGURA 2.5: Processo do Identificação em Sistemas Biométricos

2.4 Verificação e Identificação Biométrica

Os sistemas de autenticação biométricos podem ser classificados em dois tipos, e é fundamental conseguir fazer a distinção entre eles.

No caso da verificação, se um conjunto de características biométricas for suficientemente semelhante a um outro conjunto previamente registado no sistema, relativo à pessoa que reivindica a sua identidade, poder-se-á confirmar ou negar a mesma. Deste modo, o conjunto de características é apresentado ao sistema juntamente com a alegada identidade da pessoa. «A classificação é feita entre as características biométricas de uma pessoa e as de todas as pessoas registadas no sistema»[34], encontrando assim, ou não, a pessoa que se quer identificar.

Ambas as abordagem têm pontos em comum, mas também algumas diferenças. A grande diferença é que na verificação é indicada, a *priori*, a identidade da pessoa que, alegadamente é a titular do conjunto de características, enquanto que, na identificação, o conjunto de características é o único parâmetro de entrada no sistema.

Isto faz com que o modo de verificação seja o mais indicado para utilizar quando se pretende desenvolver um sistema de autenticação, pois o que se quer garantir é a verdadeira identidade de um indivíduo mas não reconhecer o indivíduo numa determinada comunidade de registos. No método de identificação, para além de efetuar processamento desnecessário, o tempo do processo de autenticação também é muito mais demorado. Os sistemas de identificação biométrica são indicados para quando a identidade é desconhecida. Uma das grandes semelhanças é que ambos necessitam de conter o registo do indivíduo que se pretende validar ou identificar na base de dados do sistema, de outra forma não será possível fazer qualquer operação.

2.5 Desempenho Mensurável

Como classificar a performance de um sistema de autenticação biométrico? A principal medida de desempenho para sistemas de identificação é a sua capacidade para identificar o proprietário de uma assinatura biométrica. Mais especificamente, o desempenho é igual à percentagem de pedidos de informação em que a resposta é correta (2.3). Por outro lado, «o desempenho de um sistema de verificação é geralmente medido através de duas taxas de erro»[4], sendo elas:

Taxa de Falsa Aceitação (*False Rejection Rate* - FRR), também designada por «erro do tipo 1»[31], representa a percentagem de rejeitar incorretamente uma pessoa legítima, devido a alguma variação na captura de características biométricas. Este erro é frustrante para quem requer validação da identidade, fazendo com que, em caso de rejeição, o indivíduo volte a apresentar novamente as suas características ao sistema.

$$\text{FRR } (\lambda) = \frac{\text{Número de falsas rejeições}}{\text{Número de utilizadores aceites}} \tag{2.1}$$

[6]

Taxa de Falsa Rejeição (*False Acceptance Rate* - FAR), também designada por «erro tipo 2»[31], representa a percentagem de aceitação incorreta, ou seja, legitimando pessoas não autorizada. Este tipo de erro é causa de fraude.

$$\text{FAR } (\lambda) = \frac{\text{Número de falsas aceitações}}{\text{Número de impostores aceites}} \tag{2.2}$$

[6]

Uma falsa aceitação ocorre quando um sistema aprova incorretamente uma identidade e uma falsa rejeição quando um sistema nega incorretamente uma identidade. Num sistema biométrico hipoteticamente perfeito, tanto o FAR como FRR seriam zero. Infelizmente, os sistemas biométricos não são perfeitos, e o administrador deve definir o nível de segurança viável de forma a conseguir alcançar o equilíbrio entre o FAR e FRR. Se o nível de segurança é aumentado para tornar mais difícil a aceitação de impostores, consequentemente também se tornará mais propício a que, a um utilizador fidedigno, seja negado o seu acesso (FAR diminui, FRR aumenta). Por outro lado, se o nível de segurança é diminuído para que o acesso a pessoas legítimas seja negado menos vezes, também aumentará a probabilidade de indivíduos não autorizados serem validados no sistema (FAR aumenta, FRR diminui).

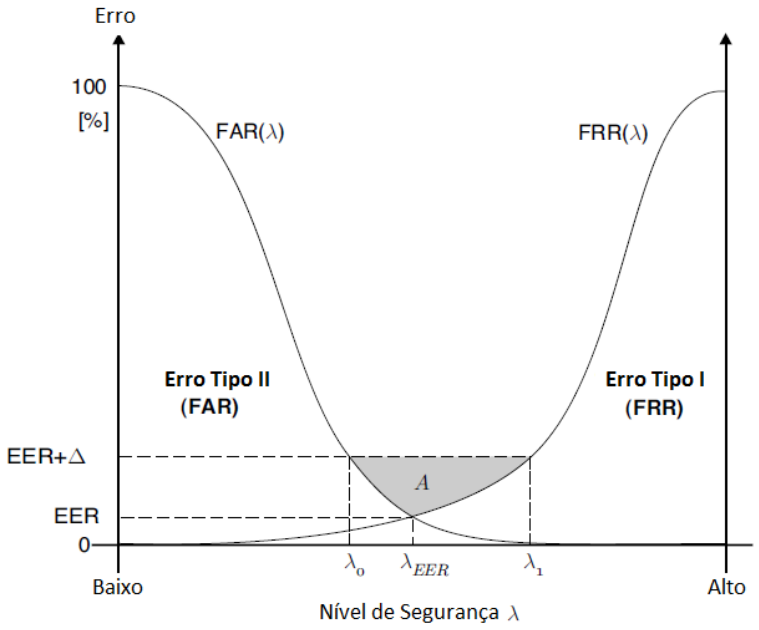


FIGURA 2.6: Dependência entre FRR e FAR.

[6]

Como se pode concluir pelo gráfico da figura 2.6 existe uma dependência entre as taxas FAR e FRR. O ponto em que estas duas curvas se cruzam é geralmente chamada de **taxa de erro igual** (*Equal Error Rate* - EER). ERR resume os valores das duas taxas num ponto de equilíbrio. Um sistema biométrico é tanto melhor

quanto menor for a ERR. A curva bidimensional que se pode ver acima designa-se por *receiver operating characteristic* (ROC)[4] e representa uma boa descrição da precisão de um sistema biométrico, sendo também útil quando se compara dois sistemas distintos.

Por fim, a percentagem que representa a precisão do sistema é calculada mediante um determinado número de amostras, como indica a seguinte equação[18]:

$$\text{Percentagem Sucesso (PS)} = \frac{\text{Número de amostras validadas}}{\text{Número amostras testadas}} \times 100 \tag{2.3}$$

[4]Esta curva é representada graficamente, traçando a taxa de falsos positivos e a taxa de falsos negativos.

Capítulo 3

Autenticação por Reconhecimento Facial

3.1 Estado da Arte

Software

A biometria, e o reconhecimento facial em particular, estão em claro crescimento, sendo considerada um tendência de futuro tecnológico. Prevê-se que a tecnologia biométrica venha dentro de alguns anos substituir por completo as *passwords* tradicionais.

Os gigantes na área tecnológica como o Facebook, Apple, Microsof e Google têm apostado no desenvolvimento de algoritmos de reconhecimento, entrando numa corrida pela obtenção da melhor fiabilidade. A Apple, neste momento, tem pendente um pedido de registo de patente que prevê uma arquitetura de reconhecimento facial[1]. O Facebook criou recentemente um novo grupo de investigação, «AI Facebook», um grupo na área da inteligência artificial que se destina ao estudo e criação de novos sistemas de forma a impulsionem o avanço tecnológico. O primeiro estudo a ser publicado foi precisamente sobre o novo *software* de verificação facial do Facebook, o DeepFace. Taigman, um dos investigados, diz mesmo que «nos aproximamos do desempenho humano»[28]. Isto pelo facto de afirmarem que o DeepFace identifica a mesma pessoa em duas fotos distintas, independentemente da iluminação ou

[1] `http://www.google.com/patents/US20150227782`

ângulo, com uma precisão de 97,25%, referindo ainda que a mesma tarefa efetuada por seres humanos tem uma precisão de 97,53%. Segundo esta publicação, «o reconhecimento facial moderno consiste em quatro etapas: deteção, alinhamento, representação e classificação»[28]. A Microsoft tem o *Project Oxford*[2], um grupo de investigação na área de reconhecimento facial, por voz, entre outros. Este grupo lançou recentemente um software que tenta determinar a idade de uma pessoa[3].

Existem ainda outras grandes empresas como a Lenovo que têm começado também a utilizar software de verificação de identidade, nomeadamente o VeriFace[4]. A Toshiba e a Fujitsu são clientes da empresa Keylemon[5], que é especializada em autenticação por reconhecimento facial. Existem outras igualmente especializadas que disponibilizam as suas API's ou *WebServices* para que os seus clientes possam rapidamente implementar um sistema de autenticação biométrica de uma forma simples.

Contudo, «o FBI tem sido líder em serviços e aplicações biométricas desde há alguns anos. O *Integrated Automated Fingerprint Identification System* (IAFIS) e o *Combined DNA Index System* (CODIS) têm servido de base a milhões de investigações nos Estados Unidos e noutros países ao longo de muitos anos»[5]. O principal objetivo deste grupo é continuar a liderar a biometria do governo dos Estados Unidos, «prestando serviços e tecnologias de qualidade para o combate ao terrorismo e auxílio nos esforços de investigação criminal»[5].

Algoritmos

Em 1991, Turk e Pentland[30] desenvolveram a técnica de *Eigenfaces*, baseada na análise de componentes principais, que permitiu efetuar reconhecimento facial de forma completamente computacional. Desde então, *Eigenfaces*, que deriva de PCA (*Principal Component Analysis*) e *Fisherfaces*[6], que deriva de LDA (*Linear Discriminant Analysis*) são das técnicas mais conhecidas em reconhecimento facial. No entanto, têm surgido imensas técnicas que advêm de modificações e melhorias de algoritmos mais tradicionais, utilizando métodos lineares e não lineares para efetuar

[2] `https://www.projectoxford.ai/`
[3] Pode ser testado em: `http://how-old.net/`
[4] `https://support.lenovo.com/us/en/videolist/ht051305`
[5] `https://www.keylemon.com/`
[6] «Uma técnica baseada na aparência, que consiste em encontrar uma combinação linear de variáveis que melhor separem duas ou mais classes»[7].

a extração de caraterísticas da face do utilizador. Um dos algoritmos lineares que mais serve de base ao aparecimento de novas técnicas é o PCA. Existem dezenas de modificações ao algoritmo, que têm como único objetivo melhorar a qualidade de reconhecimento. Surgiram então algumas técnicas a partir de modificações do PCA, como pro exemplo: o ICA (*Independent Component Analysis*), uma técnica estatística e computacional «que encontra direções ideais em vez de localizações específicas em altas densidades de dados»[3]; o Kernel-PCA, técnica que consiste numa «modificação a partir do PCA, de forma a representar mapeamentos não lineares num espaço de características de grande dimensão»[10]; o *Modular Image* PCA, que consiste numa divisão da imagem antes da extração de caraterísticas para «melhorar variações da face como expressões, iluminação e posição da cabeça»[21]; o *Multiple Similarity* PCA, que processa três matrizes de similaridade explorando as suas medidas, «matrizes onde são utilizados os vetores próprios para produzir novos subespaços»[11];

Recentemente tem surgido uma nova família de algoritmos conhecidos pela composição inversa, e que são amplamente utilizados em AAM's (*Active Appearance Models*)[2]. Desde que o algoritmo de Lucas-Kanade foi publicado em 1981[**?**], o alinhamento de imagens tornou-se uma das técnicas mais utilizadas em visão por computador. Hoje em dia, é adaptado à área do reconhecimento facial e conjugado com algoritmos de extração de caraterísticas, surgindo desta forma o TI-PCA[8], algoritmo que serviu de base para a elaboração deste trabalho.

3.2 Etapas do Sistema Biométrico

Neste capítulo serão descritas as etapas do sistema de autenticação biométrico, enquadrando estas etapas num sistema de identificação facial.

3.2.1 Aquisição de Características Biométricas (Imagem)

A aquisição de características é a primeira etapa em qualquer processo biométrico. No reconhecimento facial, deve ser obtida uma imagem que contenha a face do utilizador que se pretende autenticar, por intermédio de um dispositivo capaz de efetuar a recolha. Neste trabalho a captação da imagem é feita pelo *browser*, através de um qualquer componente de vídeo disponível. A imagem deve conter apenas o

rosto de uma pessoa pois, no caso de existirem vários, ou não existir nenhum, o processo de verificação ou registo será cancelado.

Esta etapa, apesar de se limitar à obtenção de imagens, deve envolver alguns cuidados na hora da recolha. O sistema deve estabelecer um conjunto de regras que o utilizador deve prezar aquando da captura da imagem. Ainda que exista a fase de pré-processamento, quanto mais normalizada estiver a imagem no momento da aquisição, melhores resultados obterá o sistema, diminuindo o FAR.

Desta forma, devem ser tomadas em conta algumas boas práticas, no que diz respeito ao utilizador e ao seu meio envolvente no momento da aquisição de características, tais como:

Meio envolvente

Qualidade do Equipamento: A câmara deve ter preferencialmente uma resolução acima dos 640 x 480.

Iluminação: É de evitar apontar a câmara para luzes brilhantes, ou qualquer área com pontos fortes de luz. A iluminação deve ser moderada e não deve incidir diretamente na câmara.

Distância: A distância da câmara ao utilizador varia consoante a resolução da mesma. Não obstante, a imagem deve ser uma fotografia tipo passe.

Focagem: A imagem deve estar focada, ajustando-se à distância a que o utilizador se encontre.

Local: A câmara deve ser posicionada num local onde não existam grandes quantidades de ruído, se possível, direcionada para uma parede ou tela branca.

Utilizador

Posicionamento: A face deve permanecer de forma frontal à câmara.

Expressões: A expressão do rosto deve manter-se neutra, evitar sorrisos rasgados, fechar os olhos, óculos escuros ou quaisquer acessórios que possam tapar a cara.

Desta forma, consegue-se ter um ambiente controlado que facilitará as tarefas do sistema. Ainda de salientar que o rigor na aquisição das imagens deve ser tomado em conta tanto no processo de registo como no de autenticação, uma vez que a decisão depende de uma comparação entre ambas.

3.2.2 Pré Processamento

Esta etapa estabelece a ponte entre a aquisição de características e a extração das mesmas. Nas imagens que dão entrada no sistema, é feita a deteção e recorte do rosto, descartando o restante, normalizar a sua dimensão e melhorar questões como a fraca ou demasiada iluminação.

3.2.2.1 Deteção da Face na Imagem

A deteção de face na imagem recolhida faz parte da fase de pré-processamento. É nesta etapa que é eliminado todo o ruído da imagem, ou seja, toda a informação (pixeis) desnecessária. Desta forma, é extraída uma nova imagem apenas com o recorte da face do indivíduo. Isto faz com que, para além da imagem ficar com menos informação supérflua, diminua também a quantidade de pixeis, fazendo com que não seja necessário tanto processamento e tenha consequentemente uma melhor performance.

Para detetar as faces nas imagens foi utilizado o algoritmo *Viola-Jones*, «uma técnica de deteção de objetos em imagens que se baseia em três conceitos: integral de imagem, treino de classificadores usando *boosting* e o uso de classificadores em cascata»[35]. Embora o algoritmo possa ser utilizado para reconhecer qualquer objeto, a motivação principal de Viola e Jones foi o reconhecimento facial. Um dos pontos fortes deste algoritmo é a rapidez com que é executado.

Por conseguinte, ao detetar o rosto na imagem, obtêm-se as coordenadas da sua localização e é feito o recorte e gerada uma nova imagem exclusivamente com a face do indivíduo, como ilustra a figura 3.1.

3.2.2.2 Normalização e Filtragem da Imagem

Um dos primeiros fatores a ter em conta nesta etapa é o facto de a imagem só poder conter uma face. Qualquer imagem em que, no processo de deteção, sejam

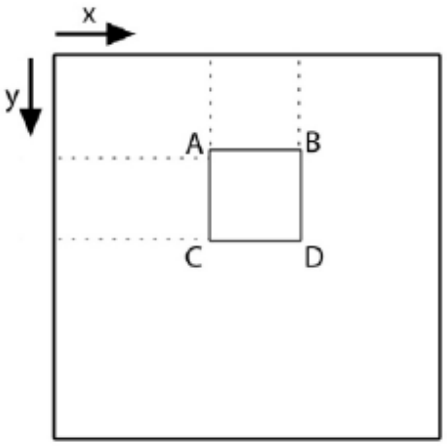


FIGURA 3.1: Recorde da face na imagem.

encontradas duas ou mais faces, este será imediatamente interrompido, informando a causa. O mesmo se aplica no caso de não ser detetada qualquer face na imagem.

Desta forma, ao extrair o recorte da única face da imagem, o passo seguinte é normalizá-la e aperfeiçoá-la, antes de se proceder à extração de características. A normalização consiste em redimensionar a imagem para que tenha a dimensão definida pelo sistema. O aperfeiçoamento da imagem recai sobre um dado muito debatido quando se fala em reconhecimento facial, a iluminação, visto que, «mudanças na iluminação originam uma descida drástica na performance de um sistema de reconhecimento facial»[27]. Para contornar esta dificuldade, foi utilizado um método de processamento de imagem, chamado *Histogram Equalization*, que consiste numa minimização automática do contraste em áreas com demasiada luz ou demasiado escuras. No entanto, não deve ser feita uma equalização de histograma considerando o contraste global da imagem pois, desta forma, pode-se ficar com zonas muito brancas e outras demasiado escuras, perdendo assim informação relevante. Sendo assim, uma forma de solucionar este problema é limitar o histograma a uma região específica aplicando o método *Adaptive Histogram Equalization*[23]. Deste modo, a imagem é dividida em pequenos blocos, dividindo-os em dimensões de três por três, que são assim equalizados separadamente.

3.2.3 Alinhamento e Extração de Características

Após o pré-processamento, a imagem da face normalizada segue para a extração de características, com o objetivo de encontrar as principais características a serem

utilizadas na etapa final, a decisão. Devido à grandeza dos vetores (imagens), é utilizada a técnica de PCA, visando reduzir a dimensionalidade da imagem a fim de aliviar o custo computacional e melhorar a precisão do classificador. Além da extração de características nesta etapa, é feito também um alinhamento das imagens que dão entrada no sistema. O alinhamento é feito pelo algoritmo SIC, que encontra os valores que melhor definem a transformação dos pontos que definem a localização da face. Assim sendo, a utilização destes dois algoritmos dá origem ao TI-PCA, algoritmo estudado no âmbito desta dissertação, que se baseia numa extração de características de faces devidamente alinhadas. O TI-PCA «alternadamente alinha as imagens, de forma a construir um *Eigenspace* melhorado, o objetivo é minimizar o erro quadrático médio entre as imagens alinhadas e as suas reconstruções»[8]. O alinhamento é feito pelo algoritmo SIC e são ainda alinhadas com base na localização dos olhos, tendo em conta que a distância entre cada olho e a sua margem mais próxima deve ser igual. Os fundamentos matemáticos dos algoritmos aqui mencionados são detalhados e explicados pormenorizadamente no próximo capítulo (4), dando a conhecer os métodos utilizados e o seu fundamento teórico.

Na fase de registo é feita uma extração das características antes e depois de alinhar todas as imagens base recolhidas. Na fase de autenticação são utilizadas as características já extraídas na fase de registo, alinhando apenas a imagem e realizando a etapa de decisão.

3.2.4 Decisão/Classificação

A decisão final num processo de autenticação é tomada com base nas características principais encontradas no processo de registo e a imagem de prova recolhida na etapa de aquisição de características. Esta etapa é feita em função do cálculo da distância euclidiana entre a imagem de prova e reconstrução da imagem da face do utilizador que pretende validar a sua identidade, obtendo assim um erro que determina a sua diferença. Como tal, deve existir um valor limite que defina a aceitação do erro uma vez que, caso o erro seja inferior ao limite, o pedido de validação será aceite, caso contrário, será negado. Este valor conhecido como *threshold* é definido e explicada a sua utilização no próximo capítulo (4).

Capítulo 4

Fundamentos Matemáticos

4.1 Análise de Componentes Principais

A análise de componentes principais (*Principal Component Analysis*) é uma técnica matemática de redução de dimensionalidade de um conjunto de dados. Inventada por Karl Pearson[20], esta técnica baseia-se na extração de componentes principais de um espaço multidimensional, formando assim um novo conjunto. É muito usada em reconhecimento de padrões para eliminar redundâncias de informação mantendo, no entanto, as principais características de um padrão. Mais tarde, Turk e Pentland[30] utilizaram esta técnica para extração de características e representação de faces.

No reconhecimento facial, a técnica PCA é utilizada para encontrar um grupo de subespaços de vetores, a que se dá o nome de *Eigenfaces*. «*Eigenfaces* são um conjunto de vetores próprios obtidos a partir de uma matriz covariância»[30]. Desta forma, a face de qualquer indivíduo pode ser representada por uma combinação linear de vetores próprios, fazendo com que um conjunto de imagens represente um conjunto de pontos num espaço enorme.

PCA consiste em promover uma transformação linear nos dados de modo que os dados resultantes dessa transformação tenham as componentes mais significativas nas suas primeiras dimensões, em eixos denominados principais. Representar graficamente o vetor correspondente a uma face é algo humanamente impossível, portanto, para uma melhor perceção, foi feita uma analogia num gráfico a duas dimensões. As figuras abaixo (Figuras 4.1, 4.2) ilustram um conjunto bidimensional, após a aplicação do PCA e a projeção dos dados utilizando apenas a primeira componente principal.

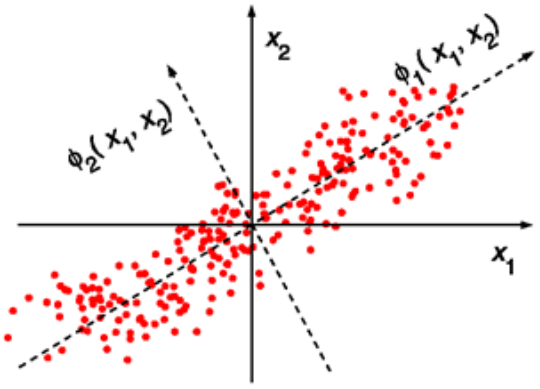


FIGURA 4.1: Análise de Componentes Principais aplicada a um conjunto de dados, onde as linhas tracejadas são as componentes principais.

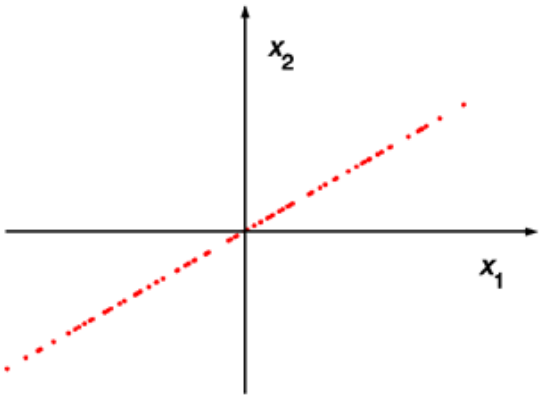


FIGURA 4.2: Projeção dos dados usando apenas a primeira componente principal.

4.1.1 Cálculo *Eigenfaces*:

A imagem de uma face designada por $I(x, y)$ deve ser uma matriz bidimensional com a dimensão de h colunas por w linhas com uma quantização de 8 bit de níveis de cinzento. Cada imagem deve ser convertida num vetor de dimensão M, de modo que uma imagem de 128×128 se torne num vetor de dimensão $16,384$. Sendo assim, cada imagem (vetor) representa um ponto num espaço vetorial.

Portanto, a explicação deste algoritmo foi decomposta em seis passos, para uma melhor compreensão.

Primeiro Passo:

As imagens de entrada originais da face de um utilizador, também denominadas como imagens de treino, são designadas por I_1, I_2, ..., I_N, onde N é o número de imagens. Estas imagens devem ser transformadas em vetores Γ_i, de dimensão M:

Segundo Passo:

Obter um conjunto S de dimensão N que contém todos os vetores Γ_i correspondentes às imagens das faces.

$$S = [\Gamma_1, \Gamma_2, \Gamma_3, ..., \Gamma_N\ ] \tag{4.1}$$

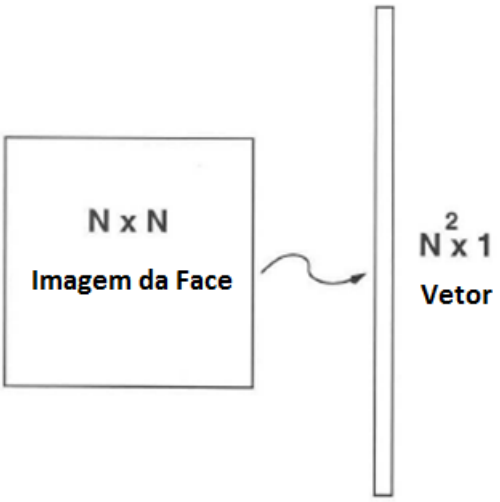


FIGURA 4.3: Representação da imagem I_i como vetor Γ_i

Terceiro Passo:

Obter a média de todas as imagens do conjunto, que deve ser representada por [30]:

$$\Psi = \left(\frac{1}{N}\right) \sum_{i=1}^{N} \Gamma_i \tag{4.2}$$

Como podemos observar na figura 4.4 foi utilizado um conjunto de imagens para gerar uma face média.

FIGURA 4.4: Face Média obtida de um conjunto de imagens

Quarto Passo:

Deve ser encontrada a diferença entre as imagens de treino e a média de todas as imagens.

$$\Phi_i = \Gamma_i - \Psi \tag{4.3}$$

Depois, deve ser formado um conjunto A, com todas essas normalizações, onde N é o número de imagens normalizadas.

$$A = \{\Phi_1, \Phi_2, ..., \Phi_N\} \tag{4.4}$$

Quinto Passo:

Após ter a matriz A, deve ser calculada a sua matriz de covariância, para posteriormente conseguir obter os vetores e valores próprios. Esta matriz obtém-se por [30]:

$$C = \frac{1}{N}\sum_{n=1}^{N} \Phi_n \Phi_n^T = AA^T \tag{4.5}$$

Sexto Passo:

Neste passo devem ser encontrados os vetores próprios u_i da matriz covariância, através do método:

$$AA^T u_i = \lambda_i u_i \tag{4.6}$$

No entanto, a matriz covariância tem dimensões de $M \times M$ e nos casos em que a dimensão do vetor (M) é menor que o número de imagens de treino (N), pode-se utilizar a matriz C. Não obstante, para os casos em que $M > N$, encontrar os vetores próprios a partir da matriz C é uma tarefa computacionalmente muito dispendiosa. Como forma de diminuir a sua dimensão, podemos construir a matriz $L = A^T A$ de dimensão $N * N$, considerando os seus vetores próprios v_i, tais que:

$$A^T A v_i = \mu_i v_i \tag{4.7}$$

Qual é a relação entre u_i e v_i?

$$\begin{pmatrix} A^T A v_i = \mu_i v_i \Rightarrow AA^T A v_i = \mu_i A v_u \Rightarrow \\ CAv_i = \mu_i Av_i \text{ ou } Cu_i = \mu_i u_i \text{ onde } u_i = Av_i \end{pmatrix} \tag{4.8}$$

Esta dedução permite perceber que AA^T e A^TA têm os mesmos valores próprios e os seus vetores próprios estão ligados[1]. Portanto, podemos calcular a matriz L e encontrar N vetores próprios, v_i de L. Calculados os vetores próprios pode-se então definir a matriz u_i que contém os *Eigenfaces* por [30]:

$$u_i = Av_i \tag{4.9}$$

Ao obter a matriz u_i, se se representar cada linha como uma imagem, percebe-se que estas apresentam espectros de faces humanas(Fig. 4.5), fazendo realçar as características principais do conjunto de imagens de treino.

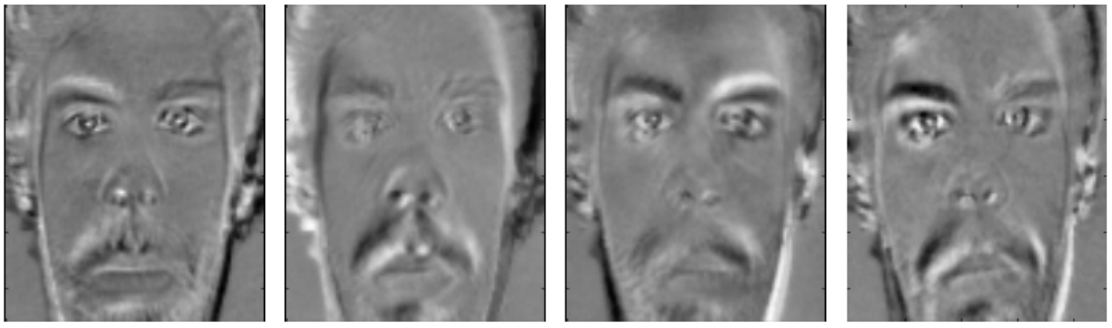


FIGURA 4.5: Primeiras 12 *Eigenfaces* mais relevantes.

A implementação do algoritmo, que executa todas as etapas acima descritas, encontra-se em 3, A1.

Análise Final

Sabe-se então que os valores próprios permitem-nos classificar os vetores próprios de acordo com a sua relevância, permitindo assim que se consiga definir um subconjunto de vetores próprios mais significativos. Quanto à escolha de quantos vetores próprios(*Eigenfaces*) devem ser utilizados, a decisão envolve um balanceamento entre a precisão de reconhecimento e o tempo de processamento. Cada *Eigenface* adicional contribui para o aumento do processamento do sistema.

Podemos concluir que o método PCA diminui consideravelmente os esforços computacionais porque reduz a ordem da matriz, possibilitando melhor eficiência. No entanto, uma das grandes desvantagens da utilização deste método é que a análise

[1]É importante referir que os N valores próprios de A^TA corresponde aos M melhores valores próprios de AA^T.

em si não garante um bom poder discriminativo, uma vez que as componentes são encontradas com base na direção da maior variância do conjunto, e não pela maior distinção. Por isso, pode-se dizer que a análise de componentes principais caracteriza bem a face, mas não funciona necessariamente como um bom classificador porque nem sempre as dimensões de maior variância são as mais relevantes para discriminar diferentes identidades.

4.2 Transformação Invariante PCA

No enquadramento dos desenvolvimentos levados a cabo por «Kirby e Sirovich, a imagem de uma face humana arbitrária pode ser obtida partindo de uma imagem média e um conjunto de imagens base (*basis images*)»[8].

Neste processo tomemos a representação vectorial de uma imagem como sendo um vetor coluna onde cada coordenada representa a *pixel coordinate* do pixel que lhe está associado. Esta representação vetorial permite um mais eficiente tratamento computacional dos dados. Designando por $I \in \mathbb{R}^d$ uma dada imagem de face humana, a mesma pode obter-se através da seguinte construção

$$I = \mu + \sum_{J=1}^{m} a_j \phi_j + e \tag{4.10}$$

onde μ representa o vetor de coordenadas da imagem média, $\phi_1, \phi_2, ..., \phi_m$ são imagens base obtidas a partir do conjunto de imagens de treino previamente capturadas e e representa a componente errática. Ainda na mesma formulação, podemos observar a existência de um conjunto de parâmetros $a_1, a_2, ..., a_m$ que podem ser obtidos através de uma projeção ortogonal num espaço linear que doravante designamos por *face space*. Este espaço pode ser construído usando os vetores próprios (*Eigenfaces*) associados aos M maiores valores próprios (*Eigenvalues*) através de PCA aplicada às imagens de treino[8].

De uma forma mais rigorosa, dado um conjunto de imagens de treino I^i, a obtenção dos *Eigenfaces* pode ser feita através do «método de minimização dos erros quadrados entre a imagem original e a sua reconstrução»[8].

$$arg \min_{\mu,\phi_j} \frac{1}{N} \sum_{i=1}^{N} \left(\min_{a} \left\| I^i - \left(\mu + \sum_{J=1}^{m} a_j \phi_j \right) \right\|^2 \right) \tag{4.11}$$

No cenário improvável de obtenção de imagens de face originalmente alinhadas, a computação seria feita com recurso às fórmulas anteriores, nomeadamente através do algoritmo PCA[2]. Contudo, devido à imprevisibilidade da captura da imagem da face e das diferentes feições humanas, o uso dos *Eigenfaces* seria enviesado por muitos tipos de desvio em relação à posição de alinhamento (por exemplo rotações, translações e escalamento). Neste caso, o método careceria de robustez para o reconhecimento de faces. Em que consiste o alinhamento de imagens? ≪O alinhamento de imagens consiste em mover e possivelmente deformar uma imagem modelo de forma a minimizar a diferença entre a imagem modelo e a imagem a alinhar≫[1].

Neste sentido, é proposto o uso do *Transform-Invariant PCA* (TIPCA). Este método assenta no mesmo princípio que o anterior mas oferece maior resistência a eventuais perturbações nas imagens originais. Esta robustez é atingida pela intercalação do método PCA com o realinhamento da imagem. Com o objectivo de acomodar no modelo a possibilidade de perturbações, vamos introduzir a *warp function* $W(x;p)$ que representa um conjunto parametrizado de possíveis transformações ao alinhamento correto de uma imagem. Esta função faz a correspondência entre um pixel x na imagem base e a localização do pixel na imagem original de acordo com o conjunto de parâmetros p. No presente trabalho foi utilizada uma matriz de transformação que consiste numa combinação entre a translação[3] e o cisalhamento vertical e horizontal[4], dando origem à seguinte matriz[5]:

$$W = \begin{bmatrix} 1 & -s & t_x \\ s & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \tag{4.12}$$

onde p= (s, t_x, t_y), cuja obtenção será explicada abaixo.

A imagem seguinte ajuda a visualizar o que acontece com a aplicação da matriz W nos pontos que indicam a localização do rosto.

[2]Este método é frequentemente utilizado após o realinhamento da imagem com recurso a outros tipos de técnicas, como por exemplo o realinhamento de olhos.

[3]Translação ou *translation* é a função que move todos os pontos com uma distância constante numa direção especifica

[4]Cisalhamento ou *shear*, um mapa linear que desloca cada ponto numa direção fixa, por um valor proporcional à sua distância a partir da linha que é paralela à direção.

[5]Arthur Coste, ≪Affine Transformation, Landmarks registration, Non linear Warping≫, CS6640: Image Processing, Project 3, 2012, pag. 15-20.

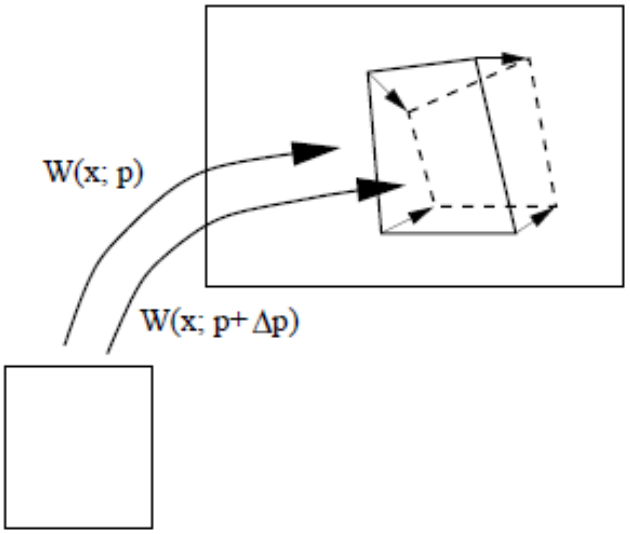


FIGURA 4.6: Transformação aplicada à imagem.

Para efeitos de notação representou-se por $I^i(x)$ as imagens de treino originais e por $\phi_j(x)$ as imagens base. Dado um conjunto de N imagens de treino, procura-se um conjunto de imagens base que minimizam a soma da distância entre as imagens originais transformadas e o *facespace*, ou seja, as suas reconstruções.

$$arg\min_{\mu,\phi_j} \frac{1}{N}\sum_{i=1}^{N}\left(\min_{p^i,a^i}\left\|\underbrace{I^i\left(W(x;p^i)\right)}_{\text{imagem transformada}} - \underbrace{\left(\mu+\sum_{J=1}^{m}a_j\phi_j\right)}_{\text{reconstrução}}\right\|^2\right) \tag{4.13}$$

Usando uma minimização simultânea dupla, este método pretende obter um alinhamento do *Eigenspace* ao mesmo tempo que as imagens vão sendo realinhadas de acordo com o *Eigenspace* resultante de cada iteração. Uma vez que neste método as imagens sofrem transformações a cada iteração, a minimização da anterior expressão requer um processo de optimização iterativa. Antes de detalharmos o algoritmo por detrás da otimização, vamos apresentar o fio condutor do processo. Em traços gerais, este otimiza os valores de $\{p^i, a^i\}$ assumindo um determinado valor para os parâmetros $\{\mu, \phi_j\}$ e depois, alternativamente, usa o valor encontrado de $\{p^i, a^i\}$ para chegar a um valor ótimo temporário de $\{\mu, \phi_j\}$.

Como ponto de partida para a cadeia de iterações, usamos o *Eigenspace* $\{\mu^0, \phi_j^0\}$ resultante da aplicação de PCA às imagens de treino (não alinhadas). Na parte que

se segue, explicar-se-á com detalhe o algoritmo usado em cada iteração.

Passo 1) Alinhamento do *Eigenface*

Neste passo começa-se por tomar o valor da imagem média e das imagens base $\{\mu, \phi_j\}$ como fixo, e otimiza-se a função em termos do par coordenado $\{p^i, a^i\}$. O algoritmo utilizado é o *Simultaneous Inverse Compositional algorithm* (SIC). Este algoritmo realiza uma *Gaussian-Newton gradient descent optimization*, em termos dos parâmetros a^i e p^i[1].

Na especificação deste modelo, a imagem média e as imagens base serão atualizadas de acordo com a *warp function* que, por sua vez sofrerá transformações após a atualização do par coordenado $\{p^i, a^i\}$. Desta forma, a anterior equação deverá ser reformulada para contemplar o efeito da transformação na imagem média e imagens base:

$$\sum_x \left(I^i\left(W(x;p^i)\right) - \mu\left(W(x,\nabla p)\right) - \sum_{J=1}^{m}(a_j + \Delta a_j)\phi_j\left(W(x,\nabla p)\right)\right)^2 \tag{4.14}$$

Realizando uma aproximação de Taylor[1] de primeira ordem em $\mu\left(W(x,\Delta p)\right)$ e $\phi_j\left(W(x,\Delta p)\right)$, obtemos:

$$\sum_x \left(I^i\left(W(x;p^i)\right) - \left(\mu(x) + \nabla\mu\frac{\partial W}{\partial p}\Delta p\right) - \sum_{J=1}^{m}(a_j + \Delta a_j)\left(\phi_j(x) + \nabla\phi_j\frac{\partial W}{\partial p}\Delta p\right)\right)^2 \tag{4.15}$$

Se ignorarmos os termos de segunda ordem a expressão anterior pode ser simplificada para:

$$\sum_x \left(\underbrace{I^i\left(W(x;p^i)\right) - \mu(x) - \sum_{J=1}^{m} a_j\phi_j(x)}_{E(x)=\text{ erro da transformação}} - \left(\nabla\mu + \sum_{J=1}^{m} a_j\nabla\phi_j\right)\frac{\partial W}{\partial p}\Delta p - \sum_{J=1}^{m}\phi_j(x)\Delta a_j\right)^2 \tag{4.16}$$

Da anterior expressão podemos retirar as $n + m$ *dimensional steepest-descent images*, que é descrita na sua forma vetorial SD:

$$SD = \left(\left(\nabla\mu + \sum_{J=1}^{m} a_j\nabla\phi_j\right)\frac{\partial W}{\partial p_1}, ..., \left(\nabla\mu + \sum_{J=1}^{m} a_j\nabla\phi_j\right)\frac{\partial W}{\partial p_n}, \phi_1(x), ..., \phi_m(x)\right) \tag{4.17}$$

Para efeitos de notação, tomemos as seguintes duas simplificações:

$$q = \begin{pmatrix} p \\ a \end{pmatrix} \text{ e } \Delta q = \begin{pmatrix} \Delta p \\ \Delta a \end{pmatrix} \tag{4.18}$$

A expressão (4.16) pode simplificar-se usando o erro da transformação e as anteriores simplificações como:

$$\sum_x \left[E(x) - SD(x)\Delta q\right]^2 \tag{4.19}$$

Sendo o erro de transformação definido por:

$$E(x) \;=\; I\left(W(x;p)\right) - \mu(x) - \sum_{J=1}^{m} a_j \phi_j(x) \tag{4.20}$$

Como demonstrado por Baker, Gross e Matthews[1], a expressão atinge o seu mínimo em:

$$arg\min_{\Delta q} \sum_x \left[E(x) - SD(x)\Delta q\right]^2 = -H^{-1} \sum_{i=1}^{N} SD^T(x)E(x) \tag{4.21}$$

onde H^{-1} representa a inversa da matriz Hessiana:

$$H^{-1} = \sum_x SD^T(x)SD(x) \tag{4.22}$$

Com a expressão anterior, após a realização das computações intermédias, obtêm-se os parâmetros atualizados para Δq, de onde podem ser extraídos ambos Δp e Δa. Estes, por sua vez, serão atualizados na atualização da *warp function* $W(x;p) \leftarrow W(x;p)_{-1} \circ W(x;\Delta p)^{-1}$ e dos parâmetros de aparência $a_i \leftarrow a_{i,-1} + \Delta a_i$, onde o índice -1 remete para os valores da iteração anterior.

Uma vez que o vetor de *steepest descent images* depende dos valores dos parâmetros a_j que sofrem alterações após cada iteração, este vetor tem de ser continuamente recalculado, bem como as computações seguintes.

Este *step-list*[26] determina de uma forma sumária as várias etapas do algoritmo SIC. A sua implementação, foi desenvolvida em duas linguagens, Python e C++, e encontra-se em 3, A3 e A5.

Pré-cálculo:

(3) Calcular os gradientes $\triangle\mu$ e $\triangle\phi_j$ para $j = 1, \ldots, m$

(4) Calcular a Jacobiana $\frac{\partial W}{\partial p}$ (x; 0)

Iterar:

(1) Warp I com $W(x;p)$ para calcular $I(W(x;p))$.

(2) Calcular o erro de transformação da imagem $E(x)$ usando a equação (4.20)

(5) Calcular as *steepest descent images* $SD(x)$ usando a equação (4.17)

(6) Calcular a matriz hessiana invertida H^{-1} usando a equação (4.22)

(7) Calcular $\sum_x SD^T(x)E(x)$

(8) Calcular $\triangle q = -H^{-1}\sum_x SD^T(x)E(x)$

(9) Atualizar $a_i \leftarrow a_i + \triangle a_i$

Até $\|\triangle p\| \leq \varepsilon$

Passo 2) Atualizar o Eigenface

Repare-se que neste momento temos um novo valor para o par de coordenadas $\{p^i, a^i\}$, pelo que, por aplicação da *warp function*, levou-se a cabo uma nova transformação das imagens de treino. Com as novas imagens, pode-se obter facilmente o *Eigenspace* atualizado através da aplicação de PCA *standard* a estas, exatamente como foi feito no conjunto de imagens inicial, no princípio do processo. Com o novo par de coordenadas $\{\mu, \phi_j\}$, retorna-se ao passo anterior para mais uma iteração.

Após a aplicação deste processo iterativo, a convergência deverá ser atingida e o valor da função diminuirá iteração após iteração. Quando este último parar de reduzir, o processo de convergência chegou ao fim, pelo que os valores $\{\mu, \phi_j\}$ encontrados deverão ser tomados como confiáveis para as fases posteriores do processo de reconhecimento facial[8].

A implementação do algoritmo TIPCA, que se baseia na interação entre a extração de características e o alinhamento, encontra-se em 3.

4.3 Processo de Decisão/Classificação

Como dito anteriormente, ao representar os *Eigenfaces* como imagens ver-se-á que os mesmos apresentam figuras semelhantes a faces humanas, ou seja, imagens contendo os componentes principais das faces humanas. *Eigenfaces* com menores valores próprios apresentam imagens com poucas características interessantes, por isso muitas vezes são descartadas e são utilizadas apenas as que têm melhores valores próprios.

Neste trabalho é desenvolvido o processo de verificação. Isto quer dizer que o processo de classificação é feito utilizando as imagens de treino armazenadas na base de dados, relativos ao utilizador que se pretende autenticar. Portanto, a projeção é feita de forma a obter Ω_j, em que deve ser utilizado o conjunto A(4.4), da seguinte forma [30]:

$$\omega_j = u_k^T \ (A \ - \ \Psi) \tag{4.23}$$

obtendo assim,

$$\Omega_j = [\omega_1, \omega_2, ..., \omega_N] \tag{4.24}$$

em que N é o número de vetores próprios utilizados para a projeção do espaço.

De seguida deve-se achar também Ω_k que corresponde à projeção da imagem de prova:

$$\omega_k = u_k^T \ (\Gamma \ - \ \Psi) \tag{4.25}$$

obtendo desta forma,

$$\Omega_k = [\omega_1, \omega_2, ..., \omega_N] \tag{4.26}$$

A imagem de prova, à semelhança das imagens de treino, é projetada no espaço, como demostrado anteriormente, traçando Ω_k e Ω_j, que contêm os chamados pesos(*weights*)[29]. Utilizando os pesos de ambas as projeções, deve ser encontrada a distância entre as duas imagens, ou seja, o erro e a medição deve ser calculada através do método da distância euclidiana:

$$\varepsilon_k = \|\Omega_j - \Omega_k\| \tag{4.27}$$

Por fim, se o erro, ε_k, for menor que o valor limiar(*threshold*) representado por Θ, é validada a identificação do utilizador com sucesso. A respetiva implementação

do processo de decisão encontra-se em 3, A4.

$$\varepsilon_k < \Theta_c \tag{4.28}$$

Definindo *threshold*

Para definir o limiar de aceitação a fim de posterior decisão, deve-se calcular Θ_c, que pode ser definido pela maior distância aceitável entre Ω_k e Ω_j calculados anteriormente[16]:

$$\Theta_c \;=\; \frac{1}{2} \max_{j,k}^{n} \{\Omega_j - \Omega_k\} \tag{4.29}$$

No entanto, esta equação seria válida se se tratasse de um processo de identificação, e não de verificação[6]. Quer isto dizer que num processo de identificação poderia ser calculada a distância euclidiana entre a imagem prova e todos os registos, identificando o registo detentor do menor erro diante todos. No caso da verificação, tratada neste trabalho, o facto de conhecer previamente o utilizador exclui a necessidade de calcular a distância entre todos os registos, devendo ser calculada apenas a distância relativa ao registo que corresponde à identidade do indivíduo. Sendo assim, o erro obtido terá de ser submetido a um limite que será, neste caso, definido pelo administrador do sistema de autenticação. Um *threshold* baixo será propício à existência de menos falsos positivos mas também a dificuldade na aceitação de utilizadores legítimos; valores altos permitem uma aceitação mais fácil a utilizadores legítimos mas também o aumento de falsos positivos, ou seja, validações erradas.

[6] A diferença entre ambos já foi debatida anteriormente na secção 2.5

Capítulo 5

Arquitetura do Sistema

O processo de autenticação deste sistema biométrico é constituído por várias etapas, como se pôde perceber anteriormente (3.2). Estas etapas foram encaixadas numa arquitetura desenhada com vista a obter um servidor de autenticação em rede.

A arquitetura presente, representa uma plataforma web constituída por um servidor que disponibiliza HTML e que, através de um *web browser*, permite ao utilizador dispor, caso exista, da componente de vídeo presente no dispositivo utilizado, seja ele qual for. Este sistema incide essencialmente em dois casos de uso, o registo e a autenticação. Sendo assim, as figuras **??** e **??** exibem o fluxo do sistema em ambas as fases, demonstrando assim, de uma forma mais clara, a arquitetura do sistema.

5.1 Casos de Uso

5.1.1 Registo

Subentendendo o pedido inicial da página (HTML) ao servidor, através de um *HTTP Request GET*, pode perceber-se, observando a arquitetura, presente em anexo (3, B1), que o processo de registo tem origem numa ação externa ao sistema, que provém da parte de um utilizador. Para dar início ao processo, o utilizador deve deter uma câmara de vídeo no dispositivo utilizado para que seja feita a recolha de imagens de modo a conseguir efetuar a etapa de aquisição de características, que carece também de alguma informação sobre o utilizador. Depois disso, todos os dados adquiridos são enviados pela rede através do protocolo HTTP para o servidor, que começará por pré processar as imagens de forma a verificar a sua veracidade, garantindo que existe

uma e apenas uma face em cada imagem[1]. Segue-se a utilização do algoritmo TI-PCA que é constituído pelos algoritmos[2] PCA e SIC, este têm como objetivo alinhar os rostos das imagens e extrair os componentes principais dos rostos, respetivamente. As imagens, já alinhadas, são armazenadas em base de dados e, por fim, é enviado o *feedback* da operação para o *browser* para que este possa apresentar o resultado ao utilizador.

5.1.2 Autenticação

O processo de autenticação segue um fluxo semelhante ao do processo de registo. A autenticação inicia-se tal como o registo, através de uma iniciativa externa por parte do utilizador. A primeira etapa, aquisição de características, é responsável por recolher o e-mail e uma imagem com a face do utilizador. O e-mail servirá para identificar o seu registo entre os demais, pois pelo facto de se tratar de um sistema de verificação requer uma identificação prévia. Ao ser feito o pedido de autenticação ao servidor, a imagem é inicialmente pré processada, seguindo para o alinhamento e extração de características, utilizando as imagens provenientes do processo de registo (imagens de treino) para proceder à execução do algoritmo. Após efetuado o alinhamento e obtidas as componentes principais, toma lugar a etapa de decisão, onde será comparada a imagem de prova e imagens de treino do respetivo utilizador, concebendo uma decisão[3] com base nos limites estabelecidos. Por fim é enviada a decisão ao *browser*, que tratará de apresentar o resultado ao utilizador. O diagrama respetivo a este caso de uso encontra-se em anexos (3, B2).

5.1.3 Conclusões

Como pudemos constatar, ambos os casos de uso são bastante similares, assentando as suas diferenças no facto de um inserir e o outro consultar na base de dados, um efetua uma tomada de decisão e o outro não, um efetua o alinhamento de todas as imagens de treino enquanto o outro alinha apenas a imagem de prova.

Nos diagramas pudemos também observar uma divisão de tarefas entre o *Web Browser* e *Web Server* que são, neste trabalho, apontados como o «Cliente» e

[1] Na secção 3.2.2 deste trabalho, é explicado tudo o que é feito no processo de pré processamento.
[2] Na secção 4.1 encontram-se detalhadamente descritos ambos os algoritmos.
[3] Na secção 3.2.4 deste trabalho, é explicado como é feita a tomada de decisão.

«Servidor» do sistema, respetivamente. Ambos serão abordados nas próximas duas secções, explicando o seu funcionamento e as tecnologias utilizadas.

5.2 Cliente (*Web Browser*)

Cliente é a designação utilizada para descrever a componente responsável pela interação com o utilizador. Este é o *web browser*, responsável por apresentar o HTML fornecido pelo *web server*. Foi elaborada uma estrutura em HTML, um protótipo simples que permite demonstrar o funcionamento do sistema. O ficheiro HTML disponibilizado pelo servidor contém código JavaScript que utiliza a *getUserMedia API*[4], para conseguir aceder e utilizar a câmara do dispositivo, notando que o *browser* utilizado deve suportar HTML5.

Deste modo, o utilizador poderá proceder aos dois casos de uso utilizando os seguintes links `IP_SERVIDOR:PORTA_SERVIDOR/enrollment/` para realizar o registo e `IP_SERVIDOR:PORTA_SERVIDOR/autentication/` para a autenticação[5]. Em ambos deve tirar fotografias à sua face respeitando as boas práticas descritas em 3.2.1 e enviá-las para o servidor carregando no botão de registo ou autenticação, respetivamente.

As imagens abaixo (5.1, 5.2) demonstram o que é apresentado ao utilizador no momento em que este, enquanto cliente, acede ao servidor através do *browser*, em ambos os casos de uso.

[4]Tutorial Capturing Audio and Video in HTML5, publicado por Eric Bidelman em 2012, `http://www.html5rocks.com/pt/tutorials/getusermedia/intro/`

[5]IP_SERVIDOR, é a variável que corresponde ao IP no qual foi executado o *web server* da *framework Django*. A variável PORTA_SERVIDOR deve indicar a porta no qual o *web server* está à escuta.

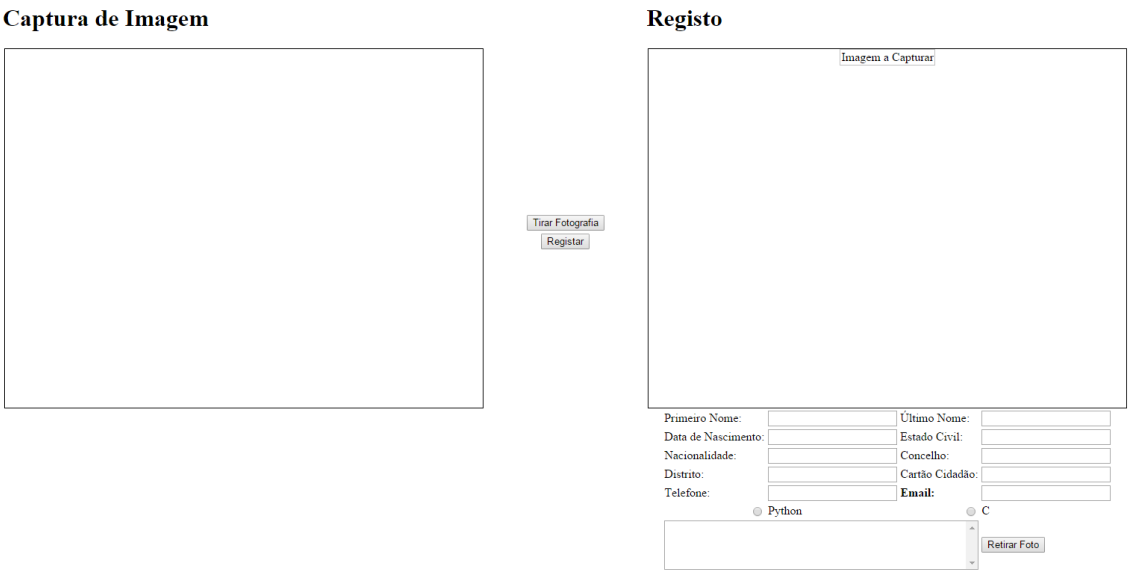


FIGURA 5.1: Interface de Registo.

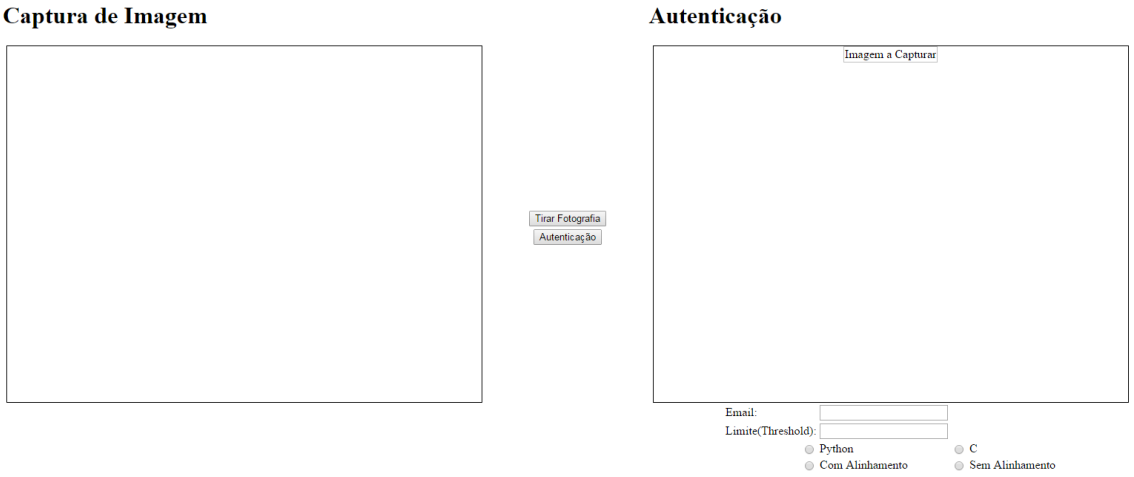


FIGURA 5.2: Interface de Autenticação.

Portanto, as principais funções do cliente são: efetuar a aquisição da informação e características biométricas do utilizador e enviá-las para o servidor.

5.3 Servidor (*Web Server*)

Pode afirmar-se que o cérebro de toda a plataforma assenta no servidor, é nele que estão contidos os algoritmos que são o cerne dos principais processos do sistema. O servidor é considerado assim responsável por toda a parte lógica do sistema.

A plataforma foi desenvolvida em Python, utilizando uma *framework* de desenvolvimento de plataformas web, o Django. A utilização desta ferramenta permite que, de uma forma simplificada, possa ser produzida e publicada uma plataforma web. O Django disponibiliza um *Web Server* simples, utilizado neste trabalho, que permite publicar a plataforma, o que faz com que, quando se acede ao *url* `IP_SERVIDOR:PORTA_SERVIDOR/enrollment/` apareça o conteúdo no *browser*, permitindo desta forma o contacto com o servidor. A implementação dos algoritmos já descritos e apresentados no capítulo 4 necessitou de utilizar algumas bibliotecas: do ramo matemático, utilizou-se a Numpy, uma biblioteca com uma larga coleção de funções matemáticas para trabalhar estruturas de dados, aplicando operações como subtração, multiplicação, transposição, etc; na área da visão por computador, a biblioteca OpenCV, que serviu para simplificar alguns processos, como o histograma de uma imagem, o simples redimensionamento de uma imagem, ou mesmo efetuar a deteção da face do utilizador na imagem recolhida, fazendo uso de um método que implementa o algoritmo *Viola e Jones* (secção 3.2.2.1), entre outros;

Os algoritmos implementados efetuam imensas operações entre matrizes multidimensionais com um tamanho gigantesco(dimensão 640 x 480), o que faz com que o tempo de execução, quer no processo de registo ou autenticação, tenha atingido tempos incomportáveis. Atendendo ao facto de o Python ser uma linguagem de programação interpretada e os seus tempos de execução não serem um ponto forte da linguagem, foi apresentada uma alternativa: desenvolver o algoritmo em C++. Posto isto, para que fosse possível executar código C++ durante a execução de Python foi utilizado o SWIG(*Simplified Wrapper and Interface Generator*), que permite invocar um método em C++ a partir do Python. Resultados e comparações de tempos são apresentados no próximo capítulo(6): Resultados Experimentais.

O servidor encontra-se ligado a uma base de dados MySQL que permite a persistência das imagens e informação do utilizador. A *framework* Django também facilita este processo pois suporta técnicas como ORM(*Object-Relational Mapping*) que permite mapear uma classe diretamente a uma tabela da base de dados, efetuando automaticamente todas as ligações e interações com o SGDB(Sistema de Gestão de Base de Dados).

Em suma, além da disponibilização do HTML, o servidor é responsável por efetuar todas as etapas dos processos biométricos, exceto a aquisição de características que é delegada à parte Cliente.

5.4 Tecnologias e Bibliotecas Utilizadas

Embora já tenham sido mencionadas algumas delas, no decorrer deste trabalho surgiram várias necessidades que levaram à procura e utilização de bibliotecas e/ou tecnologias. As linguagens utilizadas foram o Python, C++ e JavaScript. O facto de este trabalho se encontrar ligado à área de visão por computador e necessitar de manipular matrizes multidimensionais, as principais bibliotecas utilizadas nestas manipulações foram o OpenCV e NumPy, devido à sua notoriedade e resultados comprovados. São apresentadas e descritas adiante as restantes bibliotecas e tecnologias utilizadas.

Bibliotecas utilizadas:

- **OpenCV** - Biblioteca de visão por computador, que fornece várias funções de processamento de imagem[6].
- **Numpy** - Coleção de funções que suportam operações entre *arrays* e matrizes multidimensionais[7].
- **PIL** - Sigla que significa *Python Imaging Library*, permite uma manipulação de imagens em diferentes formatos [8].
- **Matplotlib** - Foi empregada para representar graficamente os *arrays*, reconstruindo assim as imagens, utilizadas nos resultados experimentais [9].
- **Pickle** - Esta biblioteca foi utilizada em conjunto com a Django, o que permitiu guardar *pickles* em base de dados, ou seja, quaisquer variáveis em tempo de execução, independentemente do seu tipo[10].
- **Django** - *Framework* que permitiu o desenvolvimento mais automatizado da plataforma [11].
- **Navigator.getUserMedia API** - Biblioteca em JavaScript que permitiu aceder aos componentes de vídeo dos dispositivos através do *browser* [12].

[6] *Website* da biblioteca - `http://opencv.org/`
[7] *Website* da biblioteca - `http://www.numpy.org/`
[8] *Website* da biblioteca - `http://www.pythonware.com/products/pil/`
[9] *Website* da biblioteca - `http://matplotlib.org/`
[10] *Website* da biblioteca - `https://docs.python.org/2/library/pickle.html`
[11] *Website* da biblioteca - `https://www.djangoproject.com/`
[12] *Website* da biblioteca - `http://www.w3.org/TR/mediacapture-streams/`

Tecnologias utilizadas:

- **Xubuntu** - Sistema operativo Xubuntu 14.04.2 LTS, x86_64, serviu como ambiente de trabalho.
- **Eclipse** - Ambiente de desenvolvimento com a utilização do *PyDev*[13].
- **Python** - Linguagem de programação utilizada, versão 2.7.9, 32 bits.
- **VirtualBox** - Foram utilizadas máquinas virtuais para testar e desenvolver a plataforma.
- **Chrome** - *Browser* utilizado para o acesso ao servidor, versão 45.0.2454.101 m.
- **MySQL** - Sistema de gestão de base de dados escolhido para o armazenamento de informação, versão 5.5.44.

[13] *Website* do *plugin* - `http://www.pydev.org/`

Capítulo 6

Resultados Experimentais

Este capítulo pretende apresentar resultados e demonstração das experiências efetuadas ao sistema, sendo importante frisar que todos os testes foram realizados em iguais circunstâncias e perante os mesmos recursos. A tabela abaixo apresenta as características da máquina virtual utilizada:

Sistema Operativo	Xubuntu 14.04.2 LTS 64bits
CPU	Intel(R) Core(TM) i7-3612QM CPU @ 2.10GHz
Memória RAM	2048 MiB
Armazenamento	20,00 GB
Sistema Virtualização	VirtualBox Versão 4.3.16

Tabela 6.1: Especificações do ambiente de testes.

6.1 Comparação entre PCA e TI-PCA

O TI-PCA não é mais que a aplicação do próprio PCA, utilizando o SIC para alinhar as imagens antes de efetuar a recolha dos componentes principais. Isto faz com que, num processo de autenticação, no momento da classificação, a imagem de prova se encontre alinhada com as imagens de treino, reduzindo assim o erro entre ambas. Isto porque quando é efetuado o pré processamento e é detetada a face do utilizador na imagem recolhida, é obtido um ponto, com coordenadas x e y, bem como o comprimento e largura, permitindo assim saber quais os outros três pontos necessários para fazer o quadrado que conterá a face do utilizador.

São apresentados três casos distintos:

- **Caso 1** (6.1 e 6.2) - Utilizador com a face direita, sem existir alinhamento.
- **Caso 2** (6.3 e 6.4) - Utilizador com a face torta, sem existir alinhamento.
- **Caso 3** (6.5 e 6.6) - Utilizador com a face torta, com a existência de alinhamento [1].

No primeiro caso, a imagem de prova é exemplo do padrão que deve ser obtido para um processo de autenticação, não necessitando de reposicionamento.

FIGURA 6.1: Deteção da cara sem alinhamento mas com face alinhada.

FIGURA 6.2: Recorte da face presente na figura 6.1.

Caso o utilizador, por algum motivo, não tenha a cara totalmente alinhada, ou seja, na mesma posição e perspetiva aquando do seu registo no sistema, pode resultar em situações como a que é apresentada na figura 6.3, o que, num processo de reconhecimento com recurso ao PCA tradicional, é algo drasticamente penalizado. Com base nisto, a figura seguinte apresenta os primeiros resultados práticos do algoritmo, alinhando a face antes de efetuar a extração.

FIGURA 6.3: Deteção da cara sem alinhamento e sem a face alinhada.

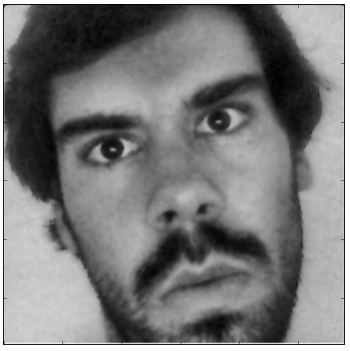


FIGURA 6.4: Recorte da face presente na figura 6.3.

[1] A imagem de prova utilizada no caso 1 e caso 2 é a mesma.

Neste último caso, a aplicação do TI-PCA à imagem de prova efetua o alinhamento com base na face do utilizador, melhorando assim o seu processo de extração e permitindo uma classificação mais séria e equiparada. Posto isto, caso exista algum desvio, inclinação ou rotação no rosto na imagem de prova, esta será alinhada previamente.

FIGURA 6.5: Deteção da cara com alinhamento mas sem face alinhada.

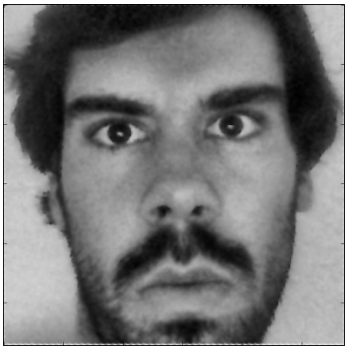


FIGURA 6.6: Recorte da face presente na figura 6.5.

Como se pode constatar, observando as figuras 6.2 e 6.6, a sua semelhança é obtida pelo facto de o alinhamento ter sido efetuado na segunda imagem, caso contrário seria efetuada a tentativa de autenticação com a figura 6.4, o que obviamente iria reduzir em muito a possibilidade de identificação.

Para demonstrar este facto foram obtidos os valores da matriz transformação e erro (distância euclidiana) de cada um dos três casos acima demonstrados. Este valores são provenientes de três experiências efetuadas ao sistema final. Os valores da tabela seguinte originam uma média de 2.99 e um desvio padrão de 1.29:

Casos	Matriz Transformação			Erro[2]
	s	t_x	t_y	
1	0	0	0	1.97
2	0	0	0	4.81
3	-0.106	-35.495	50.586	2.19

TABELA 6.2: Tabela com resultados dos três diferentes casos.

Percebe-se através dos resultados obtidos que o primeiro caso foi o que conseguiu obter um erro menor, ou seja, uma maior similaridade com o registo do utilizador. No segundo caso o erro aumentou mais que o dobro, o que corresponde à face

ligeiramente inclinada e sem ser efetuado qualquer alinhamento pelo sistema. Por último, o terceiro caso apresenta um erro pouco maior do que o primeiro caso. No entanto, a imagem de prova foi exatamente a mesma do caso dois, a diferença incide no facto de ter sofrido o alinhamento.

6.1.1 Imagens Médias

Não só no processo de autenticação mas também no registo são alinhadas as imagens que dão entrada no sistema. Mas por que razão são alinhadas as imagens no processo de registo? Embora se preveja que o utilizador respeite as regras de boas práticas, quanto mais dispersas estiverem as imagens base, na fase de registo, mais dificultada será a tarefa de obter os componentes principais desse conjunto de imagens. Portanto, é calculada a face média das imagens base e de seguida alinhadas as imagens com base nessa face média. Quando se calcular novamente a face média sobre as imagens alinhadas esta terá de se encontrar em melhores condições de efetuar a extração de características, ou seja, achar os componentes principais. As imagens abaixo apresentam a imagem média antes e depois de efetuar o alinhamento e percebemos claramente que há uma melhoria notória na sua sobreposição na figura 6.8.

FIGURA 6.7: Face média antes do alinhamento.

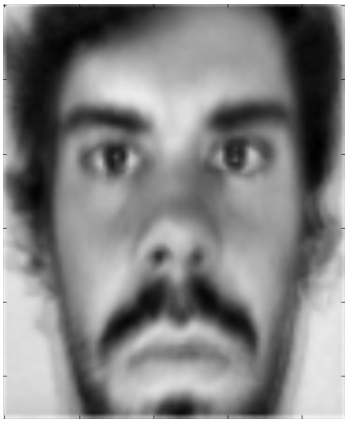


FIGURA 6.8: Face média depois do alinhamento.

6.2 Alinhamento através do SIC

O algoritmo SIC é o responsável pelo alinhamento da imagem, como já foi fundamentado anteriormente (secção 4.2), achando uma matriz de transformação, que melhor descreve o alinhamento entre um *template* e uma imagem de origem. Como podemos observar este processo é um processo iterativo, em que a cada iteração é feita uma transformação que acaba quando atinge o valor de erro mais baixo. Foram retirados os valores de um processo de alinhamento que é apresentado na tabela 6.3, na qual se pode perceber uma alteração decrescente no erro e alteração dos valores da transformação. Na figura 6.9 pode também ver-se, de uma forma mais real e representativa, os dados presentes na tabela e, por fim, o gráfico 6.2 permite uma análise do erro (representado pela fórmula (4.20)) a cada iteração.

Na tabela seguinte estão presentes os valores da matriz transformação, que vão sofrendo alterações até chegarem aos melhores valores possíveis, que são obtidos quando é atingido o valor mais baixo de erro, achado quando se dá uma inversão na sua descida (neste caso na iteração dezoito). Como se pode perceber, este processo sofre dezassete iterações até obter os valores finais da matriz transformação, sendo que a iteração dezoito é descartada. Estes valores foram obtidos com base numa experiência efetuada ao sistema final, em que, os mesmos, foram extraídos a cada iteração no processo de alinhamento. Valores de s, t_x e t_y multiplicados por e^{-5}:

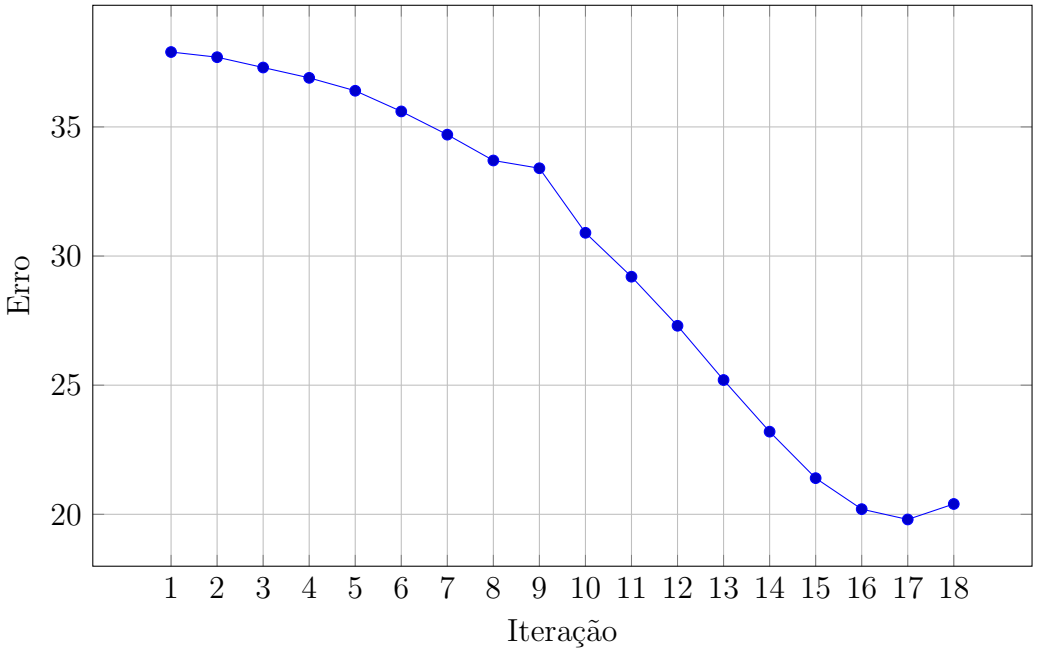
Iterações	s	t_x	t_y	**Erro**
1	-0.002	-0.496	0.600	37.9
2	-0.005	-1.597	1.640	37.7
3	-0.011	-3.316	3.412	37.3
4	-0.019	-5.561	5.724	36.9
5	-0.029	-8.368	8.603	36.4
6	-0.040	-11.757	12.083	35.6
7	-0.054	-15.733	16.140	34.8
8	-0.069	-20.296	20.788	33.7
9	-0.087	-25.442	26.021	32.4
10	-0.106	-31.202	31.868	30.9
11	-0.127	-37.471	38.214	29.2
12	-0.149	-44.210	45.032	27.3
13	-0.172	-51.249	52.159	25.2
14	-0.195	-58.151	59.106	23.2
15	-0.217	-65.034	66.002	21.4
16	-0.239	-71.861	72.826	20.2
17	-0.260	-78.591	79.531	19.8
18	-0.280	-85.122	86.026	20.4

TABELA 6.3: Valores do processo de alinhamento.

A cada iteração pode-se obter uma matriz transformação, que nos permite descrever essa mesma transformação perante a posição inicial. Desta forma foi construída uma imagem onde são representadas todas as transformações até obter a final. É então apresentada a imagem original, seguida da imagem com todas as transformações sofridas e finalmente a transformação final.

FIGURA 6.9: Alinhamento através do SIC.

Por fim, o gráfico abaixo representa o percurso decrescente tomado pelo erro ao longo das dezassete iterações e o seu ponto de inversão na iteração dezoito. Inicialmente o erro diminui moderadamente, começando a aumentar a sua descida gradualmente até que começa a estagnar e atinge o mínimo.



6.3 Comparação entre Tempos de Execução Python e C

Este trabalho foi desenvolvido em Python, não só a plataforma web mas também toda a parte lógica e algorítmica inerente. No entanto, durante os primeiros testes ao sistema percebeu-se que os tempos de execução do algoritmo eram impraticáveis, o que levou à escolha da linguagem C para efetuar a execução do mesmo. Como tal, o mesmo algoritmo foi implementado em duas linguagens diferentes que concedem assim performances distintas ao sistema. Para comprovar esta realidade, foram extraídos os tempos de ambas as linguagens nos dois casos de uso existentes, o registo e a autenticação. Para efetuar testes de aceitação ao sistema foi necessário a utilização de uma base de dados criada para o efeito. Foi utilizada a ≪Exteded Yale Face Database B≫[12], selecionando dezassete amostras de entre as disponíveis.

Registo

Este processo conta com cinco imagens de base por utilizador, que serão alinhadas e determinarão os componentes principais do utilizador. Na tabela seguinte estão representadas as médias dos tempos obtidos nas dezassete amostras utilizadas, em cada uma das linguagens. Percebe-se que a diferença é gigantesca, o tempo de execução em Python é aproximadamente vinte e duas vezes maior do que em C. A média do tempo de execução do algoritmo PCA é de aproximadamente 2.9 segundos, notando que, mesmo aquando da sua utilização no processo TIPCA este é sempre executado em Python. Tempos apresentados em segundos.

	Python	C
TIPCA	826.16	37.22

Tabela 6.4: Média de tempos de execução no processo de registo.

Autenticação

Contrariamente ao processo anterior, neste é apenas alinhada a imagem de prova a ser validada, reduzindo em muito a quantidade de dados a processar. Foram utilizadas também dezassete amostras para testar o sistema. Como era de esperar, pela experiência do caso anterior, a discrepância entre a média dos tempos nas duas linguagens continua a ser enorme. No entanto, os tempos obtidos pela linguagem C são um bom resultado para um sistema de reconhecimento.Tempos apresentados em segundos.

	Python	C
TIPCA	169.35	7.07

Tabela 6.5: Média de tempos de execução no processo de autenticação.

6.4 Testes de Aceitação

Para efetuar testes de aceitação ao sistema foi necessário a utilização de uma base de dados criada para o efeito. Foi utilizada a «Exteded Yale Face Database B»[12], selecionando dezassete amostras de entre as disponíveis.

As taxas de aceitação num sistema de autenticação biométrico dependem do valor limite de aceitação (*threshold*), pois as taxas irão variar consoante o valor definido.

Nos testes mais à frente apresentados será indicado o *threshold* utilizado bem como a taxa de aceitação obtida num determinado conjunto de amostras. Foram efetuados dezassete pedidos de autenticação em dois casos distintos. Num deles foi utilizado somente PCA e no outro TIPCA, isto requer inerentemente dois registos prévios e distintos, um que contém as imagens de treino alinhadas (TIPCA) e outro sem qualquer alinhamento (PCA).

Com os resultados provenientes da autenticação em cada caso, em cada amostra, foi construída a seguinte tabela, que apresenta o erro calculado pela distância euclidiana (norma) entre a imagem original e a sua reconstrução[3]. Na utilização do PCA, foi obtida uma média de $5,30$ e um desvio padrão de 2.68 enquanto que no TIPCA, a média foi de $2,98$ e o desvio padrão de $2,39$.

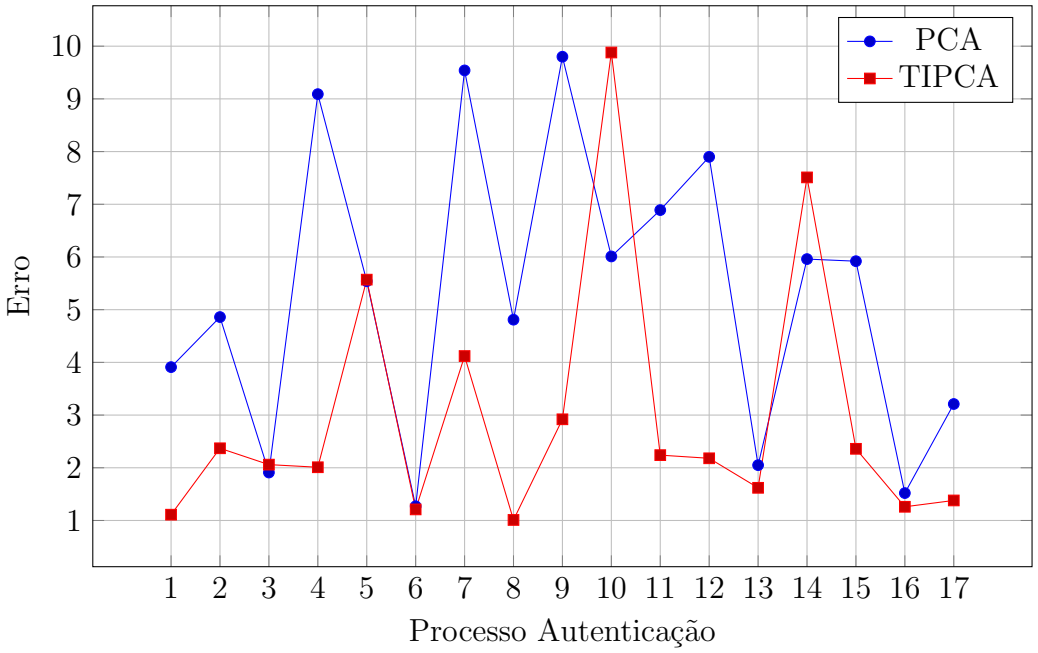
	Erro PCA	Erro TIPCA
1	3,91	1,11
2	4,86	2,37
3	1,91	2,06
4	9,09	2,01
5	5,54	5,57
6	1,27	1,21
7	9,54	4,12
8	4,81	1,01
9	9,80	2,92
10	6,01	9,88
11	6,89	2,24
12	7,90	2,18
13	2,05	1,62
14	5,96	7,51
15	5,92	2,36
16	1,52	1,26
17	3,21	1,38

TABELA 6.6: Tabela com resultados de autenticação. (Valores em segundos)

[3]Explicado na subseção 4.3.

O resultados indicam que a distância calculada, ou seja, o erro que representa a diferença entre as duas faces é menor aquando a utilização do algoritmo TIPCA. O que nos leva a concluir que este apresenta melhorias face ao seu precedente (PCA).

Utilizando os valores obtidos nos dezassete processo de autenticação em ambos os casos, foi elaborado um gráfico para analisar de uma forma mais representativa os resultados. Observando o gráfico seguinte, destaca-se a superioridade da linha azul perante a vermelha. Indicando, que o valor do erro em TIPCA é muito menor do que em PCA, excepto quatro amostras, onde duas delas os valores TIPCA são mais elevados, piorando o processo, e nas outras duas os valores são aproximadamente iguais o que quer dizer que, para estas duas amostras, é indiferente a escolha do algoritmo.



Perante este gráfico, se se definir um *threshold* de três, que equivale a traçar uma linha reta em $y = 3$. Depois disto, todos os pontos que ficassem abaixo da reta serão considerados validados enquanto os que estivessem acima será negada a sua identidade. Posto isto, perante TIPCA, em dezassete amostras, existiriam quatro que não seriam reconhecidas, consideradas de falsas negações, obtendo uma taxa de sucesso de aproximadamente 76%. Contundo, subindo o valor do *threshold* para o dobro, a taxa de sucesso também seria mais elevada, atingindo os 88%.

Utilizados os mesmo valores de *threshold* para analisar a taxa de sucesso do utilizando apenas PCA, obter-se-ia aproximadamente 17% com limiar de três e 65%

com um limiar de 6. Comparando as taxas de sucesso dos dois algoritmos percebemos claramente a superioridade do TIPCA perante o PCA.

No gráfico podemos observar que, na amostra dez e catorze o erro aumentou com a utilização de TIPCA. Isto deve-se ao facto de que, o alinhamento efetuado nem sempre é feito corretamente, podem existir dévios que prejudicarão na fase de decisão.

6.5 Redução da Dimensionalidade da Imagem

Para saber até que ponto a dimensão das amostras utilizadas pode influenciar o processo de autenticação do utilizador, foram feitos vários testes com diferentes dimensões, como apresenta a imagem seguinte:

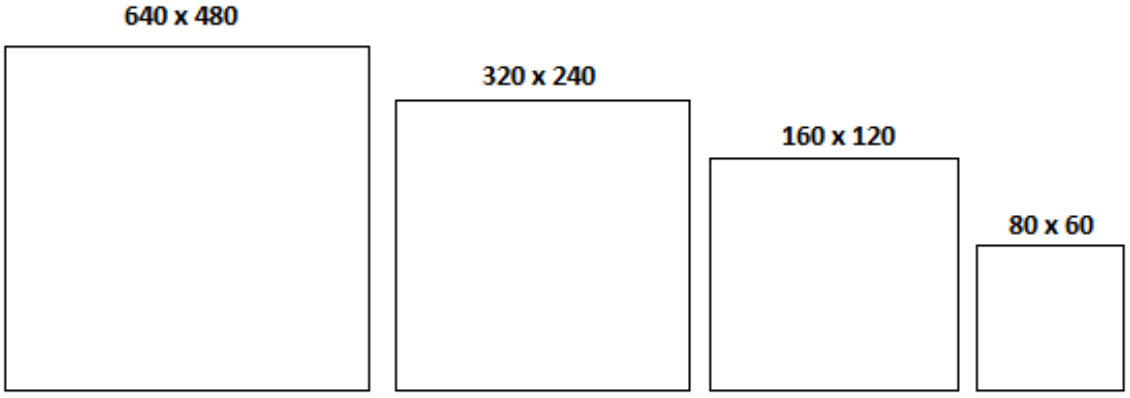


FIGURA 6.10: Redimensionamento das imagens para autenticação.

Foram efetuados os registos das dezassete amostras em quatro dimensões diferentes, utilizando a base de dados «Exteded Yale Face Database B»[12]. No processo de identificação, em cada uma das amostras nas quadro diferentes dimensões, foram obtidos os erros que determinam o grau de similaridade entre as imagens de prova e de treino. Para estes testes foi construído um *script* que reduzir a dimensão da imagem antes de a processar. Na dimensão original de 640×480 foi obtido o mesma média e desvio padrão do que no caso anterior (os valores são os mesmo pois as dezassete amostras são iguais em todas as experiências). Reduzindo a dimensão para 320×240 a média é de 4.49 e o desvio padrão de 2.77. Reduzindo novamente para 160×120 a média é de 3.84 e o desvio padrão de 2.37. Na última redução, de 80×60,

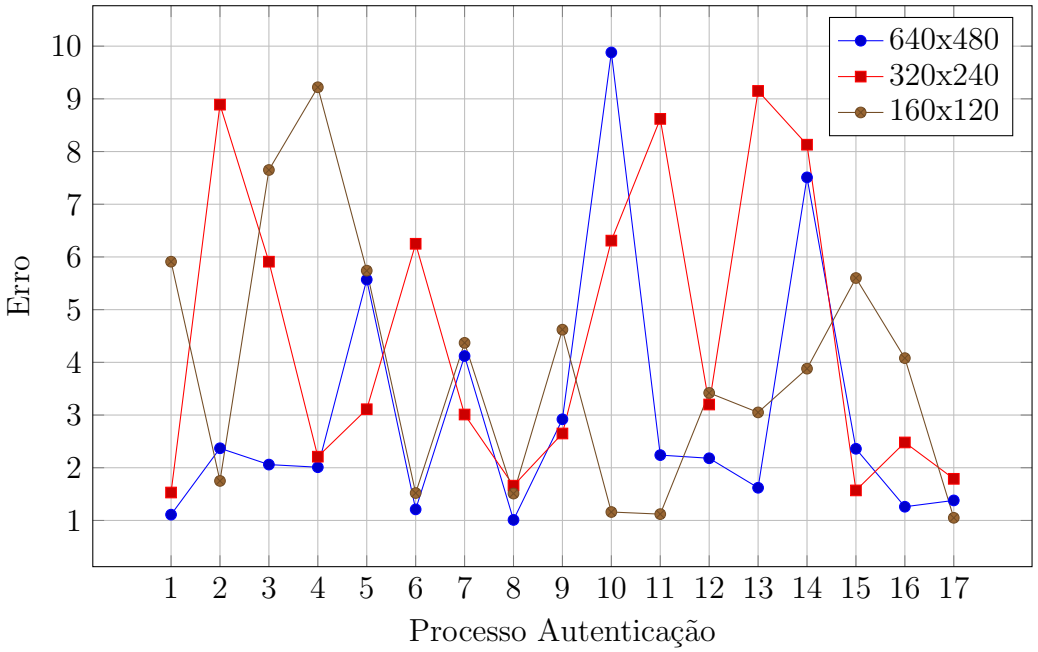
existiram algumas amostras que já não foram processadas pelo sistema, impedindo desta forma calcular a média e desvio padrão a comparar com as restantes.

Os valores obtidos encontra-se na seguinte tabela:

	640 x 480	320 x 240	160 x 120	80 x 60
1	1,11	1,53	5,91	3,93
2	2,37	8,89	1,75	-
3	2,06	5,91	7,65	-
4	2,01	2,21	9,22	9,35
5	5,57	3,11	5,74	-
6	1,21	6,25	1,52	1,74
7	4,12	3,01	4,37	-
8	1,01	1,66	1,51	2,37
9	2,92	2,65	4,62	1,06
10	9,88	6,31	1,16	6,00
11	2,24	8,62	1,12	1,31
12	2,18	3,20	3,42	6,65
13	1,62	9,15	3,05	2,89
14	7,51	8,13	3,88	6,5
15	2,36	1,57	5,60	-
16	1,26	2,48	4,08	-
17	1,38	1,79	1,05	4,07

TABELA 6.7: Erros perante nas diferentes dimensões de imagens.

Representando os valores graficamente, obtemos a seguinte gráfico:



Utilizando o gráfico, se existisse como no exemplo anterior um *threshold* de três e traçando a linha no eixo do y, obtendo uma taxa de sucesso em cada dimensão. A dimensão de 640x480 tem uma percentagem de sucesso de 76%, 320x240 de 47%, 160x120 de 37%, tendo por último 80x60 com 23%. Tentou-se ainda testar a dimensão de 40x30, no entanto, em nenhuma das amostras foi possível sequer identificar a face do utilizador. Como se pode constatar a taxa de aceitação varia dependendo da dimensão da imagem original, e neste caso, quanto menor é a dimensão de imagem, mais difícil fica o processo de autenticação.

As taxas de sucesso obtidas, tal como já foi referido, não são fixas, variam também consoante o *threshold* definido. Caso se aumente o valor do limite de aceitação, as percentagem de sucesso irão também aumentar.

6.6 Conclusões e Alternativas

Tendo em conta todos os testes efetuados e resultados obtidos, existem várias conclusões que podemos retirar quanto à performance do algoritmo implementado (TIPCA). Um dos pontos principais a destacar prende-se com o facto de o algoritmo TIPCA melhorar significativamente os resultados do tradicional PCA, como demonstrara *Weihong Deng*, *Jiani Hu* e *Jiwen Lu*[8]. No entanto, nas amostras utilizadas para os testes deste trabalho, pudemos observar que existem exceções em que o TIPCA

não apresenta melhorias face ao PCA; pelo contrário, aumenta a incerteza da autenticação. Ainda assim, num conjunto de dezassete amostras, apenas duas produziram um efeito negativo na utilização do TIPCA, como se constatou anteriormente.

Uma das conclusões que se pôde tirar da utilização do algoritmo TIPCA é que, em faces que estejam inclinadas: se as faces estiverem numa perspetiva frontal, o resultado é muito mais satisfatório do que quando a face está numa perspetiva lateral ou de perfil, o que se pode considerar como um ponto fraco. Outro ponto fraco da utilização deste algoritmo é o facto de existir a possibilidade de o alinhamento não ser feito corretamente, o que resulta num aumento do erro na fase de decisão e consequente resultado negativo. Como constataram Sanchez-Lozano e Alba-Castro, «muitos autores têm advertido que os algoritmos de composição inversa falham na regra de composição, permitindo a existência de formas inválidas»[26].

O tempo de execução do algoritmo é outro dos pontos mais abordados. Sendo este projeto direcionado para um sistema de autenticação, o tempo é um fator fundamental. O elevado processamento de dados, como é o caso, requer a utilização de uma linguagem adequada ao efeito. Foi então proposto o Python, pela sua facilidade de utilização, devido ao facto de ser uma linguagem de alto nível e possuir uma grande quantidade de bibliotecas de engenharia e computação científica. No entanto, um dos pontos fracos do Python é precisamente o seu tempo de execução. Para contornar esta dificuldade, foi utilizado o C para realizar os processos mais morosos. Como pudemos observar nos testes anteriores, a diferença de tempos de execução entre Python e C é gigantesca. Foi portanto conjugado o que de melhor tem cada uma das linguagens, unindo-se através do SWIG.

Não obstante, existem outras linguagens que podem ser utilizadas e que, pelas suas caraterísticas, são uma mais valia neste tipo de sistemas, como é o exemplo do Julia. O «Julia é um novo design de alta performance e linguagem de programação dinâmica para a computação técnica e científica.»[9]. Julia é extremamente mais rápido do que Python, ainda assim não contém um grande número de bibliotecas ao seu dispor, no entanto, é compatível com Python. Isto quer dizer que o código Python pode ser misturado e invocado em Julia. Desta forma podemos obter ganhos na rapidez de execução sem a necessidade de recorrer a linguagens de baixo nível. Ioana e Radu Dogaru apresentaram algumas comparações de tempos de execução entre

código Python e Julia, utilizando ainda optimizações como o Numba[4] e NumbaPro CUBA[5][9]. A diferença entre os códigos *standard*, sem optimizações, é enorme, e embora sejam ambas linguagens de programação dinâmica, Julia é muito mais rápida do que Python. É possível, a partir do Julia, utilizar bibliotecas como o Matlab ou SciPy, de Python, tornando assim Julia uma linguem alternativa às utilizadas neste trabalho.

[4]Numba é uma biblioteca que acelera a execução de processos, recorrendo a funções de alto desempenho, escritas diretamente em Python.

[5]NumbaPro é uma versão melhorada do Numba e adiciona funcionalidades que permitem criar rapidamente código otimizado-o para arquiteturas GPU. NumbaPro permite também a escrita de código CUBA diretamente no Python.

Capítulo 7

Conclusões e Trabalhos Futuros

7.1 Conclusões Finais

Este trabalho estudou o algoritmo TI-PCA de forma a melhorar o grau de confiabilidade em sistemas de autenticação biométrica. Posteriormente foi desenvolvida uma plataforma web com base no mesmo, permitindo efetuar registos e pedidos de autenticação.

Hoje em dia, qualquer sistema de autenticação biométrico não pode cingir-se apenas a algoritmos tradicionais. Embora estes ofereçam a base, devem ser melhorados para responder às necessidades atuais. O TI-PCA veio provar que, com base num algoritmo tradicional, é possível melhorar resultados e suportar uma maior diversidade de situações inesperadas. Apesar de não existir uma certeza imaculada em sistemas de autenticação facial, as taxas de aceitação obtidas começam a ser bastante propícias à sua utilização em sistemas de segurança. Este algoritmo acrescenta uma nova etapa ao processo, o alinhamento, que, como provado, melhora significativamente os resultados obtidos no processo de decisão. Este apresenta no entanto, algumas fragilidades, nomeadamente o facto do alinhamento poder ser feito incorretamente. O desenvolvimento deste algoritmo teve também alguns problemas quanto aos tempos de execução, pelo facto de trabalhar uma grande quantidade de dados, no entanto, a utilização de linguagens de baixo nível veio acelerar o processo e colocá-lo com tempos bastante aceitáveis. Constatou-se, portanto, que a diferença de tempos de execução entre Python e C é tremenda, sendo o C muito mais apropriado para a execução deste algoritmo. Foi conjugado o que de melhor têm ambas as linguagens: no Python a facilidade e rapidez de desenvolver uma plataforma web, simplificando

processos como por exemplo, acessos à base de dados e gestão de pedidos HTTP; e no C, foi aproveitado o facto de ser uma linguagem de baixo nível, em que a execução de grandes operações e volume de dados é muito mais eficiente.

Outro dos pontos prementes deste trabalho foi a importância do alinhamento de imagens, efetuado através do algoritmo SIC e dando origem ao TI-PCA, que fez com que o algoritmo PCA conseguisse criar um *Eigenspace* otimizado, melhorando a projeção das imagens que requerem validação.

Ao longo desta dissertação pôde-se perceber também que a biometria é uma área que tem vindo a estar cada vez mais presente em dispositivos, sejam eles para consumo particular ou empresarial, apontando assim esta área como uma tendência tecnológica, que é cada vez mais explorada a nível comercial.

7.2 Trabalho Futuro

Uma das ideias para trabalho futuro consiste no desenvolvimento de uma API de autenticação por reconhecimento facial, no qual qualquer plataforma web poderia utilizá-la para efetuar o login nos seus sistemas, delegando essa responsabilidade. Esta API seria comprada por empresas que quisessem reforçar ou introduzir nos seus sistemas, uma autenticação através de reconhecimento facial. Esta vertente, mais comercial, seria pensada de forma a constituir uma empresa em que, o seu produto principal, fosse a API.

Apesar dos tempos de execução neste trabalho terem sido reduzidos em muito com a utilização de uma linguagem de baixo nível, existem formas de reduzir ainda mais os tempos e melhorar o desempenho. Um dos trabalhos futuros poderia consistir no uso da paralelização, através de programação heterogênea, para reduzir ainda mais os tempos de execução. O OpenCL é uma *framework* utilizada no desenvolvimento de programas heterogêneos que estabelece uma ligação mais direta entre o programador e a utilização de processadores, como por exemplo o GPU. Deste modo, um trabalho futuro poderia ser a implementação do algoritmo TIPCA recorrendo ao OpenCL, ou quaisquer outras tecnologias similares, com o objetivo de melhorar os tempos de execução.

Foi demonstrado nesta dissertação que a utilização do TIPCA trouxe algumas melhorias perante o PCA tradicional, no entanto, existem casos em que o TIPCA prejudica o reconhecimento e piora o processo. Um dos trabalhos futuros poderia

ser, estudar as causas que levam alguns desdes casos a acontecer e saber de quer forma se poderia melhorar o algoritmo, com o objetivo de prevenir essas situações.

Embora neste trabalho toda a orientação do reconhecimento facial recaia sobre a segurança em sistemas de informação, este sistema poderá ser utilizado para outros fins. Deste modo, outra possibilidade seria aproveitar o reconhecimento facial na área de *marketing* e publicidade. Numa loja de venda de produtos ao consumidor final, poderia existir um sistema de reconhecimento facial, a que o cliente se submetia e desta forma lhe eram indicadas promoções e campanhas de produtos do seu interesse.

Referências Bibliográficas

[1] S. Baker and I. Matthews. Lucas-kanade 20 years on: A unifying framework, part 1: The quantity approximated, thewarp update rule, and the gradient descent approximation. *International Journal of Computer Vision*, 2002.

[2] S. Baker, I. Matthews, and J. Schneider. Automatic construction of active appearance models as an image coding problem. *IEEE Trans Pattern Anal Mach Intell*, 2004.

[3] M. Bartlett, J. Movellan, and T. Sejnowski. Face recognition by independent component analysis. *Neural Networks, IEEE Transactions on*, page 1450–1464, 2002.

[4] M. Brown and S. J. Rogers. A practical approach to user authentication. *Computer Security Applications Conference, 1994. Proceedings, IEEE*, pages 108–116, 1994.

[5] R. W. Bruegge. The bcoe and the future of biometrics at the fbi. *Biometrics: Theory, Applications and Systems, 2008. BTAS 2008. 2nd IEEE International Conference*, 2008.

[6] P. C. Cattin. Biometric authentication system using human gait. Master's thesis, Swiss Federal Institute of Technology ETH Zurich, 2002.

[7] F. Z. Chelali, A. Djeradi, and R. Djeradi. Linear discriminant analysis for face recognition. *Multimedia Computing and Systems, 2009. ICMCS '09. International Conference on, IEEE*, 2009.

[8] W. Deng, J. Hu, J. Lu, and J. Guo. Transform-invariant pca: A unified approach to fully automatic face alignment, representation, and recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on Vol. 36*, 2014.

[9] I. Dogaru and R. Dogaru. Using python and julia for efficient implementation of natural computing and complexity related algorithms. *Control Systems and Computer Science (CSCS), 2015 20th International Conference*, pages 599 – 604, 2015.

[10] H. M. Ebied. Feature extraction using pca and kernel-pca for face recognition. *Informatics and Systems (INFOS), 2012 8th International*, pages MM–72–MM–77, 2012.

[11] Z. Fan, Y. Xu, W. Zuo, J. Yang, J. Tang, Z. Lai, and D. Zhang. Modified principal component analysis: An integration of multiple similarity subspace models. *Neural Networks and Learning Systems, IEEE Transactions on Vol. 25*, 2014.

[12] A. Georghiades, P. Belhumeur, and D. Kriegman. From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE Trans. Pattern Anal. Mach. Intelligence*, 23(6):643–660, 2001.

[13] A. Jain, L. Hong, and S. Pankanti. Biometric identification. *Communications of the ACM*, 43:90–98, 2000.

[14] A. Jain, S. Pankanti, and A. Ross. Biometrics: A tool for information security. *IEEE Transactions on Information Forensics and Security*, pages 125–143, 2006.

[15] A. K. Jain, K. Nandakumar, and A. Nagar. Biometric template security. *EURASIP Journal on Advances in Signal Processing*, 2008.

[16] T. S. Korting and N. L. D. Filho. Utilizando eigenfaces para reconhecimento de imagens. *Engenharia de Computação - Fundação Universidade Federal do Rio Grande*, 2004.

[17] D. Lowrence. Biometrics and retail: moving towards the future. *In Biometric Technology Today*, pages 7–9, 2014.

[18] O.A.Esan, S.M.Ngwira, and I.O.Osunmakinde. Bimodal biometrics for financial infrastructure security. *Information Security for South Africa, 2013 , IEEE*, 2013.

[19] K. O'Flaherty. Security in 2015: Biometrics. *SC Magazine: For IT Security Professionals (UK Edition)*, pages 25–27, 2014.

[20] K. Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine 2(11)*, pages 71–86, 1901.

[21] J. Pereira, G. Cavalcanti, and T. I. Ren. Modular image principal component analysis for face recognition. *Neural Networks, 2009. IJCNN 2009. International Joint Conference*, pages 2481–2486, 2009.

[22] S. Rahal, H. A. H.A., and K. Muteb. Multimodal biometric authentication system - mbas. *Information and Communication Technologies, 2006. ICTTA '06. 2nd*, pages 1026–1030, 2006.

[23] K. Ramírez-Gutiérrez, D. Cruz-Pérez, and H. Pérez-Meana. Face recognition and verification using histogram equalization. *World Scientific and Engineering Academy and Society (WSEAS)*, 2010.

[24] M. A. Sasse. Red-eye blink, bendy shuffle, and the yuck factor: A user experience of biometric airport systems. *Security and Privacy, IEEE*, 2007.

[25] C. A. Shoniregun and S. Crosier. *Securing Biometrics Applications*. Springer, 2008.

[26] E. Sánchez-Lozano, D. González-Jiménez, and J. L. Alba-Castro. Parameter constraints for accurate face alignment in the simultaneous inverse compositional algorithm. *Image and Signal Processing and Analysis (ISPA), 2011 7th International Symposium*, 2011.

[27] V. Starovoitov, D. Samal, and D. Briliuk. Image enhancement for face recognition. *Submitted to International Conference on Iconics*, 2003.

[28] Y. TaigmanMing, YangMarc'Aurelio, and R. Wolf. Deepface: Closing the gap to human-level performance in face verification. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[29] M. A. Turk. Interactive-time vision: Face recognition as a visual behavior. Master's thesis, Massachusetts Institute of Technology, 1991.

[30] M. A. Turk and A. P. Pentland. Face recognition using eigenfaces. *Computer Vision and Pattern Recognition, 1991. Proceedings CVPR '91., IEEE Computer Society Conference on*, pages 586–591, 1991.

[31] J. R. Vacca. *Biometric Technologies and Verification Systems*. Butterworth-Heinemann, 2007.

[32] M. Vashek and R. Zdenek. Security of biometric authentication system. *Computer Information Systems and Industrial Management Applications (CISIM), 2010 International Conference on, IEEE*, page 175, 2010.

[33] C. Vielhauer. *Biometric User Authentication for IT Security: From Fundamentals to Handwrinting*. Springer Science+Business Media, 2006.

[34] C. Vielhauer. *Biometric User Authentication for IT Security: From Fundamentals to Handwrinting*. Springer, 2006.

[35] P. A. Viola and M. J. Jones. Rapid object detection using a boosted cascade of simple features. *Proceedings of the 2001 IEEE Computer Society Conference Vol.1*, 2001.

Apêndice A

A1. PCA.py

```
# -*- coding: utf-8 -*-
'''
Created on Apr 9, 2015

@author: Joel Silvestre
'''

import numpy as np
from numpy import linalg as LA
import pylab
import Image

def PCA(trainingImages):
        #trainingImages is cv2 objects
        (imageHeight, imageWidth) = get_dimensions(trainingImages)
        N = imageWidth * imageHeight
        M = len(trainingImages)
        #Images to Vector
        vectorImages = []
        for i in xrange(M):
                imagePIL = Image.fromarray(trainingImages[i])
                vectorImages.append([pix for pix in imagePIL.getdata()])

        #Matriz Gama
        gamma = np.zeros((M, N)) #N - Number of Rows, M - Number of Columns
        for i in xrange(M):
                vectorPixels = np.asfarray(vectorImages[i])      #asfarray because
    float numbers
                gamma[i] = vectorPixels
        #Matriz Media #axis=0 Columns #axis=1 Rows
        psi = np.mean(gamma, axis=0) #return average
        #Matriz A of phi's     #Subtract psi in gamma
        A = np.subtract(gamma, psi)     #phi = gamma - psi
        A = A.T
        #Matriz Covariancia
        C = np.cov(A, rowvar=False)
```

```
        #Ordered eigen vectors and eigen values
        eigenvals, eigenvects = LA.eigh(C)
        idx = np.argsort(eigenvals)[::-1]
        w_dec = eigenvals[idx]
        v_dev = eigenvects[idx]
        #Matriz U with ordered eigenvectors by highest eigenvalues
        U = np.dot(A, v_dev).T
        #Normalize U
        for i in xrange(M):
                normU = U[i]
                normU.shape = (imageHeight, imageWidth)
                U[i] = U[i] / np.trace(np.dot(normU.T, normU))
        #return (U, v_dev, w_dec, psi, A, gamma)
        return (U, psi, A, gamma)

def get_dimensions(trainingImages):
        image = trainingImages[0]
        return image.shape

def draw_image(image, imageWidth, imageHeight):
        imgDraw = image.reshape(imageHeight, imageWidth)
        pylab.figure()
        pylab.gray()
        pylab.imshow(imgDraw)
        pylab.show()
```

A2. facedetect.py

```
# -*- coding: utf-8 -*-
'''
Created on Apr 9, 2015

@author: Joel Silvestre
'''

import cv2
from PIL import Image
import sic
import utils.utils as utils

face_cascade = cv2.CascadeClassifier('//usr//share//opencv//haarcascades//
    haarcascade_frontalface_default.xml')
eye_cascade = cv2.CascadeClassifier('//usr//share//opencv//haarcascades//
    haarcascade_eye.xml')


def detect_face(image_gray_to_detect):
        faces = face_cascade.detectMultiScale(image_gray_to_detect, 1.3, 5)
        resultFacePosition = ()
        if(len(faces) == 1):
```

```
                for (x,y,w,h) in faces:
                        resultFacePosition = (x,y,w,h)
                        #cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
                        #print "X: {0}, Y: {1}, W: {2}, H: {3}".format(x, y, w, h)
        else:
                print "Erro: Existe mais do que uma face na imagem."
        return resultFacePosition

def detect_eyes(image_gray_to_detect, omega_detect=0):
        resultFacePosition = []
        roi_gray = 0
        if omega_detect == 0:
                roi_gray = image_gray_to_detect
        else:
                (x,y,w,h) = omega_detect
                roi_gray = image_gray_to_detect[y:y+h, x:x+w]
        eyes = eye_cascade.detectMultiScale(roi_gray)
        if len(eyes) == 2:
                for (ex,ey,ew,eh) in eyes:
                        resultFacePosition.append((ex,ey,ew,eh))
                        #cv2.rectangle(roi_gray,(ex,ey),(ex+ew,ey+eh),(0,255,0),1)
                #cv2.imshow('img',roi_gray)
                #cv2.waitKey(0)
        elif len(eyes) != 1:
                eyesSorted = []
                for x in eyes:
                        (xs, ys, ws, hs) = x
                        eyesSorted.append((ys, xs, ws, hs))
                eyesSorted = sorted(eyesSorted)[:2]
                for (ey,ex,ew,eh) in eyesSorted:
                        resultFacePosition.append((ex,ey,ew,eh))
        return resultFacePosition

def alignEyes(s, tx, ty, img, omega):
        iter = 0
        while iter < 100:
                iter = iter + 1
                W = sic.warp_matrix(s, tx, ty)
                img_cropped = utils.croppedTransformation(img, W, omega)

                eyes = detect_eyes(img_cropped)
                (hi, wi) = img_cropped.shape
                if eyes == []:
                        return sic.warp_matrix(s, tx, ty)

                (ex0,ey0,ew0,eh0) = eyes[0]
                (ex1,ey1,ew1,eh1) = eyes[1]

                pointLeftEye, pointRightEye = 0, 0
                if(ex0>ex1):
                        pointLeftEye = (ex1+(ew1/2),ey1+(eh1/2))
                        pointRightEye = (ex0+(ew0/2),ey0+(eh0/2))
```

```
                else:
                        pointLeftEye = (ex0+(ew0/2),ey0+(eh0/2))
                        pointRightEye = (ex1+(ew1/2),ey1+(eh1/2))

                LeftDistanc = utils.dist(pointLeftEye, (0, pointLeftEye[1]))
                RightDistanc = utils.dist(pointRightEye, (wi, pointRightEye[1]))

                if(LeftDistanc != RightDistanc and LeftDistanc > RightDistanc):
                        tx=tx+0.5
                elif(LeftDistanc != RightDistanc and LeftDistanc < RightDistanc):
                        tx=tx-0.5
                else:
                        break
        return W
```

A3. SIC.py

```
# -*- coding: utf-8 -*-
'''
Created on Jul 14, 2015

@author: Joel Silvestre
'''

import numpy as np
import cv2
import math
import cv2.cv as cv
from PIL import Image

def warp_matrix(s, tx, ty):
        W = [[1, -s, tx],
                [s, 1, ty],
                [0, 0, 1]]
        return np.asarray(W, dtype=np.float32)

def bilinear_interpolate(im, x, y):
        return cv2.remap(im,x,y,cv2.INTER_LINEAR)[0][0]

def steepest_descent(X, gX, gY):
        stdesc = np.zeros((3, 1), dtype=np.float32)
        stdesc[0] = (-X[1]*gX+X[0]*gY)
        stdesc[1] = (gX)
        stdesc[2] = (gY)
        return stdesc

def get_vector(u, v):
        X = np.zeros((3, 1), dtype=np.float32)
        X[0] = u
        X[1] = v
```

```
        X[2] = 1
        return X

def update_vector(a, SD, E):
        a[0] += SD[0] * E
        a[1] += SD[1] * E
        a[2] += SD[2] * E

def update_hessian(hessian_matrix, SD):
        for l in xrange(3):
                for m in xrange(3):
                        hessian_matrix[l][m] += SD[l]*SD[m]


def align_SIC_image(templateI, sourceI, omega, show=False):
        last_error, last_s, last_tx, last_ty = 0, 0, 0, 0
        newImage = np.copy(sourceI)

        (imgHeight, imgWidth) = sourceI.shape

        gradientTemplateX = cv2.Sobel(sourceI, cv2.CV_16S, 1, 0, scale=1.0)
        gradientTemplateY = cv2.Sobel(sourceI, cv2.CV_16S, 0, 1, scale=1.0)

        s, tx, ty, mean_error = 0, 0, 0, 0
        inverseH = np.zeros((3, 3), dtype=np.float32)
        pixel_source = np.zeros((3, 1), dtype=np.float32)
        a = np.zeros((3, 1), dtype=np.float32)
        W = np.zeros((3, 3), dtype=np.float32)

        iter = 0
        iterations = 100
        while(iter < iterations):
                iter = iter + 1
                error, pixel_count = 0, 0

                W = warp_matrix(s, tx, ty)
                hessian_matrix = np.zeros((3, 3), dtype=np.float32)

                omegaX, omegaY = omega[0], omega[1]
                heightI, widthI = omega[3], omega[2]
                for i in xrange(widthI):
                        x = i + omegaX
                        for j in xrange(heightI):
                                y = j + omegaY
                                pixel_template = get_vector(x, y)
                                cv2.gemm(W, pixel_template, 1, 0, 0, pixel_source)
                                (calculatedU, calculatedV, inv) = pixel_source
                                floorU = cv.Floor(calculatedU)
                                floorV = cv.Floor(calculatedV)

                                #Se o valores de u e v se encontram dentro da
    imagem
```

```
                                if(floorU >= 0 and floorU<imgWidth and floorV >= 0
    and floorV<imgHeight):
                                        pixel_count = pixel_count +1
                                        gX = float(bilinear_interpolate(
    gradientTemplateX, calculatedU, calculatedV))
                                        gY = float(bilinear_interpolate(
    gradientTemplateY, calculatedU, calculatedV))
                                        #Calcular o steepest descent
                                        SD = steepest_descent(pixel_template, gX,
    gY)
                                        #Calculate intensity of a transformed pixel
                                        I = float(bilinear_interpolate(sourceI,
    calculatedU, calculatedV))
                                        #Calcular o erro
                                        E = float(templateI[int(j), int(i)] - I)
                                        error += math.fabs(E)
                                        #Atualizar valores da matrix b
                                        update_vector(a, SD, E)
                                        #Atualizar matrix hessiana
                                        update_hessian(hessian_matrix, SD)
                if(pixel_count!=0):
                        error /= pixel_count
                cv2.invert(hessian_matrix, inverseH)
                #inverseH = np.linalg.inv(H)
                delta_p = np.zeros((3, 1), dtype=np.float32)
                cv2.gemm(inverseH, a, 1, 0, 0, delta_p)
                s += delta_p[0]
                tx += delta_p[1]
                ty += delta_p[2]

                if(show):
                        W = warp_matrix(last_s, last_tx, last_ty)
                        warped_image(newImage, omega, W)

                if last_error < mean_error and iter != 1:
                        return (last_s, last_tx, last_ty, last_error, iter-1)
                else:
                        last_error = mean_error
                        last_s = s
                        last_tx = tx
                        last_ty = ty
        pass

def warped_image(imageSource, rect, W):
        import pylab
        image = np.copy(imageSource)

        pixel = get_vector(rect[0], rect[1])
        Z = cv2.gemm(W, pixel, 1, 0 ,0)
        (x1, y1, inv) = Z.astype(int)

        pixel = get_vector(rect[0], rect[1]+rect[3])
```

```
        Z = cv2.gemm(W, pixel, 1, 0 ,0)
        (x2, y2, inv) = Z.astype(int)

        pixel = get_vector(rect[0]+rect[2], rect[1])
        Z = cv2.gemm(W, pixel, 1, 0 ,0)
        (x3, y3, inv) = Z.astype(int)

        pixel = get_vector(rect[0]+rect[2], rect[1]+rect[3])
        Z = cv2.gemm(W, pixel, 1, 0 ,0)
        (x4, y4, inv) = Z.astype(int)

        cv2.line(image,(x1,y1),(x3,y3),(255,0,0),1)
        cv2.line(image,(x3,y3),(x4,y4),(255,0,0),1)
        cv2.line(image,(x4,y4),(x2,y2),(255,0,0),1)
        cv2.line(image,(x2,y2),(x1,y1),(255,0,0),1)

        import transform as transform

        pts = np.array([(float(x1), float(y1)), (float(x3), float(y3)), (float(x4),
    float(y4)), (float(x2), float(y2))], dtype = "float32")

        warped = transform.four_point_transform(image, pts)

        pylab.figure()
        pylab.gray()
        pylab.imshow(image)
        pylab.show()
        pylab.close()

        pylab.figure()
        pylab.gray()
        pylab.imshow(warped)
        pylab.show()
        pylab.close()

        return warped
```

A4. classificator.py

```
# -*- coding: utf-8 -*-
'''
Created on Jun 25, 2015

@author: Joel Silvestre
'''

import math
import numpy as np
import Image
import pylab
```

```
def classifier(U, mean, A, gamma, templateImage, threshold=3.0):
        ##########################################
        img = Image.fromarray(templateImage)
        imageProva = np.asfarray([pix for pix in img.getdata()])

        imageT = imageProva - mean

        #-----------------------------------------------------------------
        omega = np.dot(U[:3,:], imageT.T).T

        weights = np.dot(U[:3,:], A).T

        dist = np.sum(np.power(weights - omega, 2), axis=1)
        idx = np.argmin(dist)

        mindist = math.sqrt(dist[idx])
        print mindist
        if mindist < threshold:
                return True
        else:
                return False
```

A5. SIC.cpp

```
#include <Python.h>
#include <stdio.h>
#include "SIC.h"
#include <cv.h>
#include <highgui.h>

void warpMatrix(CvMat* warp_matrix, float shear, float translate_x, float
    translate_y)
{
        CV_MAT_ELEM(*warp_matrix, float, 0, 0) = 1;
        CV_MAT_ELEM(*warp_matrix, float, 1, 0) = shear;
        CV_MAT_ELEM(*warp_matrix, float, 2, 0) = 0;

        CV_MAT_ELEM(*warp_matrix, float, 0, 1) = -shear;
        CV_MAT_ELEM(*warp_matrix, float, 1, 1) = 1;
        CV_MAT_ELEM(*warp_matrix, float, 2, 1) = 0;

        CV_MAT_ELEM(*warp_matrix, float, 0, 2) = translate_x;
        CV_MAT_ELEM(*warp_matrix, float, 1, 2) = translate_y;
        CV_MAT_ELEM(*warp_matrix, float, 2, 2) = 1;
}

warpV alignSIC(const char*  templateL, const char*  sourceL, int omega_x, int
    omega_y, int omega_w, int omega_h)
{
```

```
    IplImage* templateI = cvLoadImage(templateL, CV_LOAD_IMAGE_GRAYSCALE);
    IplImage* sourceI = cvLoadImage(sourceL, CV_LOAD_IMAGE_GRAYSCALE);
    CvSize size_image = cvSize(sourceI->width,sourceI->height);

    warpV warp_values;
    //const float threshold = 0.01;
    CvMat* tranformation_matrix = cvCreateMat(3, 3, CV_32F);
    CvMat* pixel_template = cvCreateMat(3, 1, CV_32F);
    CvMat* pixel_source  = cvCreateMat(3, 1, CV_32F);
    CvMat* param_a = cvCreateMat(3, 1, CV_32F);
    CvMat* param_delta_a = cvCreateMat(3, 1, CV_32F);
    CvMat* hessian_matrix = cvCreateMat(3, 3, CV_32F);
    CvMat* inverse_hessian_matrix = cvCreateMat(3, 3, CV_32F);

    IplImage* gradientTemplateX = cvCreateImage(size_image, IPL_DEPTH_16S, 1);
    IplImage* gradientTemplateY = cvCreateImage(size_image, IPL_DEPTH_16S, 1);

    cvSobel(sourceI, gradientTemplateX, 1, 0);
    cvConvertScale(gradientTemplateX, gradientTemplateX);
    cvSobel(sourceI, gradientTemplateY, 0, 1);
    cvConvertScale(gradientTemplateY, gradientTemplateY);

    float error= 0, shear=0, translate_x=0, translate_y=0;
    float last_error=0, last_shear=0, last_translate_x=0, last_translate_y=0;
    int iter = 0;
    int iterations = 100;

    while(iter < iterations){
            iter++;
            error=0;

            warpMatrix(tranformation_matrix, shear, translate_x, translate_y);

            cvSet(hessian_matrix, cvScalar(0));
            cvSet(param_a, cvScalar(0));

            int pixel_count=0;
            int x, y, i, j;
            for(i=0; i<omega_w; i++){
                    x = i + omega_x;
                    for(j=0; j<omega_h;j++){
                            y = j + omega_y;

                            SET_VECTOR(pixel_template, x, y);
                            cvGEMM(tranformation_matrix, pixel_template, 1, 0,
0, pixel_source);

                            float warp_x, warp_y;
                            GET_VECTOR(pixel_source, warp_x, warp_y);

                            int floor_warp_x = cvFloor(warp_x);
                            int floor_warp_y = cvFloor(warp_y);
```

```
                            if(floor_warp_x>=0 && floor_warp_x<sourceI->width
&& floor_warp_y>=0 && floor_warp_x<sourceI->height){
                                    float gX = interpolate<short>(
gradientTemplateX, warp_x, warp_y);
                                    float gY = interpolate<short>(
gradientTemplateY, warp_x, warp_y);

                                    float SD[3];
                                    SD[0] = (float)(-y*gX+x*gY);
                                    SD[1] = (float)gX;
                                    SD[2] = (float)gY;

                                    float I = interpolate<uchar>(sourceI,
warp_x, warp_y);
                                    float E = CV_IMAGE_ELEM(templateI, uchar, j
, i) - I;

                                    error += fabs(E);

                                    float* a = &CV_MAT_ELEM(*param_a, float, 0,
 0);
                                    a[0] += SD[0] * E;
                                    a[1] += SD[1] * E;
                                    a[2] += SD[2] * E;

                                    int l,m;
                                    for(l=0;l<3;l++){
                                            for(m=0;m<3;m++){
                                                    CV_MAT_ELEM(*hessian_matrix
, float, l, m) += SD[l]*SD[m];
                                            }
                                    }
                            }
                    }
            }
            if(pixel_count!=0)
                    error /= pixel_count;
            cvInvert(hessian_matrix, inverse_hessian_matrix);
            cvGEMM(inverse_hessian_matrix, param_a, 1, 0, 0, param_delta_a);
            float delta_shear = CV_MAT_ELEM(*param_delta_a, float, 0, 0);
            float delta_translate_x = CV_MAT_ELEM(*param_delta_a, float, 1, 0);
            float delta_translate_y = CV_MAT_ELEM(*param_delta_a, float, 2, 0);

            shear += delta_shear;
            translate_x += delta_translate_x;
            translate_y += delta_translate_y;

            if(last_error < error && iter != 1){
                    warp_values.shear=last_shear;
                    warp_values.translate_x=last_translate_x;
                    warp_values.translate_y=last_translate_y;
                    return warp_values;
```

```
                }else{
                        last_error = error;
                        last_shear = shear;
                        last_translate_x = translate_x;
                        last_translate_y = translate_y;
                }
        }
        cvReleaseMat(&tranformation_matrix);
        cvReleaseMat(&pixel_template);
        cvReleaseMat(&pixel_source);
        cvReleaseMat(&hessian_matrix);
        cvReleaseMat(&inverse_hessian_matrix);
        cvReleaseMat(&param_a);
        cvReleaseMat(&param_delta_a);

        cvReleaseImage(&templateI);
        cvReleaseImage(&sourceI);
        cvReleaseImage(&gradientTemplateX);
        cvReleaseImage(&gradientTemplateY);

        warp_values.shear=shear;
        warp_values.translate_x=translate_x;
        warp_values.translate_y=translate_y;
        return warp_values;
}
```

A6. TIPCA.py

```
# -*- coding: utf-8 -*-
'''
Created on Jun 25, 2015

@author: Joel Silvestre
'''
import pca
import sic
import pylab
import utils.utils as utils
import facedetect
import scipy.misc
import algorithms_c.SIC as SIC

IMAGE_BASE = "source.jpg"
IMAGE_TEMPLATE = "template.jpg"
def enrollment_TICPA(training_images, omega_training, WITH_ALIGN=True,
    RUN_C_ALGORITHM=True):
        (resize_w, resize_h) = utils.min_dimension(omega_training)
        omega_training_new = []
        for x in xrange(len(omega_training)):
                omega_training_new.append((omega_training[x][0], omega_training[x
    ][1], resize_w, resize_h))
```

```
        cropped_training_images = utils.croppedImages(training_images,
    omega_training_new)

        (U, mean, A, gamma0) = pca.PCA(cropped_training_images)
        eigenface = (U[4]+mean).reshape(resize_h, resize_w)

        count = 0
        training_images_W = []
        if(RUN_C_ALGORITHM):
                scipy.misc.imsave(IMAGE_TEMPLATE, eigenface)
        for img in training_images:
                if(WITH_ALIGN):
                        #Alinha todas as imagens de base para voltar a efetuar PCA
                        if(RUN_C_ALGORITHM):
                                scipy.misc.imsave(IMAGE_BASE, img)
                                (xc, yc, wc, hc) = omega_training_new[count]
                                warp_values = SIC.alignSIC(IMAGE_TEMPLATE,
    IMAGE_BASE, int(xc),int(yc),int(wc),int(hc))
                                (s, tx, ty) = (warp_values.shear, warp_values.
    translate_x, warp_values.translate_y)
                        else:
                                (s, tx, ty, mean_error, iter) = sic.align_SIC_image
    (eigenface, img, omega_training_new[count])
                        #Mediante a transformacao efetuada por SIC e feito o
    alinhamento com base nos olhos da pessoa
                        W_align_eyes = facedetect.alignEyes(s, tx, ty, img,
    omega_training_new[count])
                        #E apenas apresentada a imagem e desenhado um quadrado para
     constatar resultados
                        #sic.warped_image(img, omega_training[count], W_align_eyes)
                        #E feita a transformacao da imagem e cortada para que possa
     servir para calcular PCA novamente
                        img_cropped = utils.croppedTransformation(img, W_align_eyes
    , omega_training_new[count])
                else:
                        W = sic.warp_matrix(0, 0, 0)
                        img_cropped = utils.croppedTransformation(img, W,
    omega_training_new[count])
                training_images_W.append(utils.resize(img_cropped, (resize_w,
    resize_h)))
                count = count+1
        return (eigenface, training_images_W, (resize_w, resize_h))

def identification_TIPCA(cropped_images_w, normalizer_image, omega, dimentions,
    WITH_ALIGN=True, RUN_C_ALGORITHM=True):
        #E efetuado novamente o PCA com todas as imagens alinhadas
        (U, mean, A, gamma) = pca.PCA(cropped_images_w)
        (w, h) = dimentions
        eigenface = (U[4]+mean).reshape(h, w)

        omega = (omega[0], omega[1], w, h)
```

```
    W = sic.warp_matrix(0, 0, 0)
    #sic.draw_warped_rect(eigenface, omega, W)
    if(WITH_ALIGN):
            if(RUN_C_ALGORITHM):
                    scipy.misc.imsave(IMAGE_TEMPLATE, eigenface)
                    scipy.misc.imsave(IMAGE_BASE, normalizer_image)
                    (xc, yc, wc, hc) = omega
                    warp_values = SIC.alignSIC(IMAGE_TEMPLATE, IMAGE_BASE, int(
xc),int(yc),int(wc),int(hc))
                    (s, tx, ty) = (warp_values.shear, warp_values.translate_x,
warp_values.translate_y)
            else:
                    #Alinha a imagem de prova para posterior classificacao
                    (s, tx, ty, mean_error, iter) = sic.align_SIC_image(
eigenface, normalizer_image, omega)
            #Alinha a imagem de prova tento como referencias a localizacao dos
olhos
            W_align_eyes = facedetect.alignEyes(s, tx, ty, normalizer_image,
omega)
            #E apenas apresentada a imagem e desenhado um quadrado para
constatar resultados
            #sic.warped_image(normalizer_image, omega, W_align_eyes)
            #E feita a transformacao da imagem e cortada para que possa
posteriormente ser classificada
            img_cropped = utils.croppedTransformation(normalizer_image,
W_align_eyes, omega)
    else:
            W = sic.warp_matrix(0, 0, 0)
            img_cropped = utils.croppedTransformation(normalizer_image, W,
omega)
    return (U, mean, A, gamma, utils.resize(img_cropped, dimentions))
```

Apêndice B

B1. Arquitetura da fase de registo

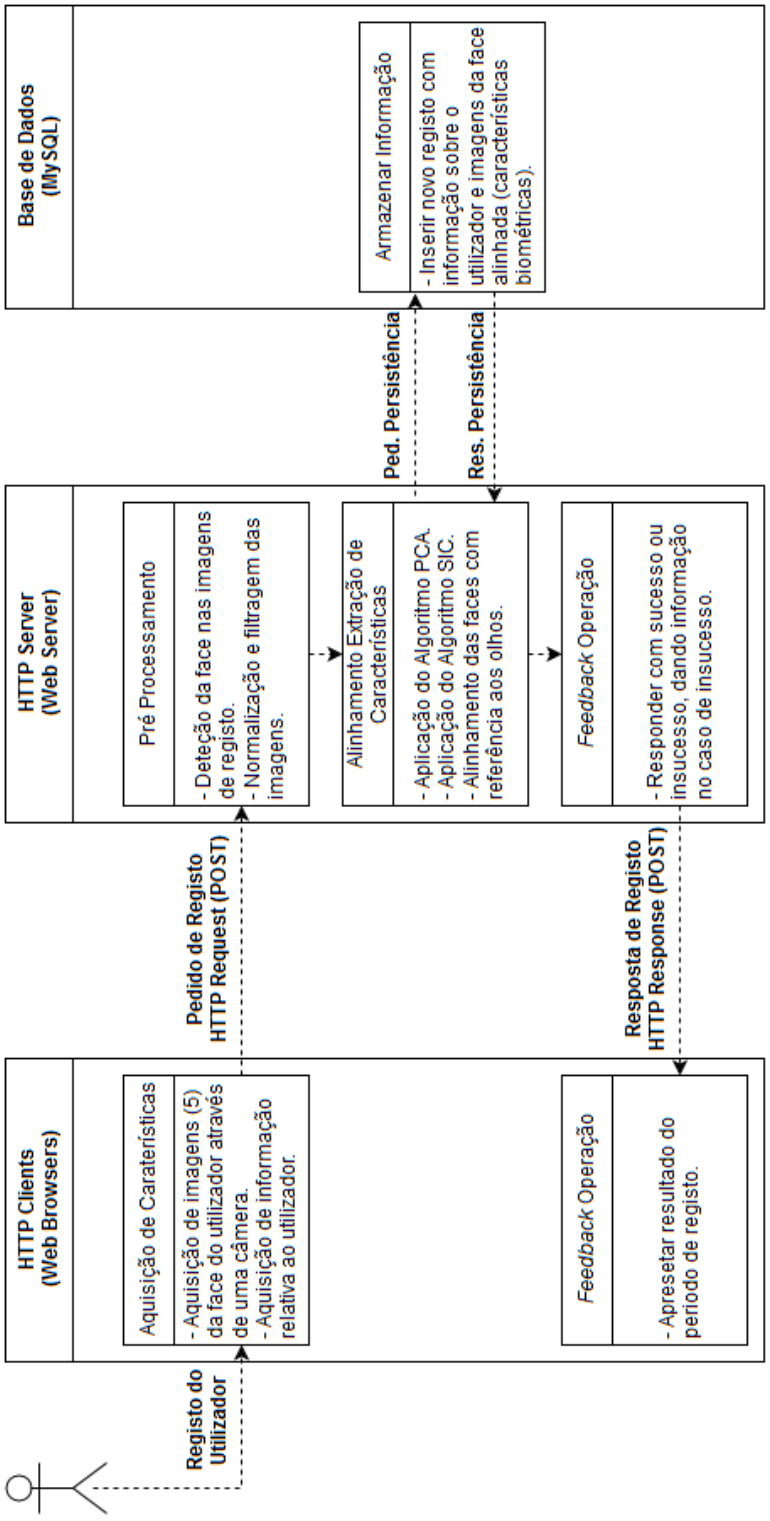


FIGURA 7.1: Arquitetura da fase de registo.

B2. Arquitetura da fase de autenticação

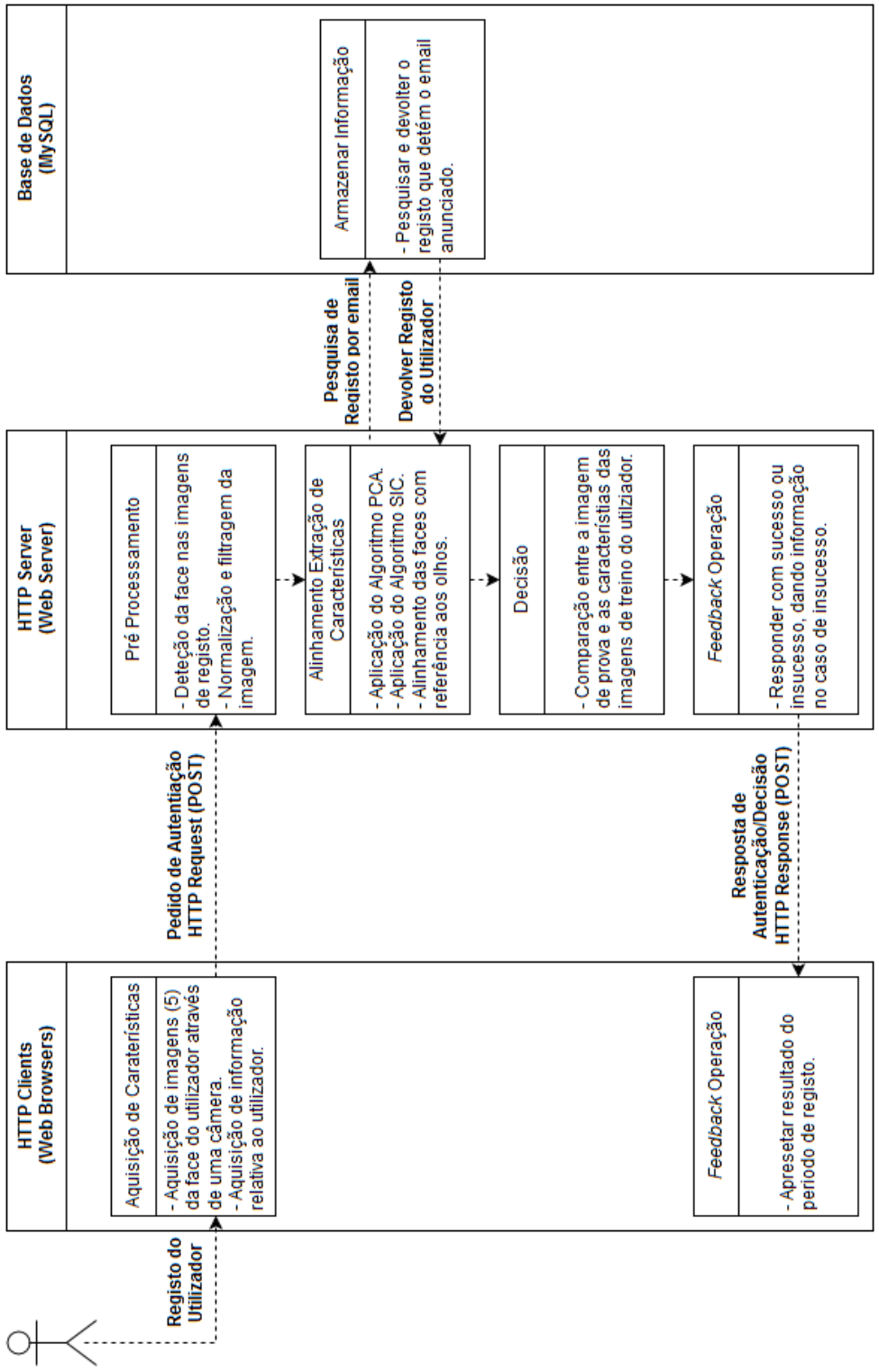


FIGURA 7.2: Arquitetura da fase de autenticação.