UNIVERSIDADE FEDERAL DO MARANHÃO
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

WHESLEY CUNHA DANTAS

SOLUÇÃO DE HARDWARE E SOFTWARE PARA RASTREAMENTO DE FACE EM TEMPO REAL

São Luís
2013

WHESLEY CUNHA DANTAS

SOLUÇÃO DE HARDWARE E SOFTWARE PARA RASTREAMENTO DE FACE EM TEMPO REAL

Monografia apresentada ao curso de Ciência da Computação da Universidade Federal do Maranhão, como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. João Dallyson Sousa de Almeida

São Luís
2013

Dantas, Whesley Cunha.

Solução de hardware e software para rastreamento de face em tempo real/ Whesley Cunha Dantas. – São Luís, 2013.

43 f.

Impresso por computador (fotocópia).

Orientador: João Dallyson Sousa de Almeida.

Monografia (Graduação) – Universidade Federal do Maranhão, Curso de Ciência da Computação, 2013.

1. Hardware – software. 2. Detecção facial. 3. Rastreamento facial. I. Título.

CDU 004..3/4:611.92

WHESLEY CUNHA DANTAS

SOLUÇÃO DE HARDWARE E SOFTWARE PARA RASTREAMENTO DE FACE EM TEMPO REAL

Monografia apresentada ao curso de Ciência da Computação da Universidade Federal do Maranhão, como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Aprovada em: 13 / 12 / 2013

BANCA EXAMINADORA

Prof. Dr. João Dallyson Sousa de Almeida (Orientador)
Doutor em Engenharia da Eletricidade
Universidade Federal do Maranhão

Prof. Dr. Anselmo Cardoso de Paiva
Doutor em Informática
Universidade Federal do Maranhão

Prof. Dr. Aristófanes Correa Silva
Doutor em Informática
Universidade Federal do Maranhão

A Deus, fonte de vida.
A meus pais, Silvana Cunha e Carlos Dantas,
pelo apoio e carinho constantes.

AGRADECIMENTOS

Agradeço primeiramente a Deus, por me abençoar nessa caminhada e permitir que conseguisse chegar até o final.

Ao meu Pai, que acreditou no meu potencial, e sempre esteve disposto a me ajudar em tudo que fosse necessário.

À minha família, em especial, minha mãe, irmãs e avó que foram compreensivas nos momentos difíceis e incentivaram o meu crescimento pessoal e profissional.

Aos amigos do curso por esse período de convivência, troca de conhecimento e experiências tanto boas quanto más, que contribuíram para o meu desenvolvimento.

Aos meus professores, pelo conhecimento que me proporcionaram, e principalmente, aos meus orientadores Anselmo e João Dallyson, pela ajuda nessa fase final do curso.

E a todos que de alguma forma contribuíram para que a realização desse trabalho fosse possível.

“Um homem livre é aquele que, tendo força e talento para fazer uma coisa, não encontra barreiras a sua vontade”.

(Thomas Hobbes)

RESUMO

Uma das áreas da tecnologia em constante crescimento é a área da automação que envolve recursos de hardware e software para um controle automático de equipamentos e sistemas. Devido ao constante crescimento, surgiram várias técnicas e ferramentas que auxiliam o desenvolvimento dos sistemas automatizados. Uma dessas ferramentas é o Arduino, que é uma plataforma de prototipagem eletrônica de hardware livre, projetada com um microcontrolador Atmel AVR e tem suporte de entrada/saída embutido. Através dele, é possível criar sistemas de hardware e software que são acessíveis, flexíveis, fáceis de manusear e com baixo custo.

Desde o seu lançamento, foi possível então notar a grande quantidade de sistemas de automação já criados através da utilização do Arduino. Desta forma, este projeto tem por finalidade a implementação de um sistema de rastreamento que faz a detecção da face humana através de uma câmera de captura e um software que utiliza a biblioteca OpenCV. A câmera, montada em um mecanismo dotado de recursos de posicionamento, acompanha automaticamente o deslocamento da face que é enquadrada pelo software de detecção e este, por sua vez, calcula as coordenadas da face. Em seguida, essas coordenadas são enviadas do computador para a porta serial do Arduino que contém, em seu microcontrolador (Atmel AVR), um software desenvolvido para controlar, em tempo real, dois servomotores que garantem o posicionamento dessa câmera sobre o objeto de interesse.

Palavras-chave: Visão computacional. Detecção Facial. Rastreamento de faces. Arduino. Servomotores.

ABSTRACT

One area of technology in constantly growing is the area of the automation that involves hardware and software resources for automatic control of equipment and systems. Based on this steady growth, have been emerged techniques and tools that helps the development of automated systems. One such tool is the Arduino, which is a electronics prototyping platform, of free hardware, designed with an Atmel AVR microcontroller and with input / output support. With it, is possible to create hardware and software systems that is affordable, flexible, easy to use, and with low cost.

Since its release, it was possible to realize the large amount of automation systems ever created by using the Arduino. Thus, the purpose of this project is to implement a tracking system that makes the face detection through the camera and software, that uses the OpenCV library. The camera mounted on a mechanism with positioning capabilities, automatically follows the movement of the face which is framed by the detection system and this, in turn, calculates the coordinates of the face. Then, these coordinates are sent from the computer to the Arduino's serial port which contains, in its microcontroller (Atmel AVR), a software developed to control, in real time, two servomotors that ensure the camera positioning on the interest object.

Keywords: Computer vision. Face Detection. Face Tracking. Arduino. ServoMotors.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

A história nos relata grandes feitos realizados pelo homem ao longo dos anos, através de descobertas e invenções, frutos de sua curiosidade e conhecimento. A busca por melhorias tem permitido ao homem desenvolver novas tecnologias para facilitar o seu dia a dia.

Para aperfeiçoar seu conhecimento foi necessário estudar as leis que regem a natureza, desenvolver mecanismos, criar métodos, equipamentos, convenções, tudo com a finalidade de melhor o seu cotidiano. Como exemplo de avanço tecnológico, tem-se a evolução dos sistemas computacionais que causam grandes mudanças no meio social e influencia diversas áreas como: engenharia, medicina, meio ambiente e outras.

No setor da engenharia, implementações cada vez mais sofisticadas, permitem o desenvolvimento de novos dispositivos, sendo estes, cada vez mais eficientes e robustos. Dessa forma, a indústria, o comercio e até mesmo o âmbito residencial estão usufruindo das novas tecnologias e, como consequência, há um grande crescimento e investimento nos sistemas de segurança e monitoramento, acrescentando assim, melhores características ao sistema de vigilância.

Técnicas de detecção e rastreamento de objetos foram desenvolvidas no campo da visão computacional, uma vez que essa linha de pesquisa apresenta vários procedimentos que podem ser aplicados em projetos que envolvem sistemas de vigilância e monitoramento. Nessa perspectiva, o foco deste trabalho está associado a mais um desenvolvimento tecnológico que tem se destacado bastante e está em pleno crescimento - solução de hardware e software para rastreamento de face em tempo real, apresentando uma metodologia que seja capaz de realizar a busca da mesma a partir do vídeo de uma câmera instalada em um dispositivo de posicionamento, de forma a mantê-la sempre direcionada ao alvo.

1.1 Motivação

Desenvolver uma metodologia de hardware e de software, capaz de realizar a detecção e rastreamento facial e, com isso, ser utilizado em sistemas de segurança e nas mais diversas aplicações de visão computacional.

1.2 Objetivos

Este trabalho tem como objetivo o desenvolvimento de uma metodologia de software e de hardware para rastreamento de faces em vídeos em tempo real. Para isso, será desenvolvido um software que localiza a face no vídeo adquirido de uma câmera e que envia sinais a um circuito eletrônico para o controle de servomotores que serão responsáveis pelo posicionamento automático da câmera em relação à face do alvo.

Os objetivos específicos são listados:

1. Desenvolver uma solução de hardware para o controle de servomotores;
2. Desenvolvimento de um algoritmo que será instalado no microcontrolador do circuito de controle para posicionamento dos servomotores;
3. Desenvolvimento de um algoritmo para a detecção e localização da face em cada frame do vídeo;

1.3 Organização do Trabalho

A apresentação detalhada dos aspectos deste trabalho está distribuída da seguinte forma:

Neste primeiro capítulo foi realizada a introdução deste trabalho.

O segundo capítulo apresentará a fundamentação teórica com as informações necessárias para a contextualização, entendimento e desenvolvimento do trabalho.

O terceiro capítulo apresenta a metodologia que será utilizada como ponto de partida para o desenvolvimento e os passos seguidos para a execução deste trabalho.

O quarto capítulo apresenta a conclusão deste trabalho, bem como os resultados obtidos. Mostra-se a ferramenta (hardware e software) desenvolvida.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, aborda-se os conceitos necessários para entender o funcionamento da metodologia computacional proposta.

2.1 Fundamentos do processamento de imagens

Nesta seção, serão abordados alguns conceitos básicos sobre técnicas de processamento de imagens que também serão úteis neste trabalho para a detecção e o rastreamento facial.

2.1.1 Conceitos Básicos de Processamento de Imagens

Para entender o que é uma imagem digital, inicialmente faz-se uma breve abordagem sobre a forma de representar informações do mundo real (dados contínuos) em relação às informações que podem ser representadas e tratadas por um computador (dados discretos).

As informações que se pode adquirir ou perceber do mundo real são contínuas, isto é, podem ser representada matematicamente por funções contínuas. Assim, estamos falando de funções reais definidas para a reta (em R), onde para cada valor x existe um único valor f(x).

A representação de dados em um computador é binária, isto é, utiliza-se uma quantidade de bits 0 ou 1 para representar determinada informação. Quanto maior for a quantidade de bits utilizada na representação de dados, mais real será a informação. Por exemplo, utilizando-se um byte (8 bits) para representar uma informação, tem-se $2^8 = 256$ valores possíveis de se representar, variando de 0 a 255.

De forma simplificada, pode-se dizer que a representação de dados em um computador é discreta, que em relação à representação contínua, significa uma perda de dados. Em outras palavras, não é possível representar uma função contínua em um computador, pode-se apenas simulá-la.

Como apresentado anteriormente, os dados discretos que são representados em um computador são uma aproximação em relação a representação real. Para uma imagem digital não é diferente. Esta pode então ser definida como uma função bidimensional *f(x, y)*, em que *x* e *y* são coordenadas espaciais (plano), finitas e discretas de um ponto em um plano cartesiano e o valor (ou amplitude) de *f* em qualquer par de coordenadas (*x*, *y*) é chamada de *intensidade* ou *nível de cinza* desse ponto (GONZALEZ & WOODS, 2010).

Para representar uma imagem em um computador, é comum utilizar a representação matricial, ou seja, a imagem será uma matriz $M_{LxC.}$ Assim, esta terá L linhas (altura) por C colunas (largura). Cada célula dessa matriz será uma unidade da imagem, ou seja, um elemento pictórico ou *pixel* que conterá um valor de intensidade (cor do *pixel*).

A imagem também poderá ter uma profundidade, isto é, a quantidade de bits utilizada para representar a cor em cada *pixel*. Em uma imagem, por exemplo, representada em níveis de cinza, com profundidade 8, terá 8 bits por *pixel*. Assim, será possível representar 256 valores de 0 a 255, variando do mais escuro (preto) ao mais claro (branco) respectivamente. Já em uma imagem RGB (*Red, Green, Blue*), ou seja, colorida e com profundidade 8, ter-se-á 8 bits para vermelho, 8 bits para verde e 8 bits para azul por *pixel*.

Uma imagem representada apenas em níveis de cinza com 8 bits por *pixel* possui apenas uma camada ou canal. Já em imagens coloridas como é o caso das imagens em RGB - mais comum, pois existem outras representações de coloração além do RGB – com profundidade 8, cada *pixel* possuirá uma canal para vermelho, outra para verde e outra para azul, sendo que cada camada terá 8 bits de profundidade. Essas imagens então terão 3 camadas ou canais. Misturando-se assim os três canais, será possível representar os mais variados tipos de cores e intensidades além das cores primárias.

2.1.2 Histograma

No estudo da estatística, o histograma é uma distribuição de frequências de um conjunto de dados, isto é, um gráfico que representa o número de ocorrências dos dados de um conjunto.

Assim, o histograma de uma imagem é uma função do tipo $h(f_k) = n_k$, onde n_k é o número de *pixels* da imagem cujo nível corresponde a k e, f_k é o k–ésimo nível dentro do intervalo [0,P]. Para uma imagem em níveis de cinza, P normalmente é 255, pois a imagem possui 8 bits por *pixel*.

Aplicado a uma imagem digital, o histograma seria um gráfico gerado para representar o número de ocorrências das cores dos *pixels* onde, o eixo das abscissas representa a cor e, o eixo das ordenadas representa o número de ocorrências desta cor, ou seja, a frequência com que essa cor aparece na imagem.

De forma simplificada, utilizando-se uma imagem em níveis de cinza, isto é, uma imagem com apenas uma camada e com 8 bits por *pixel* (intensidade dos *pixels* variando de 0 a 255), o histograma servirá para mostrar a frequência com a qual cada cor aparece na

imagem. Assim, o eixo das abscissas representam as intensidades dos *pixels* de 0 a 255 e, o eixo das ordenadas representa quantas vezes determinada cor apareceu nessa imagem.

Através do histograma da imagem, é possível determinar a qualidade dos *pixels* quanto ao nível de contraste e quanto ao brilho de cada *pixel*, ou seja, indicando se a imagem é escura ou clara.

Uma dentre várias técnicas utilizadas em processamento de imagem com o objetivo de aumentar a qualidade de uma imagem para um futuro processamento, é a equalização do histograma, que serve para melhorar o contraste de uma imagem. Essa equalização consiste em dividir as frequências de cada *pixel* pelo número de *pixels* da imagem, de acordo com a seguinte equação:

$$p(f_k) = \frac{h(f_k)}{n} = \frac{n_k}{n} \quad (1)$$

onde:

$0 \leq f_k \leq 1$, f_k é a escala de cinza equalizada;

k = 0, 1, ..., P-1, (P é o número de níveis de cinza da imagem);

n - número total de *pixels* da imagem

$p(f_k)$ – probabilidade do *k-ésimo* nível de cinza;

n_k – número de *pixels* cujo nível de cinza corresponde a k;

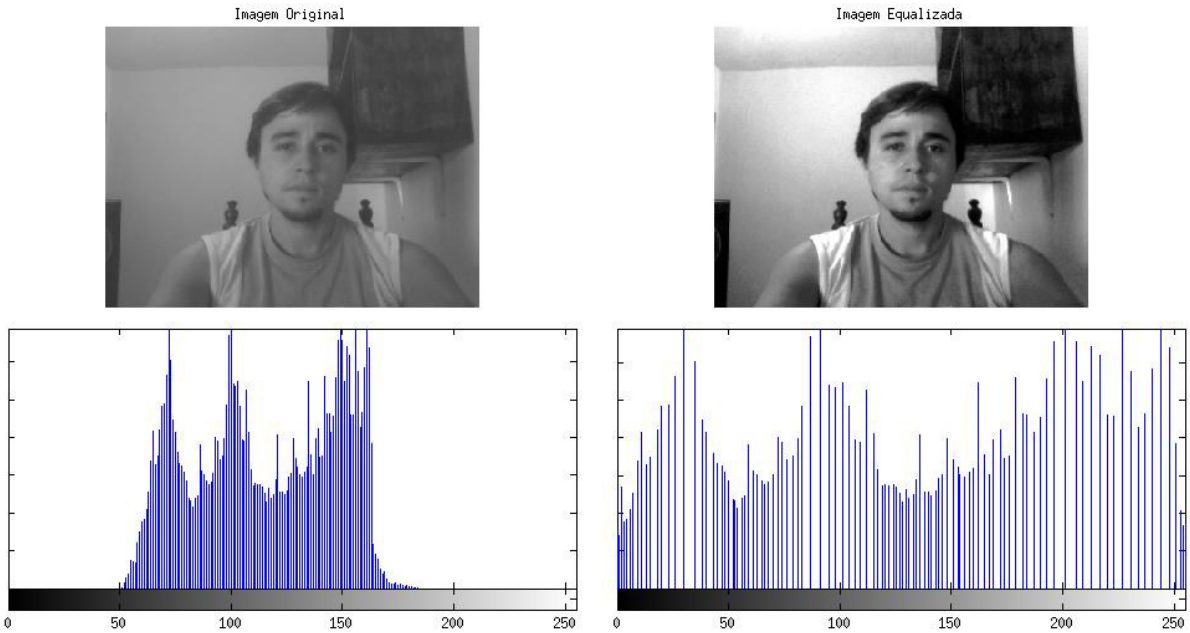

Figura 1 - Imagem normal e com histograma equalizado. Fonte: DANTAS, 2013

2.2 Algoritmo de Detecção Facial

A localização e segmentação da face que pode ser encontrada em uma imagem digital é o primeiro passo para que a etapa de rastreamento possa ser realizada, uma vez que a posição da face na imagem será utilizada como referência para as etapas posteriores. Esta etapa, de certo modo, é considerada crítica para as próximas, visto que uma falha nesta pode resultar em falhas no rastreamento.

Um dos algoritmos mais eficientes em relação a detecção de objetos em imagens através de detecção de padrões, é o algoritmo proposto inicialmente por Paul Viola e Michael Jones (VIOLA & JONES, 2004). Esse algoritmo inova pela rapidez de processamento e, é largamente utilizado no campo da visão computacional para detecção facial em imagens digitais.

Os autores dividem o trabalho em três partes principais. A primeira baseia-se no uso de *HaarFeatures* (características *Haar*) básicas, avaliadas rapidamente por meio de uma nova forma de representação a partir da imagem original, sendo chamada de *Imagem Integral*. A segunda baseia-se na aplicação de um algoritmo de aprendizado *boosting* chamado de *AdaBoost* (FREUND & SCHAPIRE, 1995), através do qual é possível filtrar todo o conjunto com o objetivo de diminuir o número de características geradas e que serão analisadas, isto é, o *AdaBoost* será utilizado para selecionar as características mais úteis de um conjunto vasto de características em potencial. A terceira e última etapa é um método que combina classificadores em forma de cascata que, desta forma, permite a rápida eliminação de regiões - que não agregarão tanto valor, como por exemplo, regiões de fundo - e maior rapidez no processamento das prováveis regiões de face (VIOLA & JONES, 2004).

No intuito de computar as *HaarFeatures* de forma eficiente, utiliza-se a imagem integral que é uma representação intermediária em relação à imagem original, como mencionado anteriormente. Para gerar essa imagem, o valor de cada *pixel* (x,y) será o resultado da soma de todos os *pixels* acima e à esquerda do *pixel* atual. Formalmente, sendo *i* a imagem original e *ii* a imagem integral, tem-se a seguinte equação:

$$ii(x,y) = \sum_{x' \leq x,\ y' \leq y} i(x', y') \tag{2}$$

Para se calcular e gerar a imagem integral em apenas um passo a partir da imagem original pode-se usar a seguinte operação:

$$s(x,y) = s(x,y-1) + i(x,y) \quad (3)$$

$$\mathrm{ii}(\mathrm{x},\mathrm{y}) = \mathrm{ii}(\mathrm{x}-1,\mathrm{y}) + \mathrm{s}(\mathrm{x},\mathrm{y}) \quad (4)$$

onde i é a imagem original, ii é a imagem integral, s(x,y) é a soma cumulativa da linha.

A ideia em utilizar a imagem integral, é usar esta representação intermediária para aumentar a velocidade de extração de características. Isso ocorre porque com a imagem integral, qualquer retângulo na imagem original pode ser calculado por meio da imagem intermediária. Desta forma, para calcular o somatório de qualquer retângulo, utiliza-se apenas o valor dos *pixels* que representam os quatro vértices deste retângulo, ou seja, apenas quatro índices da imagem integral, como ilustrado na Figura 2. Assim, será necessária apenas uma única passagem para obter dados desejados em sub-regiões de uma imagem.

O cálculo da região D ilustrada na Figura 2 pode ser representado pelas operações a seguir:

$$\mathrm{D} = \mathrm{ii}(4) - \mathrm{ii}(2) - \mathrm{ii}(3) + \mathrm{ii}(1) \quad (5)$$

$$\mathrm{D} = \mathrm{ii}(4) + \mathrm{ii}(1) - (\mathrm{ii}(2) + \mathrm{ii}(3)) \quad (6)$$

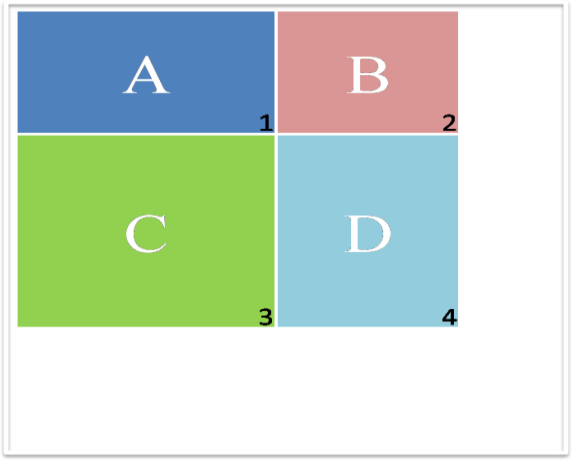

Figura 2 - Regiões calculadas a partir de uma imagem integral. Fonte: DANTAS, 2013

Considerando as regiões da imagem exibida na Figura 2 e o valor do *pixel* na posição 4 equivale à soma das somas dos *pixels* das regiões *A, B, C* e *D*. O *pixel* na posição 1 equivale a soma dos *pixels* da região *A*, o da posição 2 às regiões *A* e *B* e o da posição 3 às regiões *A* e *C*. Desta forma, resumidamente, a soma de *pixels* na região *D* poderá ser calculada com uma simples operação aritmética: (4 + 1 – (2 + 3)).

Como mensionado anteriormente, a imagem integral é utilizada para computar rapidamente o valor das regiões retangulares, as *HaarFeatures*.

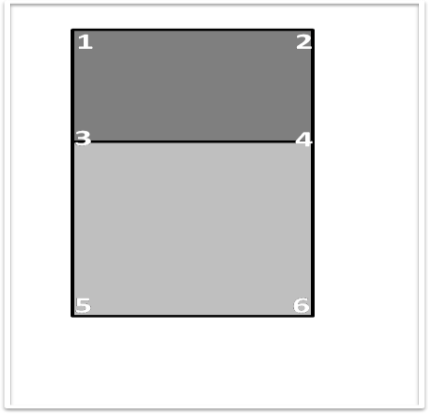

Figura 3 - Cálculo de uma Feature. Fonte: DANTAS, 2013

O valor de um *feature,* como pode ser visto na Figura 3, sobre uma janela é dado por:

$$Área\_cinza\_escuro = V4 + V1 - (V2 + V3) \quad (7)$$

$$Área\_cinza\_claro = V6 + V3 - (V4 + V5) \quad (8)$$

$$f(x) = Área\_cinza\_escuro - Área\_cinza\_claro \quad (9)$$

onde $f(x)$ é a característica.

O algoritmo de detecção de faces de Viola-Jones encontra-se implementado na biblioteca OpenCV (Open Source Computer Vision) (BRADSKI & KAEHLER, 2008) utilizada no desenvolvimento deste trabalho, como será apresentado a seguir. Esse algoritmo busca localizar em uma imagem características que codifiquem alguma informação do padrão sendo detectado.

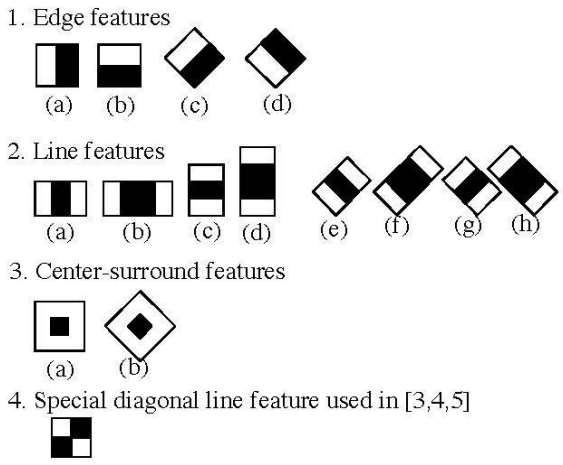

Figura 4 - Características do tipo Haar-Like. Fonte: LIENHART; MAYDT, 2002

Mesmo sendo relativamente fácil encontrar certas características do objeto de interesse em uma imagem integral, a quantidade de combinações que se pode ter das *HaarFeatures* é elevado. Desta forma, para que o processo de classificação seja rápido e eficiente, obtém-se um pequeno subconjunto composto das características, sendo que estas serão as mais significativas para representar o objeto de interesse, excluindo a maioria das características disponíveis.

A próxima etapa deste algoritmo (método *AdaBoost*) será então construir um classificador "forte" através da junção de vários sub-classificadores "fracos" que dependem apenas de uma característica.

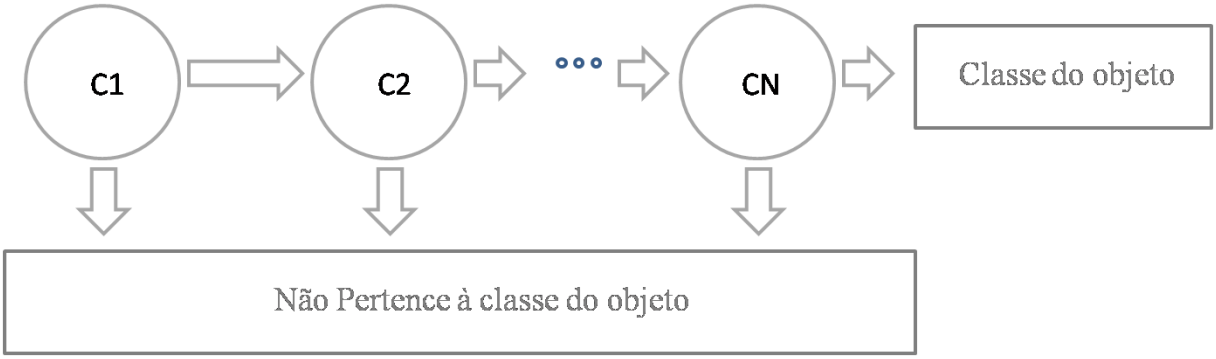

Figura 5 - Cascata de Classificadores. Fonte: DANTAS, 2013

Com o uso dos classificadores em cascata, reduz-se o tempo que o algoritmo necessita para realizar o processamento. Neste processo, o classificador (nó) seguinte só será invocado se a avaliação no anterior tiver sido realizada com sucesso. Se a detecção falhar em alguma etapa, todo o procedimento será interrompido. Desta forma, um determinado objeto só será detectado caso o resultado seja positivo em todas as etapas de classificação.

Analisando a Figura 5, percebe-se que existem vários classificadores, sendo que cada nó representa um conjunto de classificadores fortes, e estes, são compostos por diversos classificadores fracos. Assim, cada classificador fraco é definido pela função abaixo:

$$h(x, p, f, t) = \begin{cases} +1, & se\ f(x) < t \\ -1, & se\ f(x) \geq t \end{cases} \quad (10)$$

onde *h* é a função que representa o classificador fraco, *x* é uma sub-janela, *f* é um *feature*, e *t* é um limiar.

A seguinte função define um classificador forte:

$$C(x) = \begin{cases} +1, & se \sum_{t=1}^{T} \alpha_t h_t(x) \leq \frac{1}{2} \sum_{t=1}^{T} h_t(x) \\ -1, & caso\ contrário \end{cases} \quad (11)$$

onde $h_t(x)$ é o valor do t-ésimo classificador fraco, α_t é uma constante calculada durante o treinamento e T é o número de classificadores fracos.

O AdaBoost é quem realiza o treinamento dos classificadores fortes. Os classificadores fortes com mais características possuem uma taxa menor de falsos-positivos e consequentemente uma alta taxa de detecção. No entanto, necessitam de mais processamento.

Além de tornar mais rápido e eficiente a detecção de objetos, a classificação em cascata também pode melhorar a acurácia através de uma boa escolha de classificadores fortes pelo algoritmo AdaBoost.

A descrição do algoritmo AdaBoost (FREUND & SCHAPIRE, 1995) é muito extensa e por estar fora do escopo deste trabalho, não será abordado.

2.3 OpenCV

O OpenCV (BRADSKI & KAEHLER, 2008) é uma biblioteca de visão computacional, multiplataforma, que foi originalmente desenvolvida pela Intel. Possui licença livre e vários algoritmos de visão computacional já implementados. Possui uma extensa documentação e códigos de exemplo para aplicações de visão computacional.

Por ser uma biblioteca livre, ela é amplamente utilizada para o desenvolvimento de aplicações tanto no meio acadêmico quanto no meio comercial. Para isso é necessário apenas seguir o modelo de licença BSD da Intel.

Foi implementado nas linguagens de programação C e C++, com grande potencial para o desenvolvimento de aplicações. Funciona em várias plataformas como Linux, Windows e Mac OSX. Também oferece suporte a outras linguagens como, por exemplo, Java, Python e Visual Basic. Também pode ser utilizada no Matlab.

Possui um conjunto de módulos de processamento de imagens, estrutura de dados, álgebra linear, interface com o usuário (GUI), controle de hardware como teclado e mouse, entrada e saída de vídeo, entre outros. Dentre os algoritmos de visão computacional, estão também presentes algoritmos para calibração de câmera, detecção de objetos - como o método proposto por Viola-Jones, filtros de imagens, etc. Também apresenta mais de 500 funções que

englobam diferentes áreas, incluindo inspeção de produtos de fábrica, imagens médicas, segurança, interface de usuário, robóticas entre outros.

Com a utilização desta biblioteca é possível ganhar bastante produtividade na implementação de algoritmos de processamento de imagens, pois foi projetada para ter eficiência e um ótimo desempenho computacional, com foco em vários tipos de aplicações e, em especial, aquelas de tempo real como: detecção e reconhecimento facial, reconhecimento de gestos, entre outros.

O OpenCV também possui IPP (*Integrated Performance Primitives*) que são bibliotecas de Primitivas de Desempenho Integradas da Intel, isto é, algumas otimizações a mais para arquiteturas Intel. Essas otimizações são rotinas escritas em baixo nível visando o melhor aproveitamento dos recursos de hardware. As bibliotecas IPP são utilizadas em tempo de execução, se elas estiverem instaladas.

Está dividido em módulos, o que significa que inclui várias bibliotecas compartilhadas ou estáticas. Os módulos basicamente são:

- **core** – compacto e que define estruturas de dados básicas como, por exemplo, Mat e Point, e funções básicas utilizadas por todos os outros módulos.
- **imgproc** – inclui filtros de imagens lineares e não lineares, transformadas geométricas da imagem como, redimensionamento. Também calcula histogramas, conversão de cor, entre outros.
- **video** – para análise de vídeo que inclui estimação de movimento, subtração do fundo, e algoritmos de rastreamento de objetos.
- **highgui** – uma interface de fácil utilização para captura de vídeo, criação de janelas, entre outras funcionalidades.

Existem também outros módulos não citados neste trabalho, mas que podem ser visualizados e compreendidos através da documentação que pode ser obtida a partir do site oficial[1].

A vantagem é que fornece uma infra-estrutura de fácil utilização que ajuda os usuários a construírem várias aplicações de forma rápida e eficiente. O OpenCV também contém um módulo de aprendizado de máquina (*Machine Learning Library*).

[1] Site: http://opencv.org/

2.4 Plataforma de Prototipagem Eletrônica

Nesta seção, serão abordados alguns conceitos sobre a plataforma de prototipagem eletrônica, o Arduino, que será utilizado neste trabalho como circuito de controle para servomotores, sendo estes, utilizados para fazer o posicionamento da câmera em relação às faces.

2.4.1 Arduino

É uma plataforma de prototipagem que tem sido adquirida por várias pessoas nos últimos anos, principalmente por ser de fácil utilização e ser acessível, com um baixo custo. Esta plataforma possui o hardware livre, o que tem feito com que várias pessoas pudessem fabricar seus modelos de Arduino.

Com o seu microcontrolador Atmel AVR, torna-se relativamente fácil desenvolver projetos inteligentes e bastante profissionais, uma vez que detalhes eletrônicos da placa não fazem mais parte do interesse principal. E por ter sua origem em *Wiring*[2], o Arduino possui suporte à entrada e saída de dados embutida e uma linguagem de programação que é essencialmente C/C++.

O Atmel, assim como outros microcontroladores existentes, dispõe de uma arquitetura computacional completa em um único chip. Eles são compostos por: memórias ROM, memória RAM, temporizadores, circuito de clock embutido, entre outros componentes em sua arquitetura de hardware.

Através da utilização de linguagens de programação e uma IDE, é possível fazer com que o microcontrolador comande equipamentos eletrônicos ou até mesmo máquinas, e pelo fato de estar composto de um grande número de funcionalidades disponíveis, seu controle acaba sendo direcionado por um programa. Isso aumenta consideravelmente flexibilidade do projeto.

A variedade de microcontroladores existentes e a otimização dos sistemas embarcados, não apenas tornou possível o desenvolvimento de projetos inteligentes, como os de automação, como diminuiu bastante a complexidade, onde é possível substituir grande parte de uma arquitetura de hardware, por um único microcontrolador capaz de realizar a tarefa de forma equivalente ou até melhor. Isso foi possível principalmente devido ao baixo custo, a alta capacidade e a velocidade de processamento desses dispositivos.

[2]Nome dado à plataformas de prototipagem eletrônica baseado na utilização de uma linguagem de programação, uma interface de desenvolvimento integrado (IDE) e um microcontrolador.

Trabalhar com microcontroladores como, por exemplo, da família Atmel, entre outros, tornou-se fundamental para desenvolvimento de projetos eletrônicos de alto desempenho, visto que as possibilidades de desenvolver circuitos inteligentes tornou-se uma tarefa menos complexa.

Por ser uma plataforma de prototipagem, o circuito do Arduino dispõe principalmente de um microcontrolador, pinos de alimentação, pinos de entrada e saída de dados digitais, entrada de sinal analógico, pinos para transmissão (Tx) e recepção (Rx) de dados na comunicação serial, leds para sinalização, entre outros, como demonstrado na Figura 6.

Vale também ressaltar que existem diferentes modelos, sendo que cada um, possui suas particularidades, que vai desde uma capacidade maior de processamento até diferentes tipos de interface de comunicação, como USB, Ethernet e outros. Dentre os vários tipos de Arduino, existem: o UNO, o Mega, o Leonardo, o Freeduino, o Severino, o LilyPad, e vários outros.

O Arduino tem sido também utilizado para o meio educacional e principalmente por desenvolvedores ou mesmo por pessoas sem formação técnica que desejam criar seus projetos, sem precisar necessariamente ter conhecimento de detalhes técnicos. Além disso, a sua IDE possui vários códigos de exemplo que facilitam a adaptação e familiarização com a placa do Arduino.

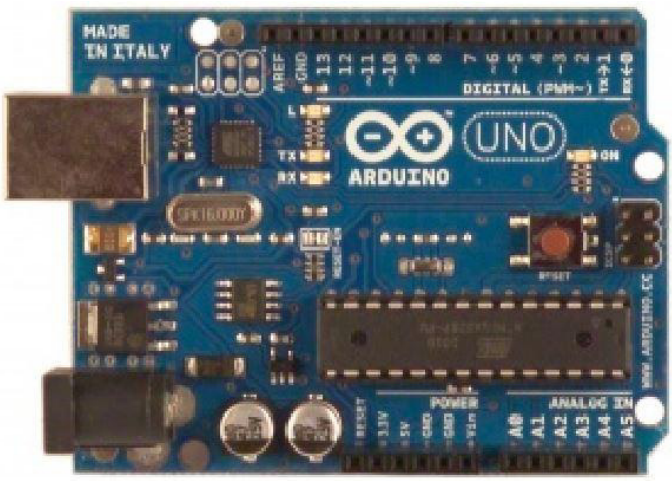

Figura 6 - Arduino UNO. Fonte: Site Oficial do Arduino[3]

A linguagem de programação utilizada é essencialmente C e C++, como mensionado anteriormente. Em sua configuração inicial, o programa possui basicamente duas funções principais, que são: "*void setup()*" e "*void loop()*". A primeira função é logo chamada quando

[3] Disponível em: <http://www.arduino.cc> Acesso em out. 2013.

o programa é transferido para a placa e é executado. Em seguida, a segunda função é chamada e fica em um loop infinito para que o programa não termine a sua execução. Assim, toda rotina que estiver dentro dessa função é constantemente executada até que uma situação force a parada, como por exemplo, o corte da corrente elétrica do circuito. Além dessas duas funções, existem várias outras que permitem ao Arduino escrever ou ler dados ou sinais em seus pinos digitais e analógicos, funções para controle de servomotores, funções para comunicação serial e várias outras.

Também existem módulos chamados de *Shields*[4], que possibilitam o controle de dispositivos diferentes, aquisição de dados/sinais, entre outros. Dentre os *Shields* disponíveis, existem aqueles que permitem a transmissão de dados sem a utilização de fios (*XBee Shield*), os que possibilitam a conexão com a internet (*Ethernet Shield*), os que o permitem controlar motores de corrente contínua (*Motor Shield*), entre outros. Além dos *Shields* existentes, outros modelos podem ser desenvolvidos por pessoas que possuem um pouco mais de conhecimentos técnicos, fazendo-os funcionar de acordo com as necessidades do projeto.

2.4.2 Comunicação Serial

Para entender como funciona a comunicação serial entre dispositivos e um Arduino, é interessante discutir, genericamente, o conceito e o funcionamento da comunicação serial.

A comunicação serial se trata de uma entre várias formas existentes de transmissão de dados entre dispositivos, ou seja, trata-se de um processo de transmissão de dados, bit por bit, sequencialmente em um canal de comunicação ou em um barramento. Apesar de, em alguns casos, não ser tão rápida quando outros tipos de transmissão de dados como, por exemplo, a comunicação paralela, apresenta a vantagem de ser mais simples e eficiente, além de poder ser utilizada na transmissão de dados em longa distância.

Existem diversas tecnologias de comunicação serial como RS-232, USB, FireWire, SCSI, SPI, I2C, dentre outros. Nelas, o tipo de comunicação pode ser síncrona ou assíncrona. Esta última é a mais simples de ser implementada e utilizada. Nela não existe um hardware específico para o sincronismo entre transmissor e receptor e, por isso, a sincronização é feita caractere por caractere.

Algumas características sobre a comunicação serial devem ser abordadas, como: Taxa de Transmissão (*Baud Rate*), Bits de Dados, Bits de Parada e Bit Paridade.

[4] São placas de circuito impresso adaptadas ao Arduino, disponibilizando assim novos recursos e expandindo suas funcionalidades.

A taxa de transmissão nada mais é do que a quantidade de bits que podem ser transmitidas por segundo, ou seja, é uma medida de velocidade de comunicação. Essa taxa está diretamente relacionada ao ciclo de clock, isto é, uma taxa de 9600 indica que o clock está gerando o sinal a uma frequência de 9600 Hz. Assim, os dados serão transmitidos a uma velocidade de 9600 bits por segundo. Quanto maior for esta taxa, menos distantes, um do outro, devem estar os dispositivos para que não ocorra erro na transmissão ou perda de dados.

Os bits de dados são os bits que representam a informação a ser transmitida. Normalmente os bits de dados são transmitidos em um pacote, que é composto também pelos bits de inicio e fim da transmissão, e o bit paridade (que pode ou não estar presente) como pode ser visto na Figura 7. Para a transmissão assíncrona, o bit de inicialização serve para que preparar o mecanismo de recepção de dados. Depois que o dado é transmitido, e logo em seguida, o bit de finalização, a linha pode ficar ociosa, ou outro bit de inicialização pode ser enviado para uma nova transmissão. Como o número atual de bits de dados depende do protocolo a ser utilizado, o termo pacote é utilizado para cobrir todas as instâncias.

O bit de parada, como dito anteriormente, serve para indicar o fim da comunicação com um único pacote. Visto que cada dispositivo possui seu próprio clock, normalmente ambos dispositivos podem estar fora de sincronia. Assim, os dados são cronometrados através da linha. Além de indicar o fim da transmissão, o bit de parada serve também para indicar uma margem de erro nas velocidades de clock dos dispositivos.

O bit paridade é aquele utilizado para verificar possíveis erros na transmissão de dados. Este bit é enviado logo após os bits de dados. Dentre os tipos de bit paridade, é comum encontrar a paridade par e a paridade ímpar. Para esses dois tipos, a porta serial define o tipo de paridade a ser utilizado. Se, por exemplo, a paridade definida for par, o somatório de todos os bits de dados mais o bit paridade, deve dar um valor par. Por exemplo, se o dado transmitido for 0111011, então para paridade par, o bit de paridade será 1. No caso de paridade ímpar, o bit paridade seria 0. Assim, a quantidade de bits 1 permaneceria ímpar. Desta forma, o dispositivo receptor pode checar se houve algum erro na transmissão.

Figura 7 - Comunicação serial assíncrona. Fonte: Wikipedia[5]

Tabela 1 - Taxas de Comunicação mais comuns. Fonte: Serial[6].

Taxa (*Baud Rate*)	Tempo
110	9.1 ms
150	6.66 ms
300	3.33 ms
600	1.66 ms
1200	833 us
2400	416 us
4800	208 us
9600	104 us
19200	52 us

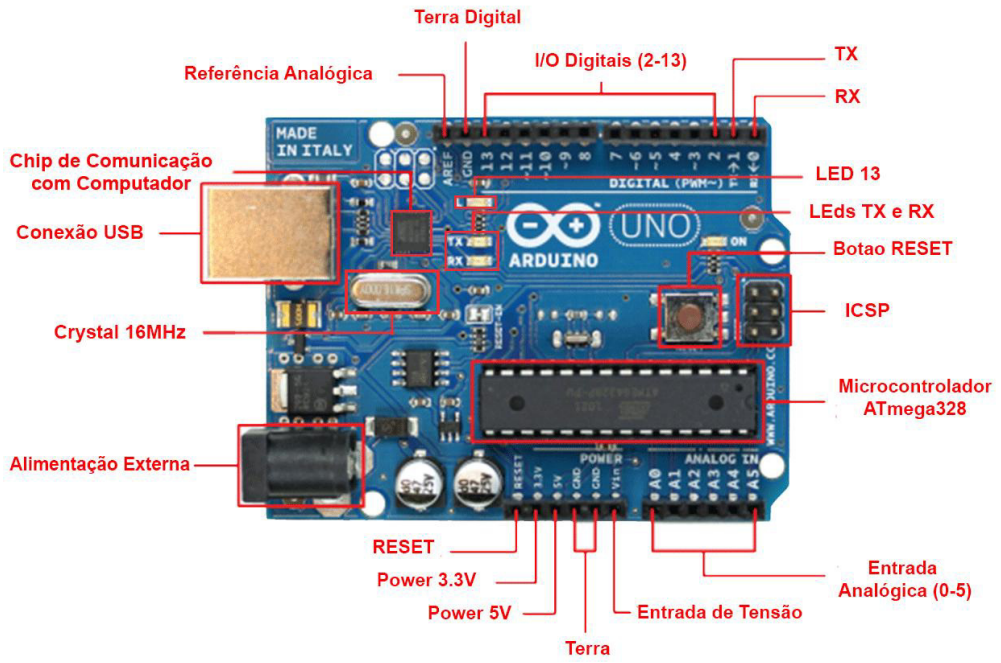

Figura 8 - Arduino UNO. Fonte: Componentes de um Arduino[7]

[5] Disponível em: <http://pt.wikipedia.org> Acesso em nov. 2013.
[6] Disponível em: <http://iris.sel.eesc.usp.br/sel337/serial.pdf> Acesso em nov. 2013.
[7] Disponível em: <http://www.smartleaf.com.br/componentes-de-um-arduino/> Acesso em nov. 2013

2.4.3 Comunicação Serial - Arduino

A plataforma de prototipagem, o Arduino, também possui interfaces para que a comunicação serial possa ser realizada. A placa utilizada neste trabalho, o Arduino UNO, possui interfaces para comunicação serial como pode ser visto na Figura 8, onde existe a interface de Conexão USB, os pinos de entrada/saída digitais 0 e 1 correspondendo respectivamente ao Rx (receptor) e Tx (transmissor), e também, comunicação SPI através dos pinos 10(SS), 11(MOSI), 12(MISO) e 13(SCK) pela utilização da biblioteca SPI.

Os pinos Tx e Rx são utilizados quando há algum circuito auxiliar com o qual se deseja realizar a conexão como, por exemplo, outro circuito microcontrolado, outro Arduino, módulos Bluetooth, entre outros. O Tx é utilizado para transmitir os dados a um circuito externo. Para que o outro circuito possa receber corretamente esses dados, este Tx deve está conectado ao Rx do outro circuito. Da mesma forma, o Tx do outro deve ser conectado ao Rx do Arduino, para que o mesmo possa receber os dados corretamente, como pode ser visto na Figura 9.

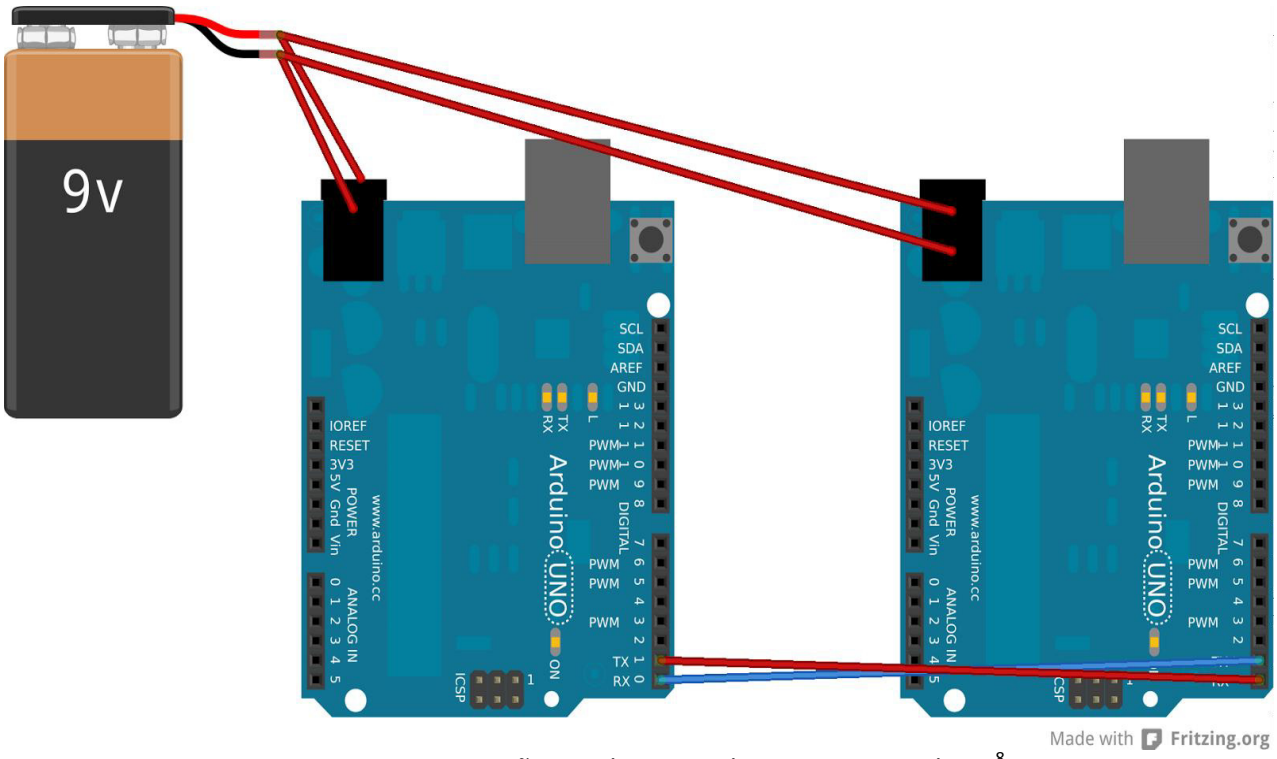

Figura 9 - Comunicação serial entre Arduinos. Fonte: Arduino[8]

[8] Disponível em: <http://arduinizando.blogspot.com.br/2013/02/comunicacao-serial-entre-arduinos.html> Acesso em nov. 2013.

A interface USB é normalmente utilizada quando se deseja realizar a comunicação serial entre o computador e o Arduino, através da qual o programa desenvolvido pode ser transferido para a placa. Além disso, uma vez que o programa está gravado no microcontrolador Atmel AVR, é possível utilizar a conexão USB para realizar testes de transferência de dados do computador para o Arduino e vice-versa. Os resultados coletados podem ser exibidos no computador. Esta interface, além de permitir a comunicação serial, também permite que o circuito seja alimentado, com a voltagem fornecida pela porta USB do computador.

A comunicação serial do Arduino é controlada por um único chip. Desta forma, a comunicação feita utilizando tanto a porta USB quanto os pinos digitais Tx e Rx são compartilhadas. Assim, não é possível utilizá-las ao mesmo tempo em conexões diferentes. Por exemplo, tendo o Arduino conectado ao PC pela porta USB, e tendo-se um circuito auxiliar conectado aos pinos Tx e Rx. Ao tentar transferir um novo programa para Arduino, haverá um erro na gravação, uma vez que ambas as conexões seriais estão sendo utilizadas, e assim, o controlador não "saberá" a quem priorizar.

Para que a utilização da comunicação serial seja possível, a própria IDE do Arduino dispõe de bibliotecas que implementam esse tipo de comunicação. Assim, para utilizar a comunicação serial, existem várias funções para isso. As principais são: *Serial.begin(), Serial.available(), Serial.readBytes()* e *Serial.write()*.

A "*Serial.begin()*" serve para definir, no início do programa, a velocidade de comunicação da porta serial. A "*Serial.available()*" realiza o monitoramento do canal de comunicação, isto é, verifica se há dados a serem tratados. Ela retorna zero, caso não haja nenhum dado, e um valor maior que zero, caso haja alguma informação. As funções "*Serial.readBytes()*" e "*Serial.write()*" servem, respectivamente, para leitura e escrita de dados.

2.4.4 ServoMotores

Servomotores, ou apenas servos, são dispositivos semelhantes aos motores de corrente contínua convencionais. No entanto, possuem a capacidade de gerar uma rotação, no eixo, proporcional ao comando aplicado ao circuito de controle do motor, isto é, é possível gerar rotação na quantidade de graus desejados, diferentemente dos motores comuns que giram seu eixo livremente sem um controle de precisão. A Figura 10 apresenta um exemplo de servomotor e a sua estrutura interna.

Este dispositivo é formado basicamente por um motor de corrente contínua, uma caixa redutora, um circuito de controle e um potenciômetro. Esse conjunto permite que o servomotor trabalhe em um range de 180°, isto é, de -90° à 90°.

Esse tipo de motor é muito utilizado em robótica e também em aeromodelos, helimodelos, entre outros. No entanto, a sua aplicação não se restringe somente a estes, podendo se extender de acordo com a necessidade e criatividade do projetista, como por exemplo, mecanismo Pan-Tilt[9] para câmeras de segurança.

Para que todo o mecanismo funcione, existem três fios que são conectados ao circuito de controle, sendo que um é para o positivo (vermelho) da fonte de alimentação, o outro é a referência para o terra (preto ou marrom) e o último é para o sinal de controle (branco ou amarelo).

A tensão de alimentação normalmente é de 5V, e mesmo a uma tensão relativamente baixa, é possível ganhar um torque considerável de acordo com a caixa de redução (conjunto de engrenagens), que possui a função tanto de reduzir a velocidade de rotação do eixo do motor, quanto atribuir força ao acionamento. O torque está diretamente relacionado ao conjunto de engrenagens que o mesmo possui e o seu tamanho como um todo. Com isso, é possível acionar mecanismos.

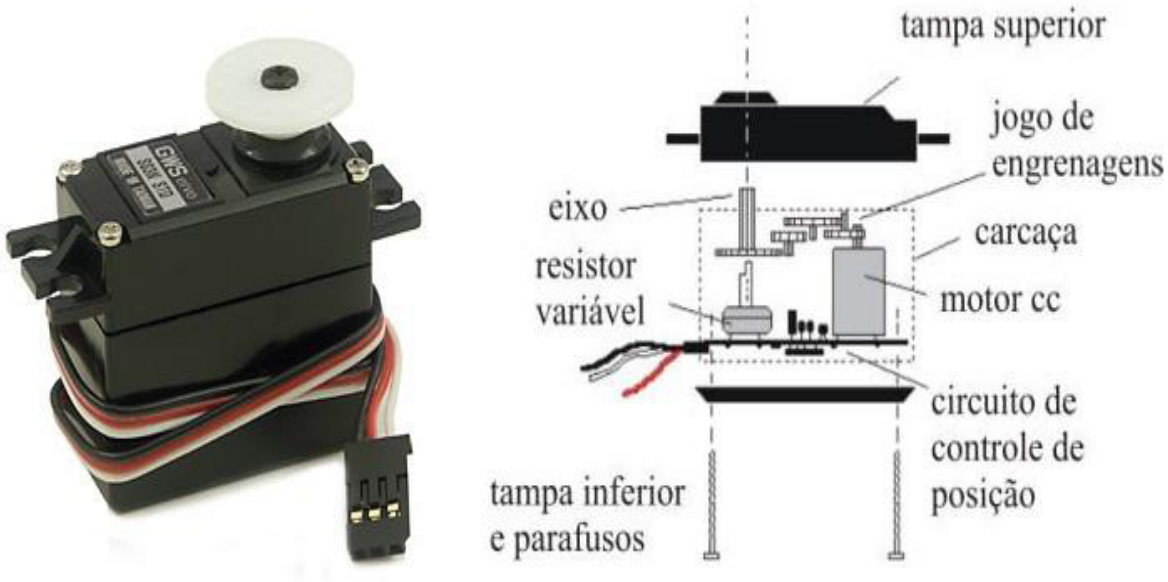

Figura 10 - Servomotor. Fonte: Posição e Rotação dos servomotores[10]

[9] Mecanismo que utiliza motores para fazer o posicionamento de câmeras de segurança na direção vertical e horizontal.

[10] Disponível em: <http://moisesricardo.xpg.uol.com.br/> Acesso em nov. 2013.

2.4.5 Funcionamento de ServoMotores

Como já mencionado, este dispositivo é composto basicamente de um motor de corrente contínua, um potenciômetro e um conjunto de engrenagens (redução). Há também três fios, sendo um positivo, o outro negativo e o terceiro é o de sinal. Diferentemente dos motores convencionais de corrente contínua, o servomotor não funciona simplesmente alimentando-o com o positivo e negativo da fonte. Para que o mesmo funcione, é preciso fornecer além desses, um sinal que será transmitido ao circuito de controle. Este será responsável por controlar a rotação do motor.

O circuito de controle, ao receber o sinal, determina o ângulo de rotação do eixo baseado na posição atual. No entanto, o controlador só "sabe" a posição atual do eixo graças à resistência marcada pelo potenciômetro que está conectado ao mesmo e, de acordo com esta resistência, é possível determinar a posição angular do eixo e calcular a nova posição para a qual o eixo deverá rotacionar.

Normalmente, um servomotor consegue fazer o deslocamento em um range de 180º, mas dependendo do fabricante, esse ângulo pode variar até 210°.

O tipo de sinal utilizado para comandar este dispositivo normalmente é o PWM (*Pulse Width Modulation – Modulação por Largura de Pulso*). Este é um tipo de sinal digital e, para controlar os servomotores, usualmente precisa de um período de 20ms de sinal. Sendo que neste intervalo, o nível lógico alto, também chamado de ciclo ativo (duty cicle), pode variar de 1ms a 2ms em alguns modelos e em outros, varia de 0.5ms à 2.5ms. No restante do tempo, o sinal fica em nível lógico baixo. Um exemplo pode ser visto na Figura 11.

Para saber ao certo o tempo que o nível lógico deve permanecer alto ou baixo no intervalo de 20ms, deve-se consultar o datasheet do dispositivo que é fornecido pelo fabricante.

Como mencionado previamente, normalmente o sinal de controle consiste de um sinal PWM com um nível lógico alto de 1ms a 2ms e o restante do tempo em nível lógico baixo. O período entre um sinal e outro deve ser utilizado de acordo com as especificações do servo, pois se o sinal for repetido rapidamente (Ex: 8ms à 15ms), será possível perceber certa instabilidade neste motor através de um zumbido gerado pelo mesmo. Se o intervalo entre um sinal e outro for muito elevado (Ex: 60ms) o servomotor não funcionará.

Ao receber o sinal PWM, que consiste em um trem de pulsos com amplitude constante de 5V, o controlador detecta a resistência atual marcada pelo potenciômetro e então aciona o motor de corrente contínua que irá rotacionar o conjunto de engrenagens da caixa de redução

e, estas por sua vez, irão rotacionar o eixo principal do dispositivo, como pode ser visto na Figura 11.

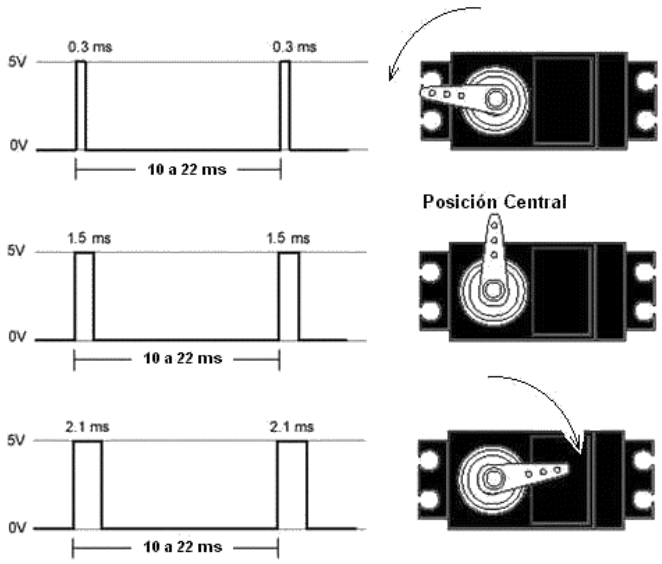

Figura 11 - Sinal PWM. Fonte: Servos[11]

O circuito de controle está composto basicamente de vários componentes discretos como resistores, capacitores, transistores e, além destes, circuitos integrados. Além disso, possui também um oscilador e um controlador PID[12] (*Proportional Integral Derivative* - *Proporcional Integral Derivativo*). A Figura 12 apresenta um servomotor desmontado para que seja possível ver a estrutura interna do mesmo.

Figura 12 - Servomotor desmontado. Fonte: Servos[13]

[11] Disponível em: <http://www.roboticapy.com/servo.asp> Acesso em nov. 2013.
[12] É uma técnica muito utilizada em controle de processos, cujo objetivo é minimizar o erro através da ação proporcional, integral e derivativa. A ação proporcional minimiza o sinal de erro, a integral zera e a derivativa antecipa a velocidade de obtenção do sinal no processo.
[13] Disponível em: <http://www.roboticapy.com/servo.asp> Acesso em nov. 2013.

3 METODOLOGIA

Este capítulo apresenta a metodologia proposta para realização do rastreamento de faces a partir do vídeo de uma câmera e então envio de sinais de controle para o circuito responsável pelo posicionamento correto da câmera, visando manter a face centralizada no vídeo. Os passos foram executados de acordo com a Figura 13.

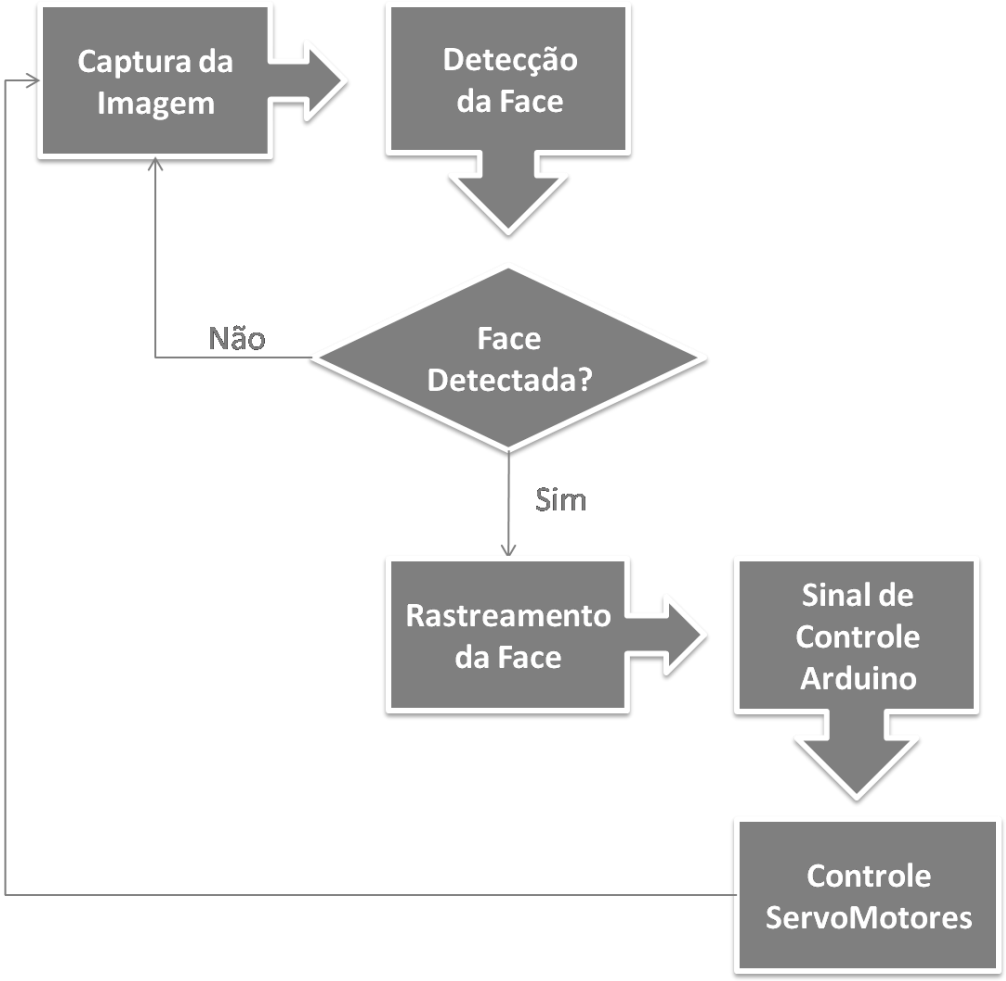

Figura 13 - Fluxograma da Metodologia. Fonte: DANTAS, 2013

A seguir, será feita uma breve explicação a respeito de cada etapa ilustrada no fluxograma da Figura 13 para um entendimento geral da metodologia.

a) Captura da Imagem – Esta é a etapa inicial, através da qual se realiza a captura de cada frame do vídeo. De posse da imagem, os processamentos subsequentes são realizados para que a detecção da face possa ser completada. A resolução do vídeo, e conseqüentemente de cada frame, é de 640x480.

b) Detecção da Face – Nesta etapa, utiliza-se o método de Viola-Jones para detecção de faces através da utilização de classificadores em cascatas e características *Haar*.

c) Rastreamento da Face – A partir da face detectada pela etapa anterior, realiza-se um cálculo necessário para obter as coordenadas da face em relação ao centro da imagem. De posse dessas coordenadas, um sinal de controle é enviado ao Arduino através da comunicação serial.

d) Sinal de Controle Arduino – Ao receber o sinal de controle do algoritmo de rastreamento facial, o Arduino converte esse sinal em comandos de posicionamento (movimento) e os envia a cada um dos servomotores.

e) Controle ServoMotores – Os motores ao receber o sinal de comando do Arduino, irão realizar os movimentos *Pan* (em relação ao eixo x) e *Tilt* (em relação ao eixo y). O movimento *Pan-Tilt* posiciona a câmera para que o rosto da pessoa fique sempre centralizado no vídeo.

A Figura 14 mostra a organização dos servomotores que serão responsáveis por realizar os movimentos *Pan-Tilt*. Depois de pronta a configuração, fez-se todas as conexões elétricas, como a alimentação e ligação dos servos aos pinos de saída digital da plataforma de prototipagem.

Apesar de o Arduino estar utilizando a conexão USB com um computador para recebimento do sinal de controle, e também para alimentação com os 5V da porta USB, faz-se necessário a utilização de uma fonte externa. Isso porque a corrente fornecida pela porta USB não é suficiente para manter um bom funcionamento dos servomotores. Com isso, utiliza-se uma fonte de 12V ligada a um regulador de tensão que gera os 5V necessários para alimentação dos motores. Com esta configuração, pode-se manter uma boa estabilidade no funcionamento dos servos.

Como o mecanismo Pan-Tilt está composto por dois servomotores, é necessário que cada um esteja conectado ao Arduino em pinos separados, para que cada um tenha o movimento independente do outro. Assim, conectam-se os servos aos pinos digitais de saída PWM da placa.

Figura 14 - Mecanismo Pan-Tilt. Fonte: DANTAS, 2013

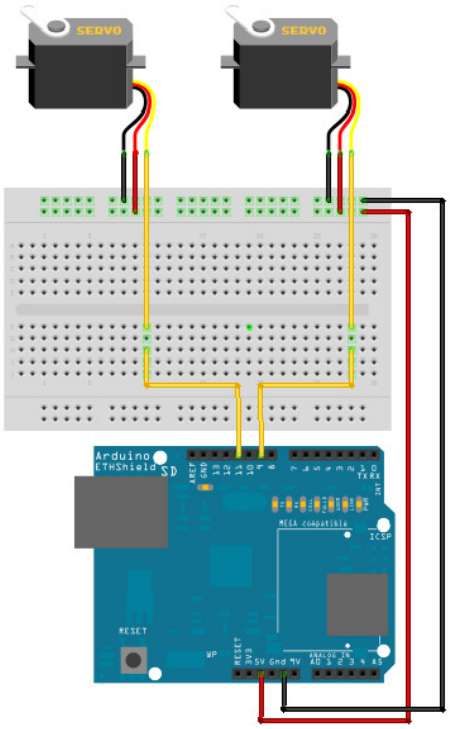

Figura 15 - ServoMotores com Arduino UNO. Fonte: Controle de dois servomotores[14]

[14] Disponível em: <http://fisicarduino.wordpress.com/2011/05/29/controle-de-dois-servo-motores-usando-python-e-arduino> Acesso em nov. 2013

Os detalhes do circuito montado, através da utilização do Arduino com o mecanismo Pan-Tilt podem ser visto na Figura 16.

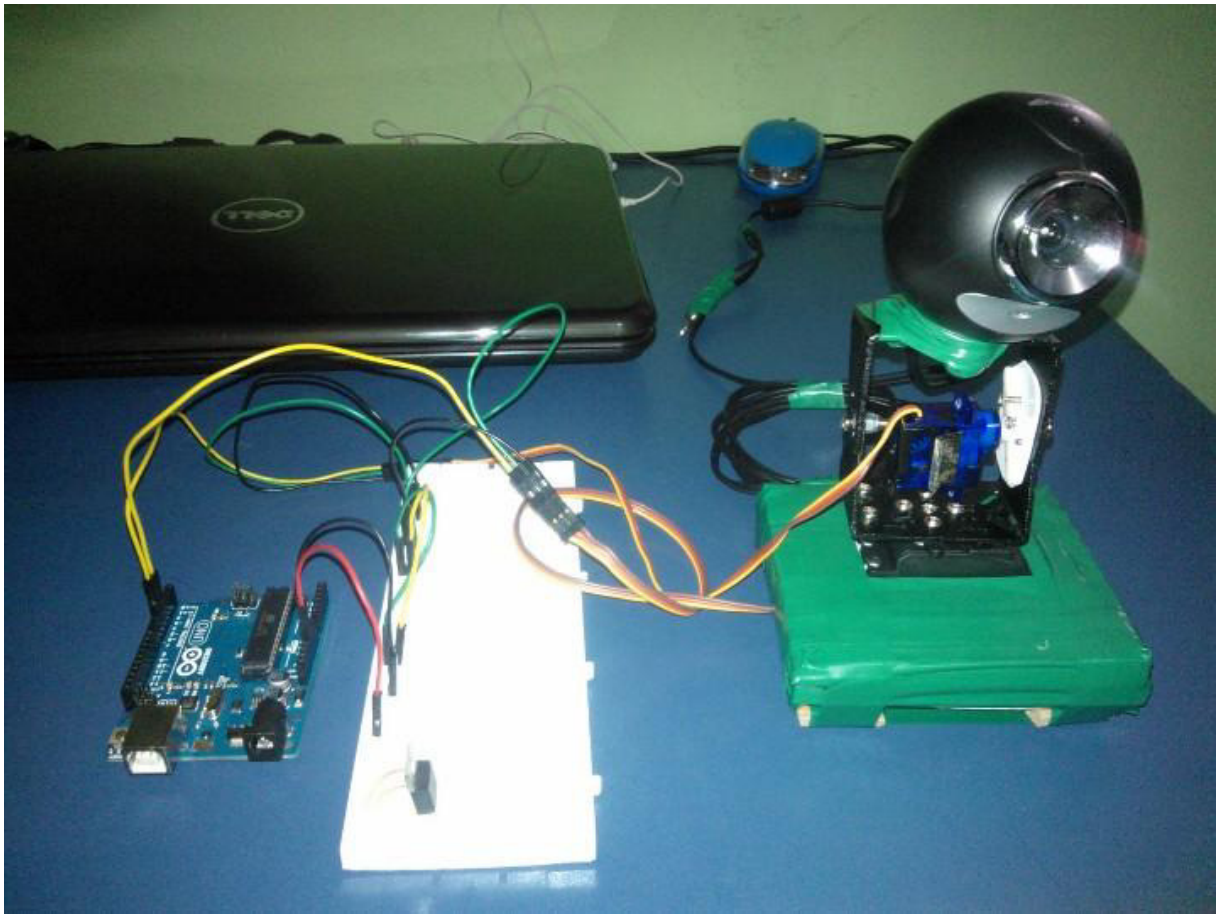

Figura 16 - Circuito com Arduino e Mecanismo Pan-Tilt. Fonte: DANTAS, 2013

Com a implementação de hardware pronta, foi necessário fazer algumas configurações de software na Plataforma Linux, na qual estaria executando o algoritmo de rastreamento implementado na linguagem Python.

Primeiramente, foi necessário realizar a instalação da IDE do Arduino, através da qual foi possível implementar o algoritmo que posteriormente seria gravado no microcontrolador Atmel AVR. Essa IDE, também possui mecanismos que possibilitam a realização de testes com a comunicação serial, para verificar se o recebimento de dados estava sendo tratado corretamente.

Posteriormente, foi necessário configurar a biblioteca OpenCV para a linguagem Python. A instalação e configuração foram realizadas através do comando: **sudo apt-get install python-opencv.**

Estando então com a biblioteca configurada, foi utilizado e adaptado o algoritmo de detecção de faces do OpenCV que utiliza o método de Viola-Jones. A adaptação foi realizada de forma que fosse possível realizar o rastreamento facial e consequentemente envio de dados ao circuito com o Arduino. O rastreamento consiste em ficar detectando consecutivamente a posição da face em relação ao frame recebido e converter essa posição em sinais de controle que posteriormente serão enviados ao circuito eletrônico, que se encarregará de controlar os motores corretamente a fim de posicionar a câmera. Isso permitirá que o rosto da pessoa fique sempre centralizado na imagem.

O algoritmo de detecção de faces ao receber o vídeo da câmera, isto é, cada *frame* (Quadro), converte a imagem de colorida para nível de cinza, ou seja, converte a imagem de 3 para 1 camada, visando reduzir o processamento necessário para detecção de faces. Posteriormente, é realizado um redimensionamento, diminuindo assim o seu tamanho. Por fim, é realizada uma equalização do histograma da imagem a fim de melhorar a qualidade dos *pixels*, o que pode melhorar substancialmente o resultado do processamento.

De posse da nova imagem com a resolução de aproximadamente 533x400, faz-se a detecção da face. Cada face detectada é marcada com um retângulo exibido na tela. A partir daí, é calculado a posição atual da face, e a posição onde a mesma deveria estar. Com base nisso, os dados são enviados ao Arduino para que o mesmo possa corrigir esse posicionamento. Ambas as imagens podem ser vistas nas Figuras 17 e 18.

Os dados enviados pela comunicação serial, para o posicionamento correto da câmera, não são enviado de uma única vez para o circuito microcontrolado para não gerar instabilidade do posicionamento dos motores. A solução adotada foi o envio dos dados de forma compassada, isto é, a cada iteração do programa, reduz-se a distância entre o centro da face e o centro da imagem e então, envia-se o novo dado ao Arduino.

A implementação do software foi feita na linguagem Python por apresentar e fornecer uma boa interface de comunicação serial (transmissão e recepção de dados) com o Arduino. Isto porque anteriormente haviam sido realizados testes na linguagem C, no entanto, eram gerados muitos erros na transmissão dos dados.

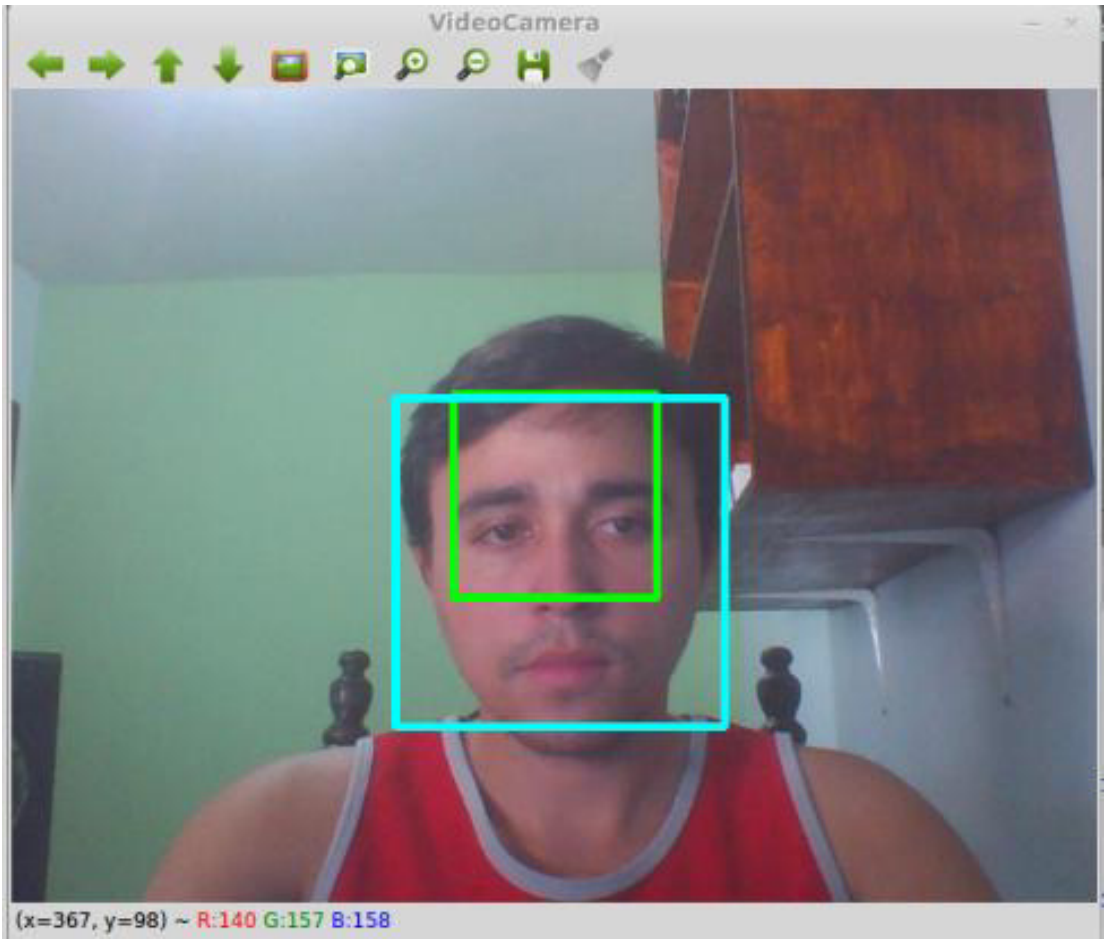

Figura 17 - Detecção de Face - Imagem Normal. Fonte: DANTAS, 2013

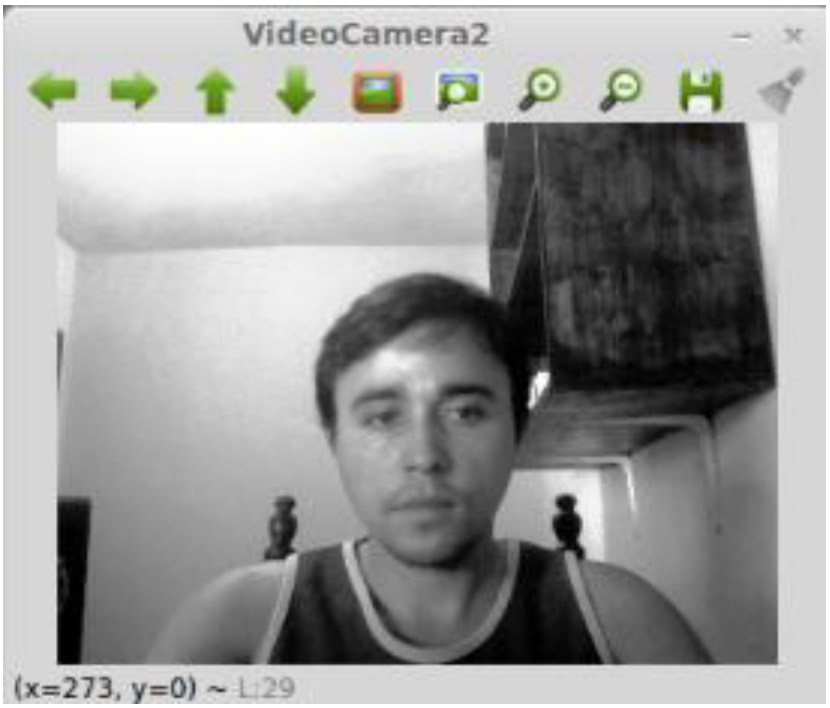

Figura 18 - Imagem redimensionada e com histograma equalizado. Fonte: DANTAS, 2013

A porta para comunicação serial utilizada para comunicar o software em Python com o Arduino foi a /dev/ttyACM0, e a taxa de comunicação (*Baud Rate*) utilizada foi de 57600 bps.

O software responsável pelo controle dos servomotores, foi criado através da utilização da biblioteca disponibilizada pela IDE do Arduino. A biblioteca utilizada foi a "Servo.h" que dispõe de funções que comandam os servos conectados aos pinos PWM do Arduino. A função "*Servo.write(angulo)*" serve para rotacionar o eixo do motor para o ângulo passado como parâmetro, que vai normalmente de 0 a 180 graus.

Apesar de existirem outras formas de implementar um sinal de saída no formato PWM, como ilustrado na Figura 19 (com delays de 1ms e 19ms), optou-se pela utilização a biblioteca por fornecer uma forma simplificada e confiável de controle de servomotores.

```
digitalWrite(outPin,HIGH);
delay(1);
digitalWrite(outPin,LOW);
delay(19);
```

Figura 19 - Trecho de código para gerar sinal PWM simples. Fonte: DANTAS, 2013

Ao energizar o Arduino, o software gravado no mesmo se encarrega de fazer a configuração inicial através da função "Setup()" como, por exemplo, a taxa de transmissão (*Baud Rate*) para 57600 bps, saídas digitais para controle dos servos, posição inicial para os motores, cujo ângulo escolhido foi o de 90º. Logo em seguida, o software entra na função "Loop()" na qual está a implementação principal de recepção de dados e controle dos motores.

3.1 Mapeamento de dados

Esta etapa de mapeamento consiste em converter os dados que são obtidos pelo algoritmo de detecção e rastreamento facial e enviá-los de forma padronizada para hardware responsável por controlar os motores, de forma que seja possível interpretá-los e convertê-los em movimentos.

Como dito anteriormente, os dados são enviados de forma compassada para não gerar instabilidade no vídeo capturado pela câmera. Assim, a cada iteração do algoritmo de rastreamento facial, são calculadas as distâncias entre o centro da face e o centro da imagem, nos eixos X e Y. Esses valores são então formatados para posteriormente serem transmitidos.

Para envio dos dados, criou-se um protocolo de comunicação para garantir a transmissão dos dados. Este protocolo consiste de um caractere especial indicando o início da transmissão, o dado referente ao posicionamento no eixo x, outro caractere especial, o dado referente ao posicionamento no eixo y, e um caractere de fim de comunicação. Assim, essa informação é transmitida em uma sequência de bytes, de acordo como o formato abaixo:

#PosicaoX.PosicaoY.

O Arduino, a cada iteração, verifica se existem informações na porta serial que devem ser tratadas. Se houver informação disponível, é verificado se o primeiro byte corresponde ao caractere de início de comunicação. A partir disso, a informação contida nessa sequência de bytes é seguimentada e convertida em valores referentes ao posicionamento angular de cada um dos servos. Posteriormente, o sinal PWM é enviado a cada um dos motores e a movimentação é realizada.

O fluxograma representado na Figura 20 demonstra as etapas executadas pelo Arduino, desde a inicialização até a recepção dos dados.

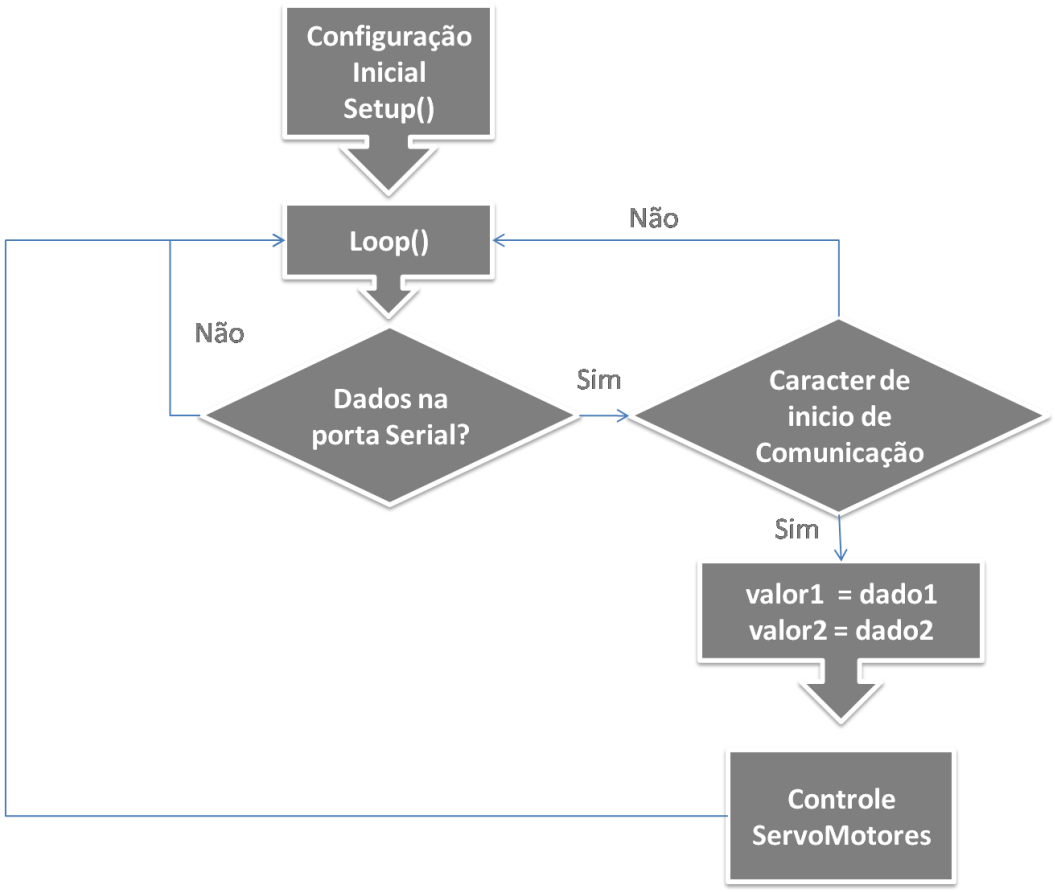

Figura 20 - Fluxograma - Passo a Passo Arduino. Fonte: DANTAS, 2013

4 CONCLUSÃO

Este trabalho apresentou o desenvolvimento de uma solução de software e hardware, cujo objetivo principal é fazer o rastreamento facial em tempo real. Para isso, foi implementado um software capaz de realizar a detecção de faces em vídeos e, a partir destes, rastrear da face em cada quadro e, com isso, gerar um sinal de controle para ser enviado a um circuito eletrônico capaz de controlar servomotores automaticamente em um movimento conhecido como Pan-Tilt.

O passo inicial consistiu em adaptar o software de detecção facial pelo método de Viola-Jones, através da biblioteca OpenCV, para além de detectar faces, poder enviar dados através da porta serial para o Arduino. Logo em seguida, foi necessário criar a estrutura que suportaria os dois servomotores responsáveis pelo posicionamento automático da câmera de vídeo.

Uma etapa, desenvolvida em paralelo, foi a criação de um algoritmo de controle que ficou armazenado na memória interna do Arduino. Este algoritmo foi implementado de forma que fosse capaz de receber dados pela porta serial, verificar o tipo de informação recebida e consequentemente, caso fosse uma informação válida, converter os dados recebidos em sinais de comando para os servomotores.

Durante a fase de implementação do algoritmo de detecção e rastreamento facial, foram feitos vários testes de forma que fosse verificada a eficiência de movimentação dos motores. Uma característica importante a se observar, é que durante os testes, foi possível perceber que a velocidade de movimento dos motores deveria ser baixa, isto porque era necessário manter uma uniformidade no movimento para se ter qualidade na aquisição do vídeo, o que não seria possível com a utilização de altas velocidades de movimentação. Foi possível perceber que movimentos bruscos faziam a câmera desestabilizar os quadros capturados.

A implementação de hardware foi realizada de forma prática, através da utilização do Arduino que é uma plataforma de prototipagem já montada, pronta para ser utilizada e que também já dispõe de uma IDE com um conjunto de bibliotecas muito úteis para as implementações mais sofisticadas.

Uma desvantagem que o algoritmo de rastreamento facial apresentou foi em situações em que a câmera filmava o rosto em locais com iluminação inadequada, tanto com uma iluminação excessiva, quanto com pouca iluminação. Isso fez com que o algoritmo apresentasse dificuldades na detecção facial. Uma solução adotada, foi realizar os testes em

locais com iluminação uniforme, garantindo assim uma melhor qualidade no vídeo e consequentemente detecção e rastreamento da face.

De maneira geral, a metodologia apresentada se mostrou bastante funcional de forma que trabalhos futuros envolvendo aplicações de rastreamento de objetos como, o rastreamento facial, possam ser aplicados, uma vez que é possível obter imagens desses objetos em tempo real. Utilizando a metodologia apresentada, poder-se-á desenvolver, além desta, aplicações como reconhecimento facial, detecção de sonolência em motoristas e outros, cujo objeto de interesse tenha que ser rastreado.

REFERÊNCIAS

ALLEN, G.; XU, R.; JIN, J. Object Tracking Using CamShift Algorithm and Multiple Quantized Feature Spaces. In: NSW 2006.

BRADSKI, G., & KAEHLER, A. (2008). Learning OpenCV. Sebastopol, CA: O'Reilly Media.

FREUND, Y.; SCHAPIRE, R. E. A decision-theoretic generalization of on-line learning and an application to boosting. In: European Conference on Computational Learning Theory. [s.l.: s.n.], 1995. p. 23–37.

GONZALEZ, R. C., & WOODS, R. C. (2010). Processamento Digital de Imagens (3ª Edição ed.). São Paulo: Pearson Prentice Hall.

GONZALEZ, R. C.; WOODS, R. C.. Digital Image Processing. Prentice Hall. Ed. 3. 2010.

LIENHART, R., & MAYDT, J. (2002). An Extended Set of Haar-like Features for Rapid Object Detection. IEEE ICIP, 1, pp. 900-903.

OpenCVWiki: cvBlobsLib. (s.d.). Acesso em 20 de Setembro de 2013, disponível em http://opencv.willowgarage.com/wiki/cvBlobsLib/

VIOLA, P., & JONES, M. (2004). Robust Real-Time Face Detection. International Journal of Computer Vision, 57, pp. 137-154.

SAWICZ, D. - Hobby Servo Fundamentals. Acesso em 12 de Novembro de 2013, disponível em: http://www.princeton.edu/~mae412/TEXT/NTRAK2002/292-302.pdf

SHIRRIFF, K. – Secrets of Arduino PWM. Acesso em 18 de Novembro de 2013, disponível em: http://www.arcfn.com/2009/07/secrets-of-arduino-pwm.html

Site: OpenCV API Reference. Acesso em 31 de Outubro de 2013, disponível em: http://docs.opencv.org/2.4.6/modules/core/doc/intro.html

Site: Arduino - ArduinoBoardUno. Acesso em 15 de Novembro de 2013, disponível em :http://www.arduino.cc/en/Main/arduinoBoardUno

Site: Comunicação Serial entre Arduinos. Acesso em 18 de Novembro de 2013, disponível em: http://arduinizando.blogspot.com.br/2013/02/comunicacao-serial-entre-arduinos.html