UNIVERSIDADE FEDERAL DO MARANHÃO
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
DEPARTAMENTO DE INFORMÁTICA
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

EDUARDO DE JESUS COELHO REIS

CONTROLE DO CURSOR DO MOUSE ATRAVÉS DOS OLHOS PARA PESSOAS COM NECESSIDADES ESPECIAIS

SÃO LUÍS
2014

EDUARDO DE JESUS COELHO REIS

CONTROLE DO CURSOR DO MOUSE ATRAVÉS DOS OLHOS PARA PESSOAS COM NECESSIDADES ESPECIAIS

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. Aristófanes Corrêa Silva

SÃO LUÍS

2014

Reis, Eduardo de Jesus Coelho

Controle do cursor através dos olhos para pessoas com necessidades especiais / Eduardo de Jesus Coelho Reis. – São Luís, 2014.

70 f.

Orientador: Aristófanes Corrêa Silva.
Monografia (Graduação) – Curso de Ciência da Computação da Universidade Federal do Maranhão, 2014.

1.Visão computacional. 2. Rastreamento do olho (Ciência da computação). 3. Template matching (Técnica). 4. Inclusão digital – Portadores de deficiência. I. Título.

CDU 004.932:301.173-056.26

EDUARDO DE JESUS COELHO REIS

CONTROLE DO CURSOR DO MOUSE ATRAVÉS DOS OLHOS PARA PESSOAS COM NECESSIDADES ESPECIAIS

Monografia apresentada ao curso de Ciência da Computação da Universidade Federal do Maranhão para obtenção do grau de Bacharel em Ciência da Computação.

Aprovada em: 13 / 01 / 2014

BANCA EXAMINADORA

Aristófanes Corrêa Silva

Prof. Dr. Aristófanes Corrêa Silva (Orientador)
Doutor em Informática
Universidade Federal do Maranhão

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

João Dallyson Sousa de Almeida

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

A Deus em primeiro lugar, meus familiares,
irmãos da igreja, colegas de classe,
amigos e professores.

AGRADECIMENTOS

Primeiramente eu agradeço a **Deus**, quem me concedeu todas as oportunidades e possibilitou as condições necessárias para que eu estivesse aqui, concedendo-me salvação, vida, perdão, paz, alegria, esperança e disposição. A Ele toda honra e toda glória.

Agradeço também à minha avó **Socorro** e meu falecido avó **João,** que proporcionaram-me todos os recursos em todas as fases da minha vida, até então.

Assim também agradeço aos meus pais **Francisco** e **Nilciléia**, que sempre me aconselharam, de forma que aquilo que sou hoje, também é fruto da educação que recebi.

Meus amigos e colegas tanto da igreja quanto os da universidade, que me acompanharam e auxiliaram e apoiaram na minha formação, aos colegas de pesquisa, em especial **Samuel Chaves, Eduardo Batista, Arthur Bernardo, Tarcísio Sousa, Sasha Nícolas e Wener Sampaio** por todo apoio e tempo disponibilizados me dando auxilio diante dos desafios na minha formação acadêmica

Também agradeço ao **Antônio Busson, Suellen Motta**, e principalmente ao **Pedro Diniz,** por toda a contribuição que deram neste trabalho. Encorajando, apresentando-me excelente sugestões assim como participaram em trabalho anteriores.

Meus professores, **Paulo, Anselmo, Carlos Salles, Geraldo, Auxiliadora**, dentre outros, pelo conhecimento, ajuda, ética, comprometimento e conselhos, que me proporcionaram, e em especial ao professor **Aristófanes**, por sempre acreditar no potencial dos seus alunos, pela orientação nessa monografia e em diversos outros trabalhos no decorrer da graduação.

“O nosso problema não é a fraqueza, mas é a falta de consciência da nossa fraqueza.”

Paul Washer

RESUMO

Segundo o IBGE, foi estimado que, por volta de 2000, 7% dos brasileiros estavam sofrendo com algum tipo de deficiência motora. Dentro desse grupo estão incluso deficientes por falta de membro como também portadores de algum tipo de paralisia. Para pessoas daquela época e também dos dias atuais, estar incluído na sociedade tem sido uma tarefa cada vez mais desafiadora, pois possuem inúmeros obstáculos em acompanhar a quantidade, crescente, de informações disponíveis atualmente à população na internet. E isso é algo relevante para o desenvolvimento educacional, assim como a capacitação profissional e a inclusão digital. Apesar da incomparável evolução que tiveram os computadores nas últimas décadas tanto a nível de hardware como de software, eles ainda apresentam as mesmas restrições para pessoas com necessidades especiais, como, por exemplo, tetraplégicos. Para essas pessoas, realizar simples ações podem ser tornar uma tarefa árdua. Tendo isso em mente, esse trabalho apresenta uma metodologia computacional para auxiliar tais indivíduos no controle do cursor do mouse através de movimentações realizadas pelo olho, a fim de ser uma alternativa menos exaustiva. Para tanto, é necessário reconhecer e rastrear os movimentos do olho humano, em conjunto com a face, com o intuito de reconstruí-los em operações do cursor. Como também, considerando o clique, que pode ser analisado a partir do piscar do olho.

Palavras-chaves: Visão Computacional, Rastreamento de Olho, Template Matching, Detecção de Piscada, Transformada de Hough, Filtro de Kalman.

ABSTRACT

According to IBGE, it was estimated that in the year 2000, 7% of the Brazilian population was suffering from physical disabilities. This statistic includes those who had suffered limb loss or had any kind of paralysis. For those people then and also in present day, to be included in society has become more and more of a challenging task due to the increasing amount of information that is only available via internet. Keeping up with the abundant sources of knowledge on the web is extremely simple for the general population but for those with physical limitations it's a different story. These days, having access to these databases is extremely vital to educational development as well as professional qualification and digital inclusion. Despite the incredible advancement in both hardware and software over the last decade, for those with physical disabilities such as tetraplegia, the same restrictions exist today. Accomplishing simple actions that many of us take for granted can easily turn into a frustrating and often impossible task. With this in mind, a computational methodology to assist people in such situations by using their eye movement to control the mouse cursor would eliminate the need to use their arms. This would provide a less exhausting way to access the web and it would make simple tasks simple, as they should be. The movement of the human eye would be tracked, and their faced recognized in the system in order to reconstruct both into cursor operations. In addition to this, the click can also be achieved by a blink of the eye.

Keywords: Computer Vision, Eye Tracking, Template Matching, Blink Detection, Hough transform, Kalman Filter.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

Conforme uma pesquisa realizada pela FGV (MEIRELLES, 2013), estima-se que existem no Brasil 118 milhões de computadores em uso, o que representa uma média de 3 computadores para cada 5 habitantes. Calcula-se que em 3 anos a média será de um computador por habitante. O acesso a internet e a troca de informações tiveram um aumento expressivo se comparado com os meios de comunicações convencionais. O estudo realizado trimestralmente pela AKAMAI (2013), aponta que, nos dispositivos móveis, por exemplo, o tráfego de dados vem crescendo significantemente de 2007 à 2013 se comparado com o tráfego de voz. Atingindo 1600 PetaBytes, ou seja, 1600 milhões de GigaBytes, por mês no segundo trimestre de 2013.

O acesso à informação é fundamental para o desenvolvimento da sociedade. A tecnologia está presente em praticamente todo os lugares, ao ponto de ser imprescindível no dia a dia das pessoas. Infelizmente, as interfaces de uso dos dispositivos utilizados ainda apresentam certos tipos de limitação. Portadores de necessidades especiais, por exemplo, veem estas como um obstáculo para o uso da tecnologia.

Segundo o IBGE (2010), 23,9% dos brasileiros sofrem com algum tipo de deficiência. Também foi estimado que 7% da população são portadores de algum tipo de deficiência motora. Dentro dessa parcela, estão inclusas pessoas que perderam um membro ou parte dele, assim como pessoas que possuem tetraplegia [1], paraplegia [2] ou hemiplegia [3] permanente. O censo realizado pelo IBGE (2000) estima que o número de deficientes no Brasil por falta de membro era de 0,28% e por uma das paralisias supracitadas era de 0,55%.

Esses números representam uma minoria da população, porém significativa. O decreto de lei 5296 de 2 de dezembro de 2004, estabelece os critérios básicos para a promoção da acessibilidade. O artigo 8, no inciso II, caracteriza como barreira nas

[1] Paralisia que afeta os membros superiores e inferiores.
[2] Perda de movimento dos membros inferiores.
[3] Paralização de metade do corpo. Comprometendo a movimentação de metade da face, braço e perna do mesmo lado.

comunicações e informações “qualquer entrave ou obstáculo que dificulte ou impossibilite a expressão ou o recebimento de mensagens por intermédio dos dispositivos, meios ou sistemas de comunicação, sejam ou não de massa, bem como aqueles que dificultem ou impossibilitem o acesso à informação.”

A deficiência devido à falta de braços ou mãos, ou até a incapacidade de usá-los com precisão impossibilita o uso de certas ferramentas que requerem, para seu funcionamento, o uso específico destes membros. Entre essas ferramentas se destaca o computador. Lamentavelmente, se tratando dos tradicionais computadores pessoais (i.e. desktops e laptops) como meios de comunicação e acesso a informação, navegar na internet pode tornar uma tarefa árdua ou até impossível para pessoas com paralisia ou perda dos membros superiores, sem a possibilidade de uso das interfaces convencionais, como o básico conjunto teclado e mouse ou telas sensíveis o toque. Sabe-se que um dos papéis do *mouse* é a navegação, que se dá pela movimentação do cursor, visível na tela do computador, como também a seleção ou movimentação de algo selecionado na tela através do clique. No entanto, não é difícil perceber que o mouse é usado por mera conveniência. Em *tablets*, por exemplo, ele não é necessário assim como em *notebooks* dotados de *touch pad*. Claro que em todos os casos usam-se as mãos, mas há outro ponto em comum nessas interfaces que pode ser levado em consideração: em todos esses casos há uma interface para o rastreamento de um objeto. Esse objeto em questão, então, não precisa ser necessariamente uma mão ou algum objeto manuseável.

Com a popularização das webcams, que são câmeras de vídeo de baixo custo geralmente utilizadas em vídeo chamadas, e o avanço das técnicas de visão computacional e processamento de imagem, tem surgido aplicativos que permitem uma interação alternativa com o usuário. Atualmente, existe uma série de linhas de pesquisa voltadas ao rastreamento de objetos em imagens de vídeos. Uma alternativa para substituir o *mouse*, é utilizar o rastreamento de objetos selecionados e reproduzir seus movimentos no cursor.

Uma opção interessante e simples, por exemplo, para as pessoas com limitações físicas dos membros superiores seria a utilização dos olhos como ferramenta de rastreamento de objetos. A vantagem de usar o olhar como referência da movimentação do cursor, é que o usuário poderia olhar precisamente no ponto em que ele deseja acessar.

Essa ideia mostra-se mais confortável do que o uso do mouse, inclusive para pessoas sem limitações.

Esse trabalho propõe uma metodologia computacional que substitua o papel do *mouse* na movimentação de cursor e clique, através de rastreamento e análise dos olhos em imagens de vídeo. Essa metodologia será desenvolvida utilizando a biblioteca de visão computacional OpenCV (BRADSKI; KAEHLER, 2008), que implementa funções de processamento de imagens, somada às técnicas de reconhecimento de padrões como o classificador em cascata, *Template Matching*, a transformada de Hough, assim como o método matemático Filtro de Kalman. Além disso, para dar suporte ao controle de cursor, também serão usadas funções específicas do sistema operacional. No caso, a metodologia proposta será aplicada para os sistemas *Windows e MAC OS X*.

1.1 Motivação

Computadores são ferramentas indispensáveis, tanto no contexto educacional quanto no trabalho. Facilitar o seu uso às pessoas com limitações físicas seria prover essas pessoas recursos para o aprendizado, capacitação profissional, e a própria inclusão digital. Para isso, é necessário soluções que substituam periféricos que requerem o uso das mãos para seu funcionamento.

1.2 Objetivos

Nessa seção estão contidos os objetivos gerais e específicos deste trabalho.

1.2.1 Objetivos Gerais

Desenvolver uma metodologia computacional que detecte e rastreie o olho do usuário, analise-os e reproduza seus movimentos em operações do cursor do mouse, de forma sincronizada, a fim de que o cursor se desloque à uma localização desejado na tela pelo usuário, e como também reproduzir o clique possibilitando selecionar e arrastar objetos na tela. Por meio dessa metodologia, busca-se oferecer uma ferramenta simples e eficiente para pessoas com limitações físicas, que permita substituir o *mouse* na movimentação do cursor e clique.

1.2.2 Objetivos Específicos

Para que seja concretizado o objetivo geral, alguns objetivos específicos devem ser alcançados:

- Detectar o olhos utilizando um método proposto por Viola e Jones (2004) que utiliza um classificador em cascata;
- Rastrear o olho detectado utilizando o *Template Matching*;
- Sincronizar movimentação do cursor ao rastreamento utilizando funções para ambos sistemas operacionais *Windows e MAC OS X*;
- Otimizar a movimentação do cursor através do Filtro de Kalman;
- Detectar piscos do olho utilizando Transformada de Hough;

1.3 Trabalhos Relacionados

Existem diversas linhas de pesquisas relacionadas às diferentes técnicas aplicadas na metodologia desse trabalho.

Primeiramente, frisamos o trabalho de Peng et al. (2005), que discorre sobre um algoritmo robusto para detecção de olhos em faces frontais. Em seu artigo, ele propõe a combinação de duas técnicas. A primeira, limitaria a região de busca dos olhos utilizando projeções verticais e horizontais partir de um gradiente aplicado sobre a imagem e a delimitação é feita a partir dos picos encontrados nessas regiões. A segunda técnica é o *Template Matching*, que é aplicado nas regiões delimitadas com o propósito de encontrar o centro da íris.

Peng et al. (2005) fala também sobre o custo computacional, em que o *Template Matching* tem a menor eficiência quando é aplicado sobre a imagem inteira. Além disso, a priori, as dimensões do olho na imagem são desconhecidas. Por esse motivo, para se obter um resultado desejado como *Tempalte Matching*, é necessário aplicá-lo repetidas vezes utilizando *templates* de diferentes tamanhos, até encontrar o casamento ideal, o que diminuiria mais ainda sua eficiência. Apesar disso, Peng et al. (2005) explica que essa técnica pode ser eficiente se houver uma limitação de uma região de interesse.

Com a técnica para limitação de região utilizada por Peng et al. (2005), é possível obter duas regiões contidas na face, em que cada uma dessas regiões contem um dos olhos. Em seguida, a partir das dimensões dessas regiões, é possível estimar as dimensões do *template* do olho a ser utilizado para a varredura. Levando em consideração a diminuição da região de busca, e a estimação do tamanho do *template* a ser utilizado, tem-se consequentemente uma melhora na eficiência do algoritmo.

Um outro trabalho voltado para o rastreamento dos olhos em imagens de vídeo é o de Jing et al. (2010). Que utiliza uma combinação maior de técnicas, entre elas temos a utilização de características Haar-Like, assim como o *Template Matching* Adaptativo sendo uma variação do *Template Matching* convencional. A adaptação do *template* utilizada por Jing et al. (2010) se baseia em definir o *template* a ser utilizado no próximo *frame* como o resultado obtido pelo *Template Matching* no *frame* atual. Jing et al. (2010) também enfatiza em seu artigo sobre a importância do desenvolvimento de interfaces alternativas para pessoas com deficiências nas mãos, e como o rastreamento dos olhos apresenta uma importante nessa área.

A metodologia utilizada por Jing et al. (2010) pode ser entendida através de 5 etapas:

1. Detecção Facial, que utiliza o algoritmo de detecção de face baseado em características Haar-*Like* para obter a região da face.
2. Detecção da região dos olhos. Esta região é utilizada a fim de restringir o escopo da busca dentro da região da face, uma região geral do olho pode ser atribuída a partir das distribuições geométricas da estrutura da face humana. Portanto tempo computacional é reduzido.
3. Definição do *template* inicial. É utilizado o método de detecção de olho baseado na transformada de Gabor e a região da íris é utilizada como *template* inicial.
4. Adaptação do *template*. É mantida a íris detectada no *frame* anterior, que é utilizada para substituir o *template* original utilizada na nova comparação a ser realizada no *frame* atual.

5. *Template Matching*. A localização do olho em cada *frame* pode ser obtida então utilizando o *Template Matching* percorrendo apenas a região de cada olho.

Existem diversas outras técnicas aplicadas no campo da detecção de olhos ou elementos oculares, como é o caso de técnicas baseadas em formas geométricas. Chavez et al. (2006) propõe em seu artigo uma metodologia para segmentação da íris, que possui um formato de coroa circular. O algoritmo utilizado por Chavez et al. (2006) possui as seguintes etapas:

1. Encontrar os limites externos e internos da íris.
2. Segmentar a região encontrada, que possui a forma de uma coroa circular.
3. Normalização da região segmentada. Essa normalização é uma espécie de registro realizado na imagem, com o propósito de "desdobrá-la", para obter-se uma imagem retangular.
4. Identificação e reconhecimento de padrões. Essa imagem posteriormente será comparada, utilizando métodos específicos, com um banco de imagens a fim de reconhecer quais outras imagens possuem o mesmo padrão.

Chaves (2006) utiliza imagens da região dos olhos aplicando na etapa 1 a transformada de Hough para detecção de círculos com a finalidade de encontrar o círculo mais externo à íris e o mais interno, no caso o círculo formado pela íris. Por possuir imagens próprias para esse fim, isto é, imagens com uma resolução do olho alta, a detecção dos limites externos e internos da íris é possível e bem eficiente. Por fugir do escopo desse trabalho, as demais etapas não serão aprofundadas.

Por fim, Diniz (2012), apresenta uma metodologia onde é possível realizar a movimentação do cursor do mouse a partir do rastreamento do olho. Não buscando solucionar problemas como a detecção dos olhos, mas ao invés disso, aborda uma metodologia que a partir da localização do olho no *frame* inicial faz um rastreamento utilizando o *Template Matching* Adaptativo transpondo os movimentos extraídos durante o rastreamento para o posicionamento do cursor.

Como dito anteriormente, o *Template Matching* pode ser custoso, e assim como Peng et al (2005) e Jing et al. (2010), Diniz (2012) restringe uma região de busca onde será feita a varredura. Porém, esta região de busca é baseada unicamente na posição do olho no *frame* anterior. Diniz (2012) assume que no próximo *frame* a posição do objeto a ser rastreado estará nas proximidades de sua localização no *frame* atual. Sendo o posicionamento dessa região de busca atualizado a cada *frame*, sendo concêntrica com a região encontrada pelo *Template Matching*. Ao transpor a movimentação do olho para cursor do mouse, Diniz (2012) observa a existência de ruídos que na fase de rastreamento não possuem muita relevância. Mas que são indesejados, deixando a movimentação cursor imprecisa.

Como parte de sua metodologia, Diniz (2012) utiliza o Filtro de Kalman em cada uma das direção para suprimir os ruídos observados na movimentação do mouse, suavizando assim seu movimento, aumentando consideravelmente a precisão.

1.4 Organização do Trabalho

Este trabalho apresenta a seguinte organização:

No Capítulo 2, Fundamentação Teórica, segue informações importantes para o contexto e entendimento do trabalho, tais como explicações sobre *Template Matching*, filtro de Kalman, transformada de Hough, outros conceitos abordados de Processamento de Imagens, assim como a biblioteca utilizada OpenCV.

No Capítulo 3, Metodologia, explica-se a metodologia utilizada como ponto de partida para o desenvolvimento desse trabalho.

No Capítulo 4, Resultados, mostra-se a ferramenta desenvolvida, através dos resultados alcançados na aplicação da metodologia, junto como uma análise de seu desempenho.

No Capítulo 5, Conclusão, apresenta-se a conclusão do trabalho. Nela está contida uma retrospectiva do que foi falado na monografia como todo, uma discussão dos resultados obtidos e também sugestões para o prosseguimento e aprimoramento desse trabalho, tais como melhorias e adição de funcionalidades, dentre outros.

2 FUNDAMENTAÇÃO TEÓRICA

Esse capítulo aborda os conceitos de processamento de imagens utilizados no desenvolvimento deste trabalho.

2.1 Conceitos importantes de processamento de imagens

Nesta subseção, seguem alguns conceitos básicos, porém importantes sobre processamento de imagens, que serão úteis no decorrer do trabalho.

2.1.1 Representação Digital da Imagem

A imagem pode ser representada como uma função bidimensional f, e a amplitude $f(x, y)$ é o valor de sua intensidade no ponto (x, y). Chamamos de imagem digital quando x e y, assim como a intensidade de f possuem valores finitos e discretos. Computacionalmente uma imagem também pode ser representada por uma matriz de tamanho $M\ x\ N$, e cada célula dessa matriz é chamada de pixel, abreviação do inglês para *picture element*. A quantidade de bits utilizada para representar este valor é chamada profundidade do pixel, quanto maior a profundidade, maior o número de diferentes intensidades que é possível representar. Em imagens coloridas os bits em cada pixel é interpretada de acordo com o sistema de cor que se encontra codificada a imagem. As imagens a serem utilizadas neste trabalho são em escala de cinza, e as mesmas foram capturadas através da webcam utilizando as funções da biblioteca OpenCV.

2.1.2 Equalização de Histograma

Segundo Filho & Neto (1999), o histograma de uma imagem pode ser entendido por um conjunto de números indicando o percentual de pixels naquela imagem para um determinado nível de cinza. E geralmente é apresentado através de um gráfico de barras fornecendo a quantidade (ou porcentagem) de pixels correspondente em uma imagem. Através da visualização desse gráfico é possível verificar a qualidade do nível de contraste da imagem , assim como observar a intensidade do brilho médio, se a imagem é predominantemente clara ou escura.

Pode-se então obter cada elemento desse conjunto pela Equação 1.

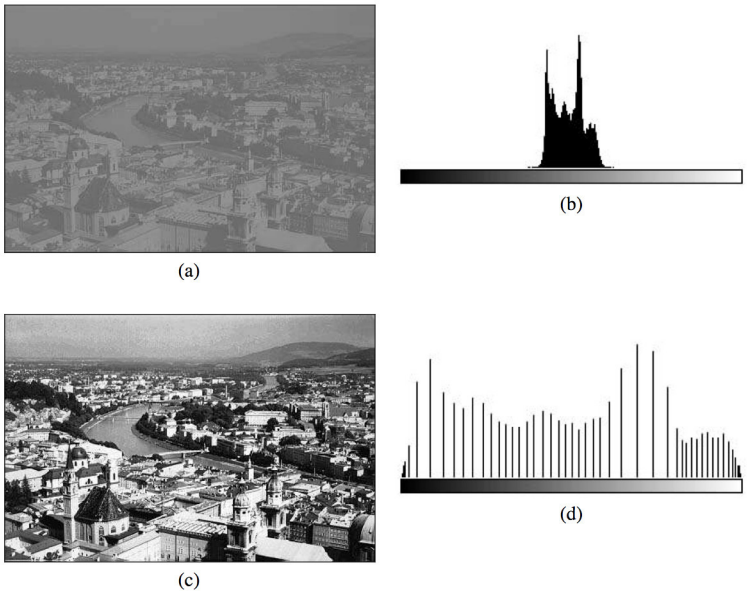
$$P_r(r_k) = \frac{n_k}{n} \quad (1)$$

Onde $0 \leq P_r(r_k) \leq 1$, r_k é o k-ésimo nível de intensidade em uma faixa $[0 - L)$, n_k é o número de ocorrências de pixels cujo valor de intensidade é r_k, e n é o número total de pixels na imagem. Em imagens em nível de cinza em cuja profundidade do pixel é 8 bits, L assume o valor de 256.

A equalização histograma é uma das técnicas mais utilizadas na literatura quando deseja-se obter um realce genérico de contrates. Tendo como finalidade redistribuir os valores dos tons de cinza, de modo a obter um histograma mais uniforme.

Um exemplo de histograma é a Figura 1, que apresenta em (a) a imagem original e seu respectivo histograma em (b). Como dito anteriormente é possível, através do histograma, observar que a imagem possui um baixo contraste, como também visualizar qual é seu brilho médio. Em (c) temos o resultado da imagem equalizada e seu respectivo histograma (d), agora com uma distribuição mais uniforme devido ao realce no contraste.

Figura 1 – Exemplo de utilização da equalização de histograma



Fonte: MARQUES FILHO; VIEIRA NETO, 1999, p. 64.

Apesar de sua utilização ampla para aprimoramento de contraste, a equalização de histograma tem como a principal característica o fato de não ser parametrizada. Podendo a

nova imagem equalizada ser obtida pela seguinte Equação 2, onde $T(\mathrm{r}_k)$ é a nova intensidade para o pixel de grau de intensidade k.

$$T(\mathrm{r}_k) = (\mathrm{L} - 1) \cdot \sum_{i=0}^{k} p_r(r_i) \qquad (2)$$

2.1.3 Imagem Integral

A Imagem Integral (*Integral Image*) é uma técnica de programação dinâmica por realizar uma série de operações armazenando os resultados para reuso futuro. A Imagem Integral consiste uma matriz de acumuladores que armazena a soma da intensidade dos pixels, permitindo rápidas operações em sub-regiões de uma imagem. O que é de grande utilidade em processamento de imagens uma vez que cálculos como média e soma podem ser realizados eficientemente, reduzindo o tempo de processamento de imagens.

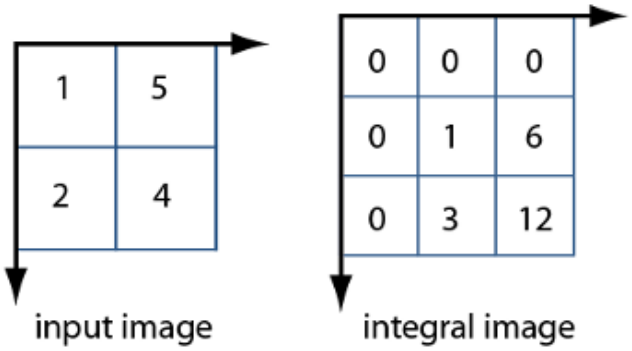
A Imagem Integral também é comumente utilizada para o cálculos das extrações de características baseadas em *Wavelets de Haar*, como proposto por Viola & Jones (2004) .

A Imagem Integral no pixel de posição (x, y) contém a soma dos pixels acima e a esquerda de (x, y) , como também o próprio valor de (x, y) na imagem original. Matematicamente é expresso pela Equação 3.

$$ii(x, y) = \sum_{x\prime \leq x, y\prime \leq y} (x', y') \qquad (3)$$

Por motivos computacionais, a imagem integral possui dimensões $N + 1\ x\ M + 1$, com a primeira linha e coluna com valores preenchidas com zeros. A Figura 2 demostra um exemplo desta operação.

Figura 2 – Representação da Imagem Integral



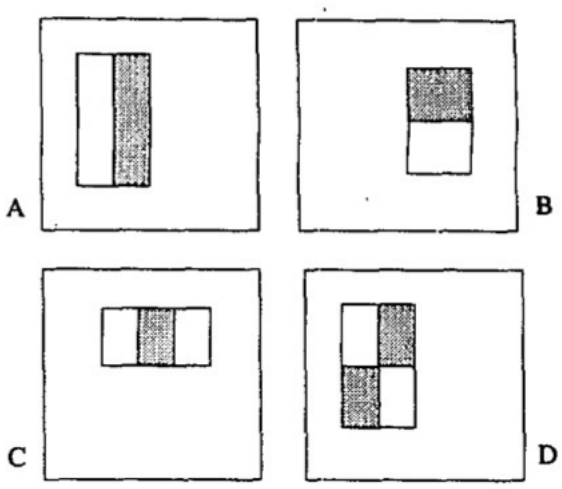
Fonte: Mathworks[4]

2.2 Classificador em cascata baseado em características *Haar* para detecção de objetos

O método de detecção descrito nessa sessão foi inicialmente proposto por Viola & Jones (2001). As características *Haar* ou *Haar features* utilizam um conjunto de características baseado nas *Haar Wavelets* que aplicam o uso de áreas retangulares. Viola & Jones (2001) adaptou a ideia utilizando mais de uma área retangular, desenvolvendo as chamadas *Haar-like features*. Estas consistem em somar a intensidade de pixels uma dessas regiões, e utilizar como característica a diferença desses resultados para áreas adjacentes. A Figura 3 mostra as *Haar-like features* utilizadas por Viola & Jones (2001), onde A e B mostram características retangulares compostas por 2 retângulos, C possui três retângulos e D quatro retângulos. Em que, em cada uma dessas características, a região a que cada retângulo pertence é indicada pela sua cor de preenchimento nas figuras apresentadas.

[4] Disponível em: <http://www.mathworks.com/help/vision/ref/integralimage.html>. Acesso em 9 de Janeiro de 2014.

Figura 3 – Exemplos de características retangulares .



Fonte: VIOLA; JONES, 2001, p. 512.

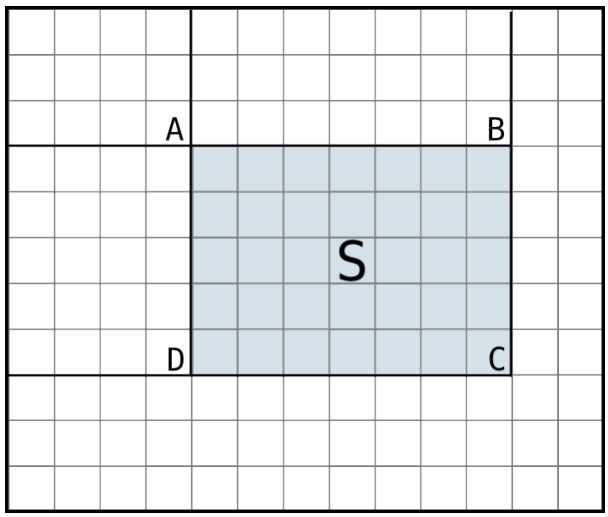
Para realizar a detecção em tempo real, a extração de milhares de características em sub-regiões de cada quadro em um vídeo seria algo muito custo. Para a rápida computação dessas características, Viola & Jones (2001) propôs também a utilização de uma Imagem Integral, que pode servir como uma tabela de consultas (*lookup table*).

Assim, a soma dos valores da área sombreada na Figura 4, pode ser calculado por:

$$Sum = I(C) + I(A) - I(B) - I(D) \qquad (4)$$

Sendo A, B, C, D, pontos pertencentes a imagem integral da Figura 4.

Figura 4 – Cálculo da área do retângulo através da Imagem Integral



Fonte: NONGNU.ORG[5].

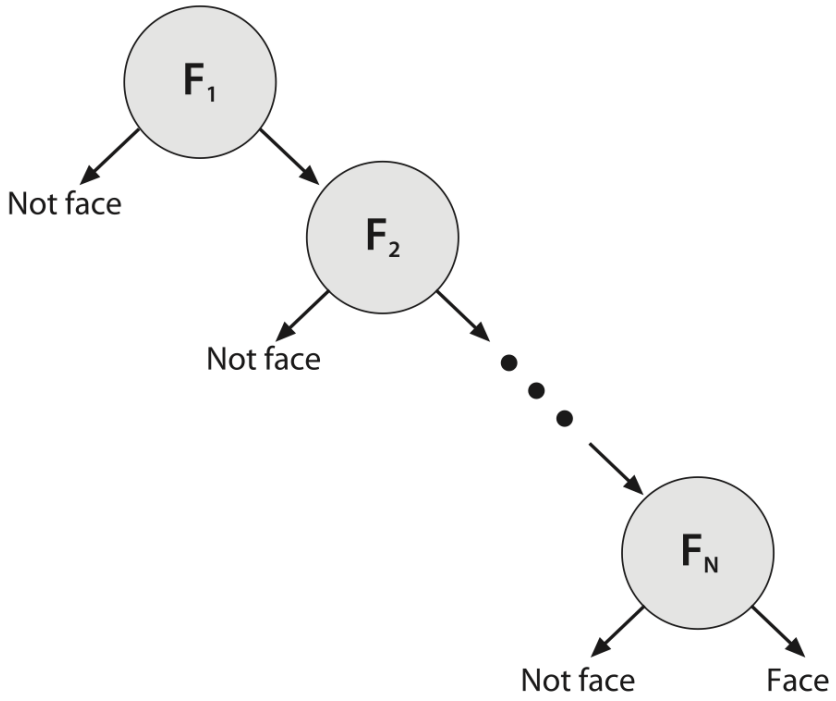
Dessa forma, a imagem integral é computada uma única vez, e para cada característica *Haar-like* são necessárias 6 consultas para características com 2 retângulos, 8 para 3 retângulos e 9 consultas para características com 4 retângulos.

Mesmo com a eficiência obtida pelo uso da Imagem Integral, a quantidade dessas características a serem extraídas são muitas. Em uma sub-região de 24 x 24 pixel existem em torno de 160000 delas (VIOLA; JONES, 2004). Por ser alto o número de características a serem avaliadas, para uma detecção em tempo real utiliza-se uma variação do algoritmo de aprendizagem *AdaBoost*, que seleciona as melhores características a serem utilizadas para treinar os classificadores.

Para a detecção, os classificadores fortes são organizados em formato de cascata de acordo com sua complexidade. Se em algum estágio um classificador rejeita uma sub-janela em uma análise, aquela região não é passada a diante, sendo rejeitada de imediato como face e a busca pelo objeto passa a ser feita na próxima sub-janela. O classificador tem, portanto, o formato de uma árvore degenerada, conforme a Figura 5.

[5] Disponível em: <http://www.nongnu.org/rapp/doc/rapp/integral.html>. Acesso em 10 de Janeiro de 2014.

Figura 5 – Cascada de rejeição utilizada no classificador Viola-Jones



Fonte: BRADSKI; KAEHLER, 2008, p. 510.

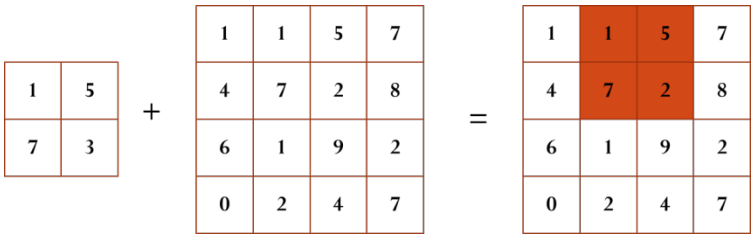
2.3 Template Matching

Template Matching (JING et al., 2010) é uma técnica de reconhecimento de padrões que baseia-se na comparação entre imagens. O *Template Matching* realiza uma varredura em uma dada imagem em busca da área que mais se aproxima a um padrão, definido pela outra imagem chamada *Template*.

Ocorre da seguinte forma: para cada sobreposição do *Template* na imagem, é gerado um valor resultante a partir de uma medida de comparação denominada métrica de similaridade. Dependendo da métrica usada, é definido uma faixa de valores (limiar), se o grau de similaridade satisfizer tal faixa, a região será caracterizada como o objeto em questão.

Computacionalmente uma imagem pode ser vista como uma matriz, através dessa matriz e considerando uma simples métrica (o somatório da diferença dos valores correspondentes em ambas), pode-se entender essa técnica a partir das ilustrações nas Figura 6.

Figura 6 - *Template Matching* utilizando matrizes



Fonte: DINIZ 2012, p. 19.

No exemplo da Figura 6, a medida em que o *template* percorre a imagem, é executada a comparação com as posições que podem ser sobrepostas pelo *template*. Na soma da diferença dos valores, os resultados de menor valor correspondem aos maiores níveis de similaridade. Nesse caso, o maior nível de similaridade é encontrado na região em laranja.

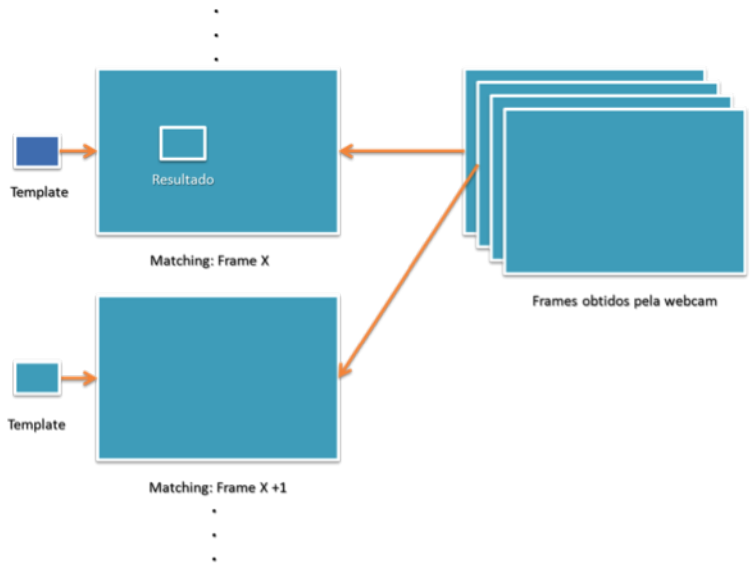
A maneira em que a comparação é feita, depende diretamente da métrica de similaridade utilizada. Posteriormente nesse trabalho serão explicadas algumas dessas técnicas.

2.3.1 Template Matching Adaptativo

O *Template Matching Adaptativo* (JING et al., 2010), assim como o *Template Matching*, analisa uma imagem em busca de um determinado objeto, porém através de um padrão que é dinâmico, se adequando conforme as características da imagem. Essa adaptação pode ser gerada a partir de características obtidas a partir da própria imagem. No caso de um rastreamento de objetos feito em vídeos, os resultados obtidos no processamento do *frame* atual poderiam modificar o *template*, adequando-o para ser aplicado nos próximos *frames*. Jing et al. (2010) realiza essa adaptação utilizando como o próximo *template* a região extraída a partir dos resultados obtidos pelo *Template Matching*.

A Figura 7 descreve esse processo de uma forma genérica, onde tem-se a sequência de *frames* obtidos pela webcam. Sobre esses *frames* está sendo realizado o rastreamento de um objeto utilizando o *Template Matching Adaptativo*, em que para um dado *frame X*, o resultado obtido é extraído e utilizado para a busca realizada no *frame* $X + 1$.

Figura 7 - *Template Matching Adaptativo* usado para rastreamento de objeto em vídeo



Fonte: DINIZ, 2012, p. 22.

2.3.2 Metricas de Similaridade

A métrica de similaridade, também chamadas de métodos de comparação ou *Matching Methods* quantifica o grau de similaridade entre dois objetos, ou seja, o quão "distante" eles estão. No *Template Matching*, durante a varredura, a métrica de similaridade determinará a maneira que a comparação será feita, afetando diretamente os resultados.

Bradski & Kaehler (2008) apresenta algumas das métricas mais comuns utilizadas. Uma delas é o quadrado da diferença ou *Square Difference*, definida pela Equação 5, onde T é o *Template*, I é a imagem original e R é o resultado.

$$R_{sq\_diff}(x,y) = \sum_{x',y'} [T(x',y') - I(x+x',y+y')]^2 \qquad (5)$$

Nesse caso, uma comparação perfeita resultaria em um valor 0, e quanto maior o resultado, pior é o grau de similaridade. Outra métrica utilizada é a correlação ou *correlation matching*, que consiste basicamente em multiplicar o *template* pela imagem. Neste caso, quanto maior o resultado, maior o grau de similaridade. A comparação por correlação é feita pela Equação 6 .

$$R_{ccorr}(x,y) = \sum_{x',y'} [T(x',y') \cdot I(x+x', y+y')]^2 \quad (6)$$

Também são apresentados métodos de normalização, que amplamente utilizados pois ajudam a reduzir os efeitos de diferenças de luminosidade entre o *template* e a imagem. O coeficiente de normalização é dado pela Equação 7.

$$Z(x,y) = \sqrt{\sum_{x',y'} T(x',y')^2 \cdot \sum_{x',y'} I(x+x', y+y')^2} \quad (7)$$

As métricas de similaridades exibidas anteriormente, da mesma forma, podem ser apresentadas em uma forma normalizada, conforme a Tabela 1.

Tabela 1 – Métricas de similaridade normalizadas

Métrica de Similaridade	Valor computado
Normalização do Quadrado da diferença	$R_{sq\_diff\_normed}(x,y) = \frac{R_{sq\_diff}(x,y)}{Z(x,y)}$
Normalização da Correlação	$R_{ccor\_normed}(x,y) = \frac{R_{ccor}(x,y)}{Z(x,y)}$

Fonte: Adaptado de BRADSKI; KAEHLER, 2008, p. 216.

2.4 Transformada de Hough

A transformada de Hough (BRADSKI; KAEHLER, 2008) é uma técnica comumente utilizada em processamento de imagens para identificar formas geométricas básicas a partir de suas equações. Foi inicialmente proposta para a detecção de retas, mas foi estendida, posteriormente, para identificação de outras formas, tais como círculos e elipses. Na maioria dos casos a utilização de um detector de bordar pode ser utilizado em uma fase inicial de pré-processamento. Entretanto, devido a imperfeições na imagem ou no resultado obtido pela detecção, podem exibir uma borda com falhas, ou apresentar ruídos que se assimilam à formas não esperadas. A transformada de Hough trata este problema

agrupando pontos que seriam de borda em grupos de objetos candidatos através de um processo de votação no plano dos parâmetros.

O caso mais simples da transformada de Hough é a detecção de retas. Uma reta pode ser descrita por $y = ax + b$, onde a é a inclinação da reta em relação ao $eixo\ x$ e b é o ponto onde a reta intercepta o $eixo\ y$. Uma reta pode ser representada pelo ponto (a, b) no plano dos parâmetros, porém representar retas verticais seria um problema, as retas $x = 0$ e $x = 1$, por exemplo, não seriam distinguíveis. Por isso utiliza-se coordenadas polares para esse fim, conforme a Equação 8.

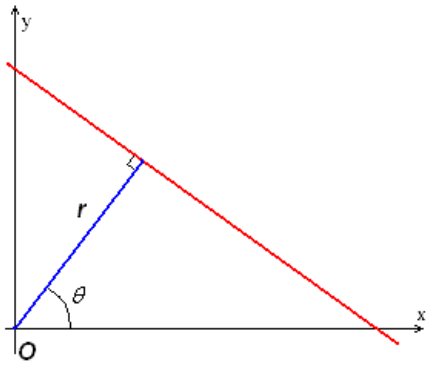
$$y = \left(-\frac{cos\theta}{sen\theta}\right)x + \left(\frac{r}{sen\theta}\right) \tag{8}$$

Que pode ser rescrito como:

$$r = xcos\theta + ysen\theta \tag{9}$$

Onde r é a menor distancia da reta para a origem, Figura 8.

Figura 8 – Reta em função do raio e do ângulo



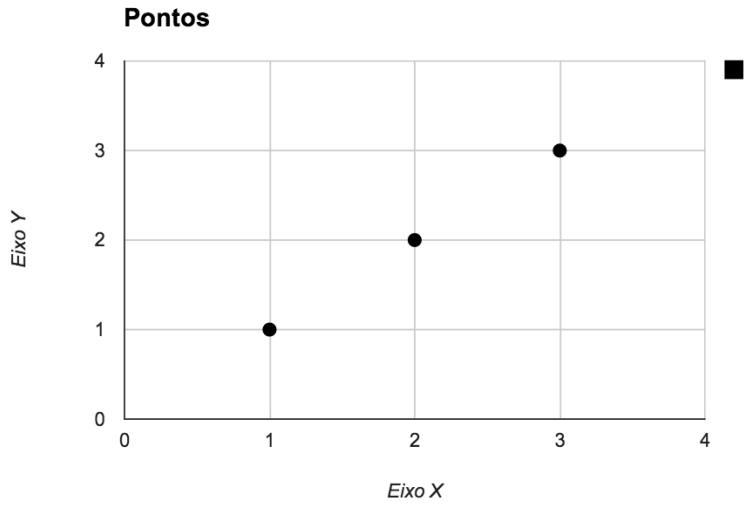
Fonte: WIKIPEDIA[6]

Associando, portanto, cada reta a um par (r, θ) que é único, se $\theta \in [0, \pi]$ e $r \in \mathbb{R}$.

Dada a Figura 9:

[6] Disponível em: <http://en.wikipedia.org/wiki/File:R_theta_line.GIF> Acesso em 7 de Janeiro de 2014.

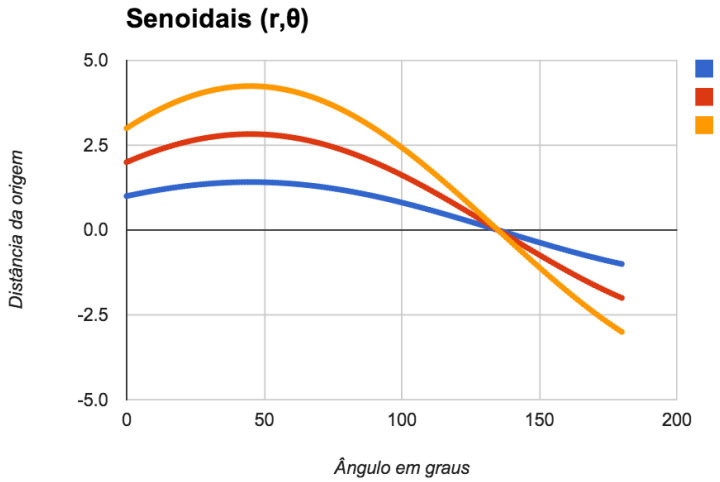
Figura 9 – Pontos em um gráfico representando os pixels de uma imagem que formam uma reta.



Fonte: Elaborada pelo autor.

Cada um desses pontos (x_o, y_o) possui uma curva senoidal no plano $(r\ x\ \theta)$ representando retas no formato (r, θ) que passam por cada um deles, conforme a Figura 10.

Figura 10 – Senoidais representando o conjunto de retas que interceptam um determinado ponto

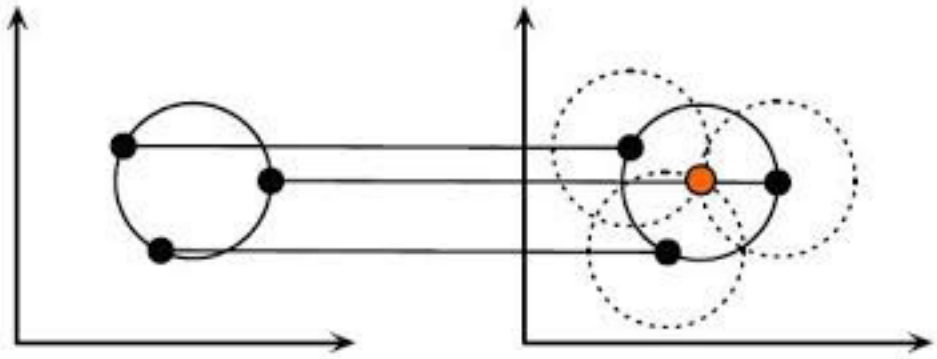


Fonte: Elaborada pelo autor.

Se as curvas correspondentes a dois ou mais pontos se sobreporem, a localização onde elas se cruzam corresponde a reta que passam por aqueles pontos. Um acumulador é utilizado para identificar quantos pontos pertencem àquela reta. Certamente, o conjunto de pontos que fazem parte de um segmento de reta na imagem irá produzir senoidais que se cruzam nos parâmetros daquela reta.

A transformada de Hough para detecção de círculo poderia funcionar de maneira semelhante, conforme pode ser observado a partir de Figura 11.

Figura 11 – Transformada de Hough para círculos



Fonte: Página da ELETRONICA.ORG[7].

Para tal fim, seria usado, semelhantemente, as equações paramétricas da circunferência, dada por:

$$x = a + rcos\theta \quad (10)$$

$$y = \mathrm{b} + \mathrm{r}sen\theta \quad (11)$$

Onde a e b são as coordenadas do centro da circunferência respectivamente nos eixos x e y.

Porém, se for tratado dessa forma o acumulador, que antes possuía duas dimensões ou variáveis e podia ser entendido como um plano, teria agora que ser substituído por um acumulador de três dimensões: uma para x, uma para y e outra para o raio do círculo. O que implicaria em consumo excessivo de memória e de tempo de processamento. A implementação disponível no OpenCV evita esse problema por meio de um método chamado de gradiente de Hough, descrito por Bradski & Kaehler (2008).

A detecção de círculos implementada pelo OpenCV utiliza de uma função com vários parâmetros dentre eles: o limiar superior para a detecção de bordas do filtro de Canny, o limiar inferior utilizado é a metade deste; uma distancia mínima entre o centro dos círculos

[7] Disponível em: <http://www2.eletronica.org/artigos/robotica/reconhecimento-de-objetos-em-tempo-real-para-futebol-de-robos/figura2.jpg> Acesso em 7 de Janeiro de 2014.

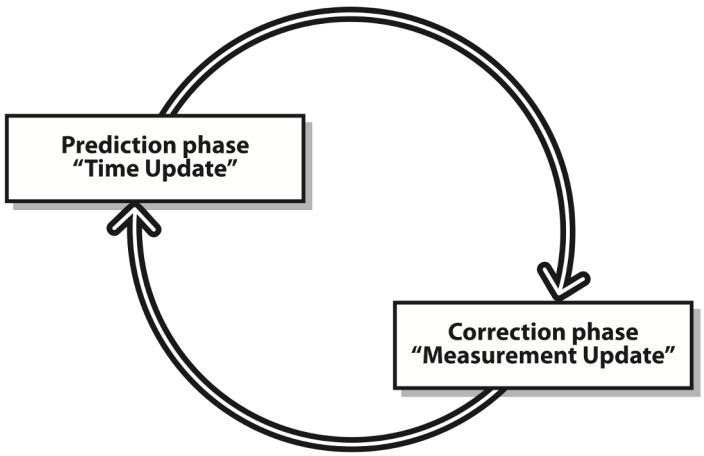
detectados; o tamanho mínimo do raio dos círculos detectados; assim como o tamanho máximo do raio.

2.5 Filtro de Kalman

O Filtro de Kalman (BRADSKI; KAEHLER, 2008) tem sido amplamente utilizado em diferentes contextos em processamento de sinais. Em um sistema onde é necessário determinar seu estado atual a partir de uma série de medições, tratar ruídos presentes nas leituras tem sido um grande desafio.

A meta do Filtro de Kalman é fazer uma estimativa do estado atual baseado na ponderação dinâmica entre o estado anterior e medição, que torna desnecessário a manutenção de um longo histórico. Em vez disso, atualiza-se iterativamente um modelo do sistema de estados e usa-se apenas esse modelo para a próxima iteração. E pode ser dividido em duas etapas, conforme a Figura 12.

Figura 12 – Fases do Filtro de Kalman



Fonte: BRADSKI; KAEHLER, 2008, p. 349.

A primeira etapa, conhecida por *prediction*, ou predição, é realizada antes da computação da medição realizada, no intuito de predizer o estado atual do objeto com base no estado anterior. A segunda etapa, também chamada de *correction*, ou correção, é calculada corrigindo o estado atual com base na nova medida.

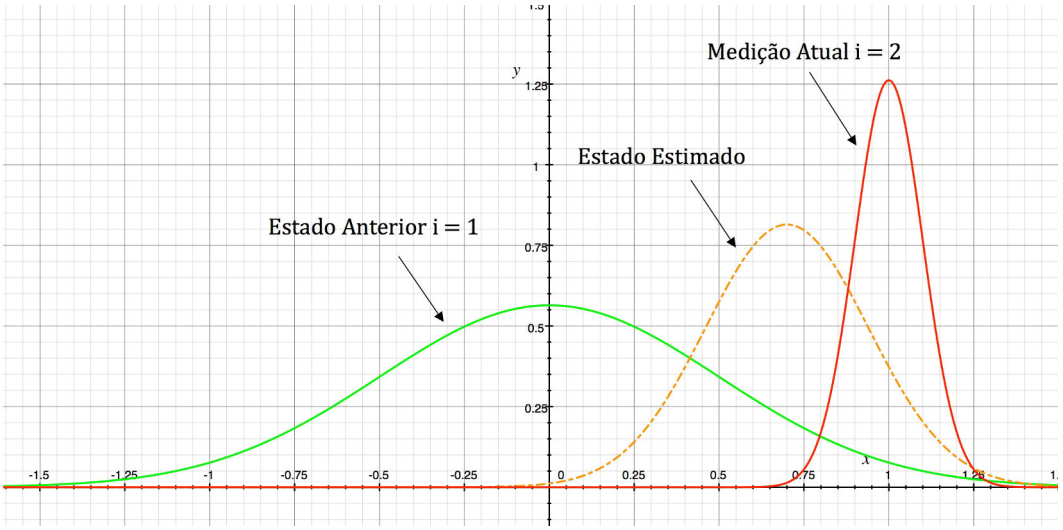
Para que a utilização do filtro de Kalman seja possível, é necessário que o sistema possua os seguintes requisitos:

- linearidade, possibilitando que o estado k, seja obtido a partir da multiplicação de uma matriz pelo estado $k-1$.
- o ruído presente na medição não seja em função do tempo, sendo sua amplitude precisamente modelada utilizando apenas uma média e covariância.

2.5.1 Teoria

Em um sistema de uma dimensão, devido ao ruído presente, seus valores podem ser representados por distribuições gaussianas, como apresentado na Figura 13. Nesta, o estado anterior é conhecido e uma nova medição é feita. Então o estado atual é estimado baseando-se nas probabilidades do estado anterior e da nova medida, ponderando esses valores para a estimação do estado atual.

Figura 13 – Combinando o conhecimento anterior com a nova medição para estimar o novo estado



Por serem aproximados, ambos os valores, do estado anterior e da medição, possuem média $\bar{x}_i$ e desvio padrão σ_i. Sendo que o desvio padrão define o quão confiável é aquela medida. Ambas as medidas podem ser expressas pela distribuição Gaussiana da Equação 12.

$$P_i(x) = \frac{1}{\sigma_i\sqrt{2\pi}}\, e^{\left(-\frac{(x-\bar{x}_i)^2}{2\sigma_i^2}\right)} \ (i = 1, 2) \qquad (12)$$

Sendo tais medidas distribuições gaussianas, espera-se que a distribuição de probabilidade de um valor x, considerando tais medidas, seja proporcional a $p(x) = p_1(x)p_2(x)$, conforme a Equação 13.

$$\begin{aligned} P_{12}(x) &\propto \exp\left(-\frac{(x-\bar{x}_1)^2}{2\sigma_1^2}\right)\exp\left(-\frac{(x-\bar{x}_2)^2}{2\sigma_2^2}\right) \\ &= \exp\left(-\frac{(x-\bar{x}_1)^2}{2\sigma_1^2}-\frac{(x-\bar{x}_2)^2}{2\sigma_2^2}\right) \end{aligned} \tag{13}$$

Esse produto gera uma nova gaussiana, cuja média e desvio padrão podem ser calculados. Sabendo que a distribuição apresenta um máximo no valor da média. Podemos obter o valor da média simplesmente igualando a primeira derivada da equação a zero. Obtendo a Equação 14.

$$\left.\frac{dp_{12}}{dx}\right|_{\bar{x}_{12}} = -\left[\frac{\bar{x}_{12}-\bar{x}_1}{\sigma_1^2}+\frac{\bar{x}_{12}-\bar{x}_2}{\sigma_2^2}\right]\cdot p_{12}(\bar{x}_{12}) = 0 \tag{14}$$

Uma vez que $p(x)$ nunca assume zero como valor, consequentemente, o valor entre colchetes é igual a zero. Rearranjando essa equação, tem-se que:

$$\bar{x}_{12} = \left(\frac{\sigma_2^2}{\sigma_1^2+\sigma_2^2}\right)\bar{x}_1 + \left(\frac{\sigma_1^2}{\sigma_1^2+\sigma_2^2}\right)\bar{x}_2 \tag{15}$$

Pela equação acima, tem-se que a media $\bar{x}_{12}$ é uma soma ponderada das duas outras medidas. Quanto maior o desvio padrão, menor o peso de uma dada medida.

Substituindo $\bar{x}_{12}$na equação $p_{12}(x)$, tem-se o valor para σ_{12}^2:

$$\sigma_{12}^2 = \frac{\sigma_1^2\sigma_2^2}{\sigma_1^2+\sigma_2^2} \tag{16}$$

Considerando as medidas (x_i, σ_i) passos no tempo, podemos computar as estimações $(\hat{x}_i, \hat{\sigma}_i)$ da seguinte forma. No passo 1, tem-se apenas a primeira medida, portanto $\hat{x}_1 = x_1$ e $\hat{\sigma}_1 = \sigma_1$. Substituindo na Equação 15, para o passo 2, temos:

$$\hat{x}_2 = \left(\frac{\sigma_2^2}{\hat{\sigma}_1^2 + \sigma_2^2}\right)\hat{x}_1 + \left(\frac{\hat{\sigma}_1^2}{\hat{\sigma}_1^2 + \sigma_2^2}\right)x_2 \quad (17)$$

Analogamente, calcula-se a o valor estimado $\hat{\sigma}_2^2$:

$$\hat{\sigma}_2^2 = \frac{\hat{\sigma}_1^2\sigma_2^2}{\hat{\sigma}_1^2 + \sigma_2^2} \quad (18)$$

Rearranjando ambos, temos:

$$\hat{x}_2 = \hat{x}_1 + \frac{\hat{\sigma}_1^2}{\hat{\sigma}_1^2 + \sigma_2^2}(x_2 - \hat{x}_1) \quad (19)$$

$$\hat{\sigma}_2^2 = \left(1 - \frac{\hat{\sigma}_1^2}{\hat{\sigma}_1^2 + \sigma_2^2}\right)\hat{\sigma}_1^2 \quad (20)$$

As Equações 19 e 20, permitem separar claramente a informação antiga (que se tinha antes a medição) da nova (obtida da ultima medição feita). Essa nova medição é também chamada innovation, ou inovação. Em adicional, observa-se a seguir o fator de ótimo de atualização iteração, conhecido como update gain ou ganho de atualização, representado pela Equação 21.

$$K = \frac{\hat{\sigma}_1^2}{\hat{\sigma}_1^2 + \sigma_2^2} \quad (21)$$

Usando esta definição, as equações 19 e 20 são reescritas da seguinte forma:

$$\hat{x}_2 = \hat{x}_1 + K(x_2 - \hat{x}_1) \quad (22)$$

$$\hat{\sigma}_2^2 = (1 - K)\,\hat{\sigma}_1^2 \quad (23)$$

Nas literaturas sobre o Filtro de Kalman, a estimativa no passo 2 é denotada por k e o passo 1 por $k - 1$.

2.5.2 Equações de Kalman

O filtro de Kalman aplica-se a qualquer modelo definido por uma função linear F de estados. Generalizando o modelo descrito anteriormente para um passo k, tem-se:

$$x_k = Fx_{k-1} + Bu_k + w_k \tag{24}$$

Onde, x_k é um vetor n-dimensional de componentes do estado e F é uma matriz $n\ x\ n$, também chamada de matriz de transferência, que multiplica x_{k-1}. O termo u_k é um vetor c-dimensional referente a entradas de controle, e B é uma matriz $n\ x\ c$ que relaciona essas entradas à mudança de estado. A variável w_k é uma variável aleatória, geralmente chamada de ruído de processo, que associada a eventos aleatórios ou forças que afetam o estado atual do sistema.

Geralmente são realizadas medições z_k, que podem ou não ser medidas diretas da variável de estado x_k. Por exemplo, quando deseja-se saber a velocidade de um automóvel, pode-se mensurar tanto utilizando um radar ou através do som do motor. No primeiro caso, z_k será x_k adicionado de algum ruído de medição, mas no segundo caso a relação não é direta da mesma forma. Podendo representar esta situação medindo o vetor m-dimensional de medições z_k dado por:

$$z_k = H_k x_k + v_k \tag{25}$$

Considerando a comparação anterior, apresenta-se o exemplo de um sistema onde são obtidas medidas de um carro em um estacionamento. Pode-se representar o estado do carro por duas variáveis de posição, x e y, e duas velocidades, v_x ev_y. Essas quatro variáveis seriam elementos do vetor de estado x_k, e F sendo sugerido por:

$$x_k = \begin{bmatrix} x \\ y \\ v_x \\ v_y \end{bmatrix}, F = \begin{bmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{26}$$

Entretanto, se utilizada uma câmera para medir o estado do caro, será possível apenas medir as variáveis de posição, tendo a medição a seguinte forma:

$$k_k = \begin{bmatrix} z_x \\ z_y \end{bmatrix}_k \tag{27}$$

Isto implica que a estrutura de H pode ser dada por:

$$H = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}_k \tag{28}$$

Neste caso, é provável que a velocidade não seja constante, considerando-se portanto um valor Q_k para refletir isso. De igual forma, é escolhido um valor R_k baseado na precisão dos valores medidos do posicionamento do carro.

A partir das expressões citadas, pode-se generalizar equações para ambas as fases do filtro de Kalman. Primeiramente calculando-se a estimação a priori x_k^- do estado, representado pelo sinal de menos sobre o estado, dado por:

$$x_k^- = Fx_{k-1} + Bu_k + w_k \tag{29}$$

Usando P_k^- para denotar a covariância do erro da estimativa do estado, a estimação a priori para este valor no tempo k pode ser obtido através dos valores no tempo $k-1$ por:

$$P_k^- = FP_{k-1}F^T + Q_{k-1} \tag{30}$$

Estas equações são as bases para a parte de predição da estimação, que diz "o que é esperado" baseado no que aconteceu. A partir daí, calcula-se o Kalman gain, ou ganho Kalman, o que indica o peso da nova informação comparado com o que já era conhecido.

$$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + R_k)^{-1} \tag{31}$$

Considerando um exemplo de uma dimensão, onde é medida apenas uma variável de posição, H_k é apenas uma matriz 1 x 1 contendo apenas um '1'. Então, se a nossa medida de erro é σ_{k+1}^2, então R_k, é também uma matriz 1 x 1 com aquele valor. De forma semelhante, P_k é apenas a variância σ_k^2. Então, a equação anterior é reduzida para:

$$K = \frac{\sigma_k^2}{\sigma_k^2 + \sigma_{k+1}^2} \tag{32}$$

Nota-se que o ganho é exatamente igual ao primeiramente descrito na subseção anterior. Permitindo calcular rapidamente a atualização dos valores de x_k e P_k quando uma nova leitura está disponível.

$$x_k = x_k^- + K_k(z_k^- - H_k x_k^-) \tag{33}$$

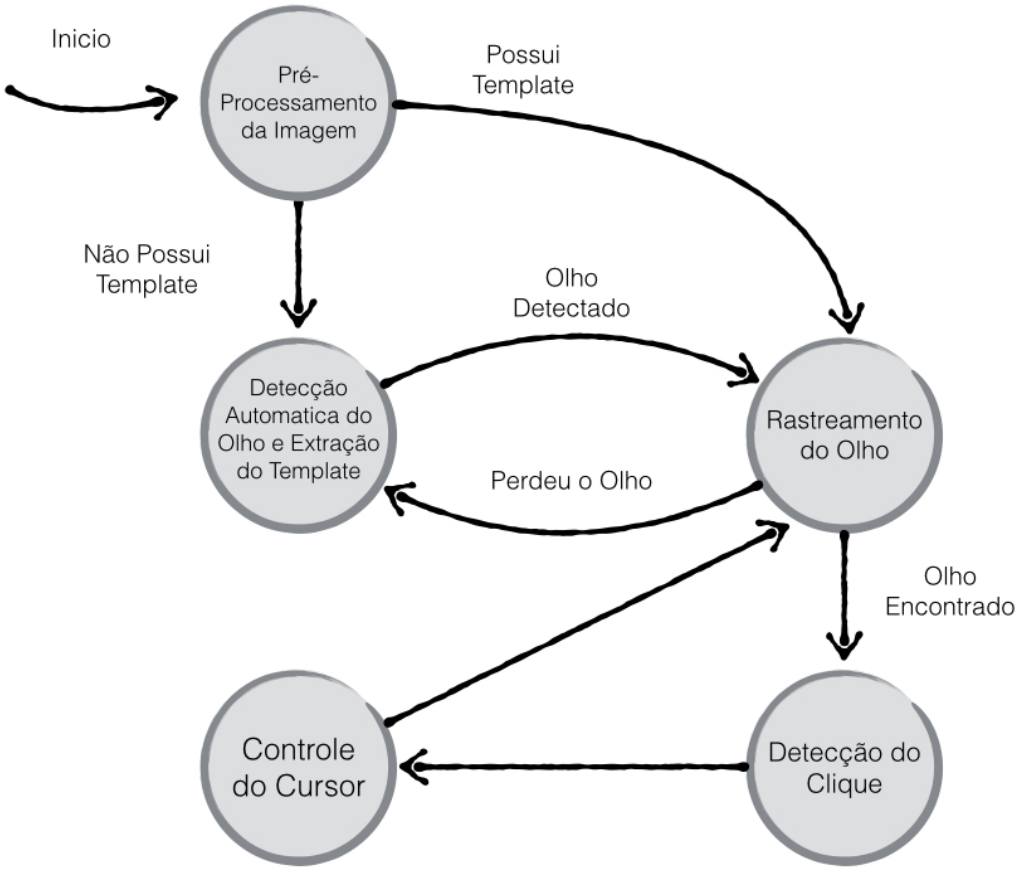
$$P_k = (I - K_k H_k) P_k^- \tag{34}$$

3 METODOLOGIA

Nesse capítulo, será descrita a metodologia usada. Será explicado como as técnicas e recursos apresentados na fundamentação teórica (Capítulo 2) foram empregados nas etapas da metodologia.

O diagrama da Figura 14 mostra a metodologia em cinco etapas.

Figura 14 – Diagrama da metodologia



Abaixo, segue uma explicação sucinta de cada uma das etapas da metodologia.

a) Pré-Processamento

Essa é a etapa inicial da metodologia, sendo também a mais simples por utilizar de técnicas triviais para fazer um pré-processamento na imagem. Após preparar o *frame*, é verificado se há ou não um *template* previamente definido. O que significa que a Detecção

do olho foi executada em algum *frame* anterior, de forma que o fluxo segue então direto para o rastreamento. Caso esse *template* não esteja definido devido ao início da execução ou foi descartado por alguma falha no rastreamento, segue então a detecção do olho.

b) Detecção automática do olho e extração do *template*

A captura do olho é feita automaticamente. Essa etapa utiliza o classificador em cascata para os seguintes frames, incluindo o atual, até que o olho seja detectado. Uma vez que a detecção ocorreu, a subimagem que contém o olho é capturada e extraída para ser usada na etapa seguinte.

Observe no diagrama (Figura 14) que se o olho já detectado antes, e, portanto, estiver sendo rastreado, essa etapa pode ser ignorada.

c) Rastreamento do olho

A subimagem extraída da etapa anterior é usada no próximo *frame* para buscar outras regiões que se assemelhem a ela, através do *Template Matching*. Uma vez encontrada a subárea mais similar ao *template*, sua localização será armazenada para ser utilizada futuramente. Esse processo será repetido até o final do fluxo do vídeo, ou até que não seja possível encontrar uma subárea com grau de similaridade satisfatório.

d) Detecção do clique

Essa etapa é responsável por detectar o momento em que houve um clique. O sinal identificador de um clique é o piscar, ou o fechar, do olho sendo rastreado. Para identificar essa operação é utilizada a transformada de Hough para detecção de círculos. O círculo que deve ser detectado é a íris do olho. Caso o círculo da íris não seja detectado por sete *frames* sucessivos, caracteriza-se um fechar, piscar, do olho. Caso o círculo seja detectado por dois *frames* sucessivos é então considerado que o olho está aberto. A operação de clique é executada através da aplicação de funções do sistema operacional.

e) Controle do cursor

O *Template Matching* será utilizado para o rastreamento do olho no decorrer do vídeo, como já falado. Através da movimentação da íris em relação aos *frames* anterior e atual, é possível transpor o seu comportamento ao cursor do *mouse*.

As coordenadas do cursor deverão ser proporcionais às coordenadas do objeto de interesse. Dessa forma, o posicionamento sugerido do cursor será feito aplicando-se uma proporção aos valores obtidos pela movimentação do olho.

A geração de posicionamento do cursor, descrita anteriormente possui alguns problemas, que são gerados por ruídos e imprecisão que se prologaram desde a etapa inicial. Esses problemas fazem com que o posicionamento do cursor seja instável no decorrer do processo. Para corrigir esse mau posicionamento, será utilizado o Filtro de Kalman.

A seguir, será mostrado mais detalhadamente cada uma das etapas da metodologia, destacando detalhes mais técnicos da implementação.

3.1 Pré-processamento

Para controlar o mouse com o movimento do olho, é necessário capturar a imagem do usuário através da webcam, transformar esta imagem para a escala de cinza, espelhar no sentido horizontal, e então é feita uma melhoria através da equalização de histograma.

O pré-processamento do vídeo é uma etapa crucial para o funcionamento da metodologia como um todo. Primeiramente, porque todas as etapas seguintes dependem dos *frames* extraídos do vídeo. Além de que, a qualidade desses quadros afetam diretamente as técnicas de processamento de imagens utilizadas.

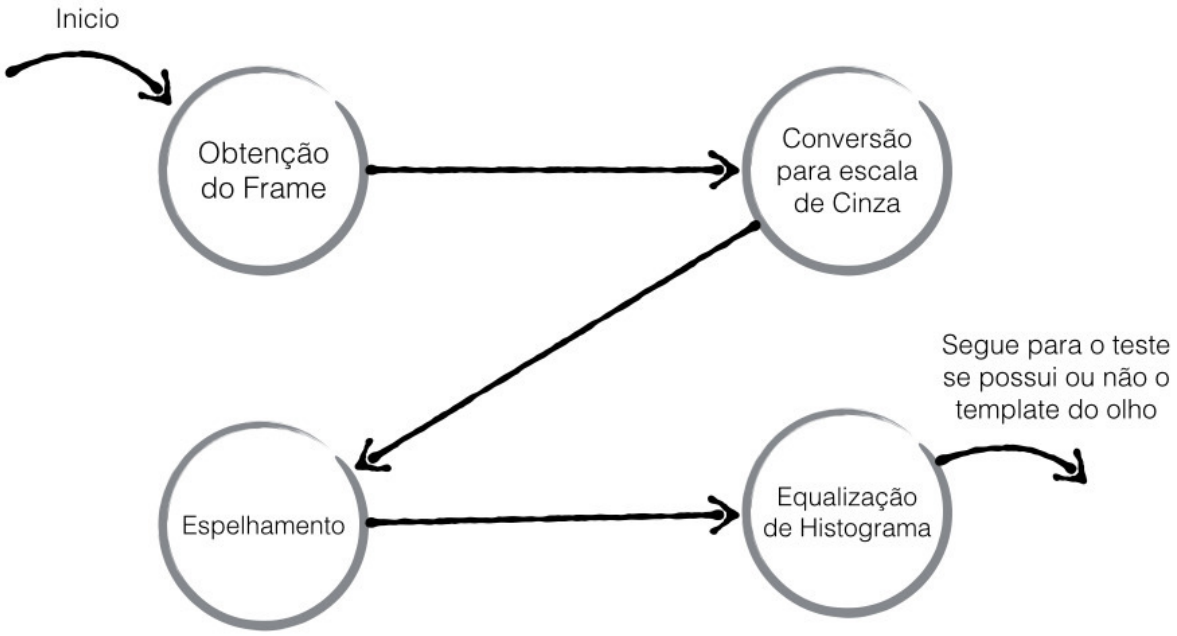
Há diversos fatores na aquisição do vídeo que podem influenciar em etapas futuras, entre eles a resolução de cada *frame*. Para o *Template Matching*, por exemplo, quanto maior a resolução da imagem, maior o tempo necessário para fazer uma busca na imagem toda, resultando em um maior consumo de tempo computacional. Por outro lado, quanto maior o *template*, maior também o número de informações a serem comparadas, aumentando assim o detalhamento. Logo, é possível ter-se um aumento da precisão.

Imagens de alta resolução, por si só, não são o suficiente para obter-se melhores resultados. Outros fatores também afetam diretamente as técnicas aqui empregadas, tais como a presença de ruído e iluminação instável. Além desses, o *Template Matching* é também afetado em situações em que a vizinhança do objeto a ser encontrado contém regiões similares ao próprio objeto.

Outro fator a ser destacado é que a metodologia computacional está fortemente voltada para computadores convencionais. Por isso, os *frames* capturados serão, quase sempre, obtidos a partir de webcams. Isso significa que, na prática, trabalharemos com *frames* de baixa resolução e, provavelmente, de baixa qualidade.

O pré-processamento realizado durante a aquisição, conforme o diagrama da Figura 15, é o espelhamento horizontal, e a equalização do histograma do *frame* em escala de cinza. O espelhamento é realizado pois é interessante que a movimentação da imagem esteja na mesma direção da movimentação do usuário. Dando ao usuário uma orientação confortável, equivalente a um espelho convencional, de forma que o cálculo do posicionamento cursor do mouse seja simplificado, seguindo a mesma direção do movimento do usuário. Caso este espelhamento não fosse realizado, o cursor do mouse seria movido em sentido oposto ao desejado. A equalização da imagem é realizada a fim dar uma melhoraria nas possíveis imagens de baixo contraste.

Figura 15 – Diagrama demostrando as etapas do pré-processamento



3.2 Captura do olho

Para realizar a detecção de olho, inicialmente é realizada a detecção de face sobre o *frame*. Caso uma face não seja detectada, o mesmo procedimento é repetido para os *frames* seguintes até obter-se um resultado. Somente após uma detecção bem sucedida, é extraída a região da face e nela é aplicada a etapa seguinte referente à detecção de olho. Caso não seja possível detectar um olho então a execução retorna para a detecção de face. Ambas as etapas são aplicadas a partir do método de detecção de objetos do classificador em cascata baseado nas Haar-*Like Features* apresentado na Seção 2.2. A biblioteca OpenCV dá suporte para ambas detecções.

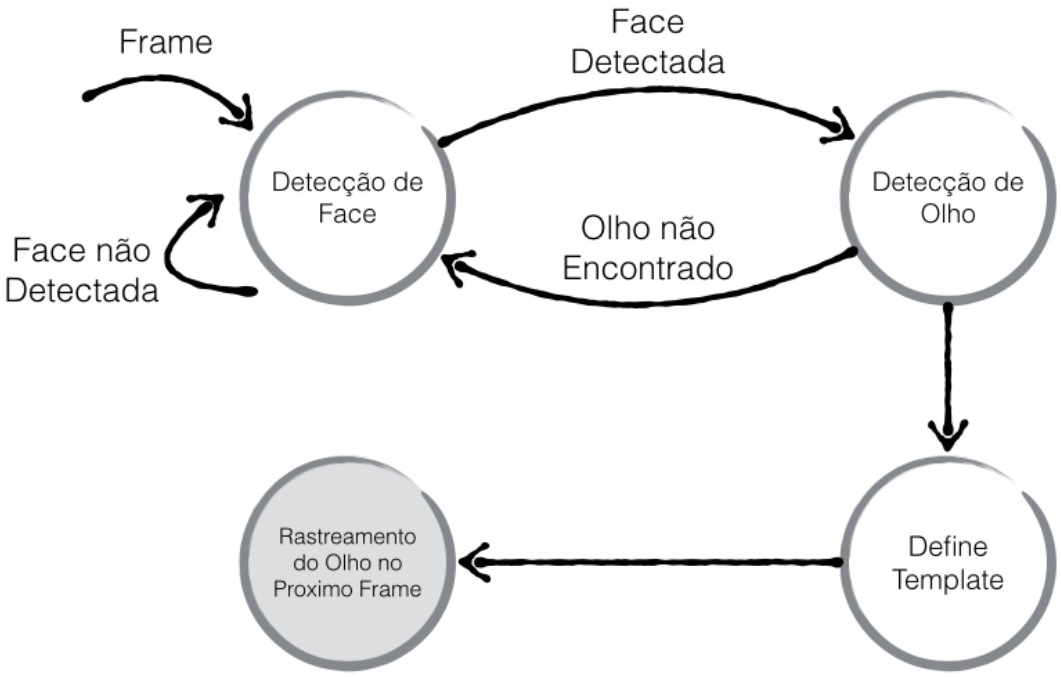
Uma vez concretizado a detecção, uma *bounding box* do olho é obtida. Dessa *bounding box* reduzida será extraída uma sub-imagem que será utilizada como *template* para o *matching* na próxima etapa.

Alguns pontos devem ser levados em consideração nessa etapa:

- Uma vez detectado o olho, a metodologia considerará que esse será sempre o objeto desejado. Por essa razão, é importante selecionar o olho com precisão para que as etapas seguintes possam ser aplicadas da forma esperada.
- Essa etapa é desnecessária caso o olho esteja sendo rastreado. Como descrito anteriormente, essa etapa busca extrair o *template* a ser utilizado.

O diagrama da Figura 16 mostra o funcionamento dessa etapa:

Figura 16 – Diagrama da detecção do olho



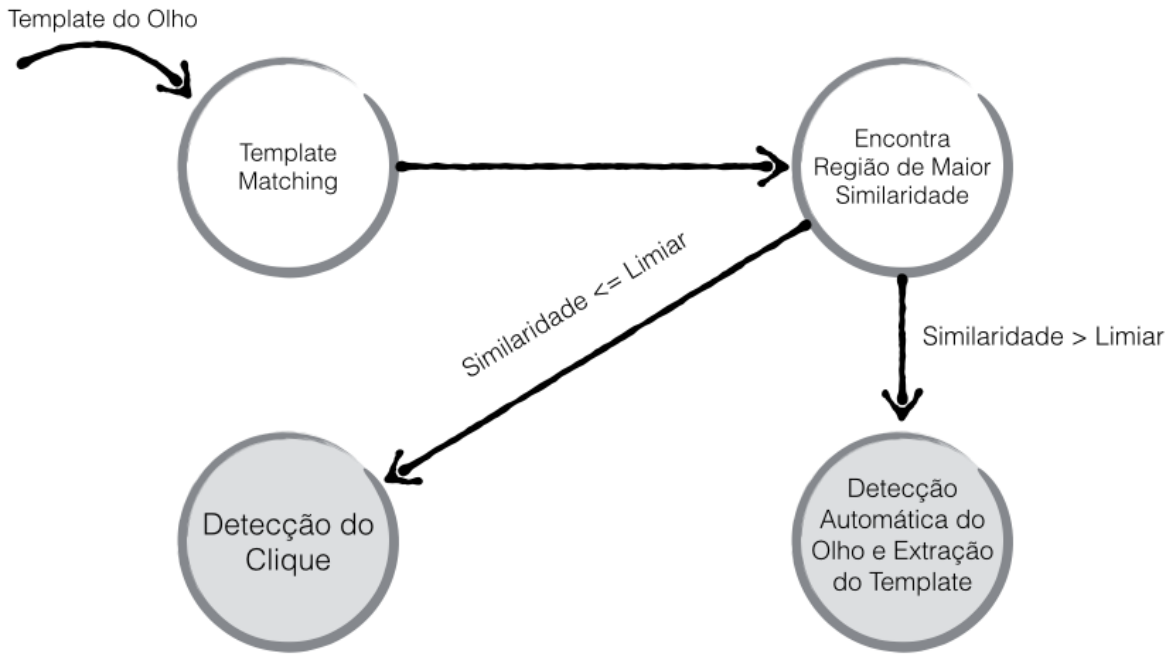
3.3 Rastreamento do olho

Como explicado na etapa anterior, a detecção do olho é feita automaticamente. A sub-imagem extraída na etapa anterior será utilizada para encontrar regiões similares nos quadros seguinte e eleger a região mais similar ao ao olho. Para essa comparação utiliza-se a técnica do *Template Matching*. Nesse caso, o olho encontrado na seção anterior se tornará o *template*.

A cada *frame,* é encontrada uma região que melhor se assemelha ao olho, baseado no *template* extraído na etapa anterior. A região resultante é indicada na tela do vídeo por um quadrado verde. Essa operação se repetirá sucessivas vezes até que ocorra uma interrupção, que será melhor explicada em seguida.

A Figura 7 representa o funcionamento do rastreamento olho por meio do *Template Matching*.

Figura 17 – Diagrama do rastreamento do olho



Neste caso, optou-se por utilizar a métrica de similaridade (Seção 2.3.2) baseada na combinação do quadrado da diferença normalizada, que possui um bom custo-benefício computacional e reduz os efeitos das diferenças de iluminação, retornando um valor de similaridade entre 0 e 1. Quanto mais próximo de zero, maior é a similaridade da região em questão.

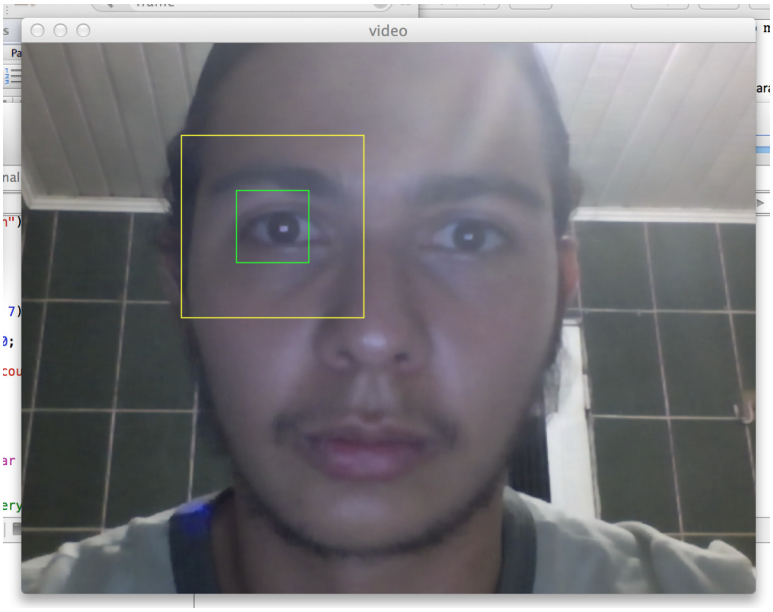
3.3.1 Restrição de Região de Busca – estratégia para melhoria de desempenho

Como dito anteriormente, o *Template Matching* é um algoritmo que pode ser custoso. No entanto, a metodologia proposta deve ser executada em tempo real. Certamente, essas questões precisam ser levadas em consideração, a fim de evitar degradação de funcionamento. Para solucionar esse impasse, foi utilizado uma delimitação na área de varredura do *Template Matching*.

Sempre que é extraído um *template*, é criada uma região de dimensões maiores e concêntrica com a área do *template*. Essa região será limitadora na busca do *template* no próximo *frame*. Pode-se entender melhor esse recurso vendo a Figura 18, onde o

quadrilátero menor, em verde, é a região do olho rastreado no *frame atual*. O quadrilátero maior, em amarelo, representa a região de busca do *Template Matching* no próximo *frame*.

Figura 18 – Delimitação da região de busca do Template Matching



Junto à detecção inicial do olho, também é delimitada essa região de busca, com base na posição do *template.* No próximo *frame*, essa região será percorrida pelo *template*.

O uso dessa região também pode trazer efeitos indesejados. Se a região for muito pequena, e ocorrer um movimento brusco, pode ocorrer de o objeto em questão não se encontrar mais nos limites dessa região. Quanto menor a região, maiores são as chances disto acontecer, pois menor será a flexibilidade para o deslocamento do objeto entre *frames* dentro dessa área em questão. Portanto, quanto menor a região de busca, mais fácil é para o objeto se perder durante o rastreamento, e quanto maior a região, maior também o tempo, e aumenta as chances de se encontrar o resultado falso devido a vizinhanças com alta similaridade.

3.4 Detecção de clique

Essa etapa tem o papel de detectar se houve a ocorrência de um clique ou não, relacionando o clique a um piscar do olho. Logo, se o olho estiver fechado, o estado do cursor estaria associado à situação em que o botão esquerdo do mouse estaria pressionado

e, se o olho estiver aberto, o cursor estaria em seu estado normal, ou seja, equivaleria ao botão liberado.

Para determinar se o olho está aberto ou fechado, é necessária uma análise sobre a região de maior similaridade obtida previamente na etapa do rastreamento do olho. Essa análise tem como base a detecção do círculo referente a íris do olho. Se esse círculo for identificado, o olho é considerado aberto, se não, é considerado fechado. A detecção círculo é feita a partir da transformada de Hough (Seção 2.4) aplicada sobre o novo *template*, que foi extraído pelo rastreamento, descrito anteriormente. Destaca-se aqui que:

1. É utilizado novamente a equalização de histograma (Seção 2.1.2), porém levando em consideração apenas a área referente ao novo *template*. Isto é feito afim de realçar o contraste da íris do olho (e consequentemente suas bordas), uma vez que, com relação aos número de pixels dessa região extraída, a íris cobre uma parte muito significativa dessa área.
2. A equalização de histograma mencionada anteriormente é implementada de tal forma que afete apenas a transformada de Hough, e não o *template* utilizado para o rastreamento no próximo *frame*.

A utilização da transformada de Hough é, de certa forma, complexa, requerendo uma certa atenção, pois a mesma é altamente parametrizada, necessitando de parâmetros ideais para detectar o círculo referente a íris. Os resultados obtidos, dependendo dos parâmetros utilizados, apresentam uma quantidade considerável de falsos círculos na imagem, isso ocorre devido ao fato de que o filtro de Canny, utilizado internamente na transformada de Hough, é muito sensível a ruídos presente na imagem. Uma forma de se contornar isso é ajustar adequadamente o limiar referente ao Canny, além de outros parâmetros que reduzem o número de falsos círculos, tais como tamanho mínimo do raio, distância mínima entre o centro dos círculos, dentre outros. Porém, fixar um valor específico é uma tarefa que pode facilmente comprometer a detecção pois os parâmetros ideais podem variar devido à condições adversas na imagem, como resolução, iluminação, contraste, e ruídos.

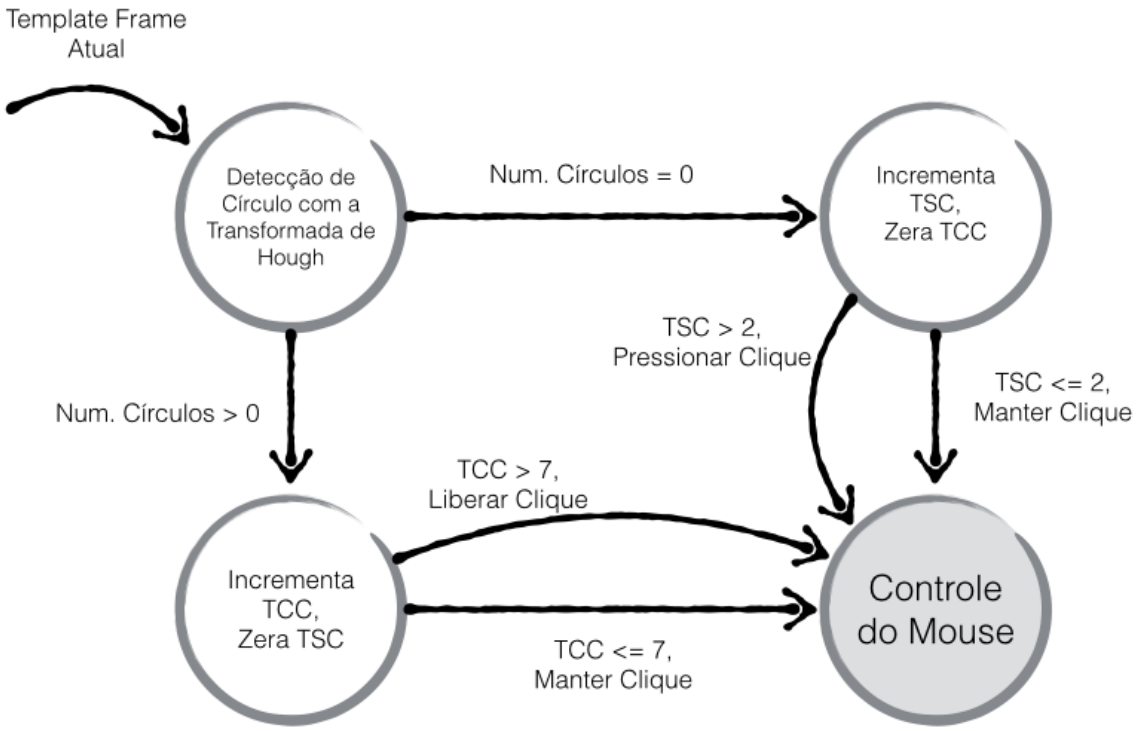
A parametrização foi escolhida empiricamente e será descrita nos testes e resultados (Capítulo 4). Uma alternativa seria utilizar parâmetros que tornem a detecção o menos

rigorosa possível, não definindo limites para as distâncias o raio mínimo e máximo. Valor do limiar do filtro de Canny também foi definido para o menor possível, no caso 1. A parametrização, por si só, ainda não é suficiente para a metodologia, pois ainda há muitos casos em que o olho está aberto, porém nenhum círculo é encontrado, devido a algum erro no rastreamento, e também há casos em que o olho está fechado e, ainda assim, círculos são detectados, devido a ruídos presentes na região do olho, ou a presença de formas que, de alguma maneira se assemelham a círculos.

Foi observado que as falhas na detecção de círculo causadas devido a ruídos na imagem seguem uma certa frequência. Sabendo isso, pode-se utilizar uma determinada frequência como valor de "tolerância" para estimar, a partir de uma sequência de *frames*, se o olho encontra-se aberto ou fechado. De forma que, se número de *frames* consecutivos com círculo (TCC) for maior que 7, é inferido, então, que o estado do mouse passou a ser o clique liberado. Entretanto, caso o número de *frames* consecutivos sem círculo (TSC) for maior que 2, é interpretado que o estado do mouse passou a ser clique pressionado. em cada iteração, se houver pelo menos um círculo detectado, a variável TCC é incrementada, a TSC é zerada e, por fim, o teste é feito para definir o estado do cursor. Caso não haja círculos, a variável TSC é incrementada e TCC é zerada, prosseguindo então para o teste a fim de definir o estado. Caso nenhuma dessas condições for satisfeita, é mantido o estado do clique anterior, seja pressionado, seja liberado.

A Figura 19 sintetiza os passos descritos nessa etapa de detecção do clique.

Figura 19 – Diagrama da detecção do clique

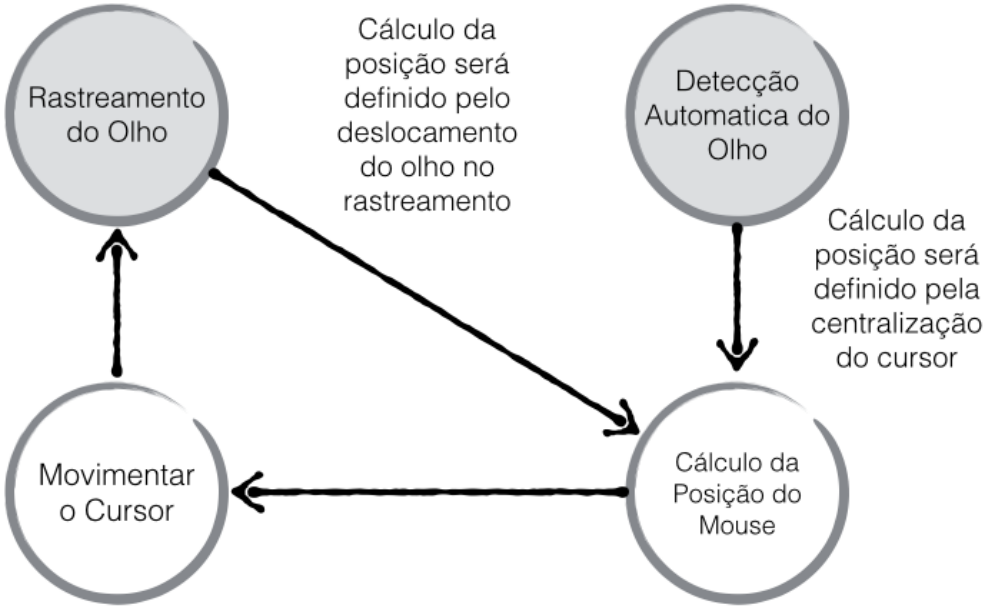


Após inferir o estado do clique, é executada então a próxima etapa, que consiste no posicionamento do cursor do mouse baseado na posição encontrada pelo rastreamento do olho.

3.5 Posicionamento básico do cursor

Essa etapa tem o objetivo de calcular as coordenadas onde o cursor deve se posicionar na tela com base na posição do olho adquirida na etapa do rastreamento. O funcionamento dessa etapa é representado pelo fluxograma da Figura 20.

Figura 20 - Diagrama do posicionamento básico do cursor



O posicionamento é feito através de um simples cálculo de proporção. Ou seja, sendo proporcional ao deslocamento entre a localização do olho extraído no *frame* anterior e a sub-imagem encontrada pelo *Template Matching* durante o rastreamento no *frame* atual. É definida então uma constante de proporção, que relacionará o deslocamento do olho no intervalo entre *frames* com o deslocamento do mouse. Essa constante poderia ser obtida através da relação entre as dimensões do *template* e as dimensões da tela. Porém optou-se por definir essa constante de forma que não degradasse a sensibilidade da movimentação do cursor. Empiricamente foi constatado que valores na faixa entre 15 e 20 apresentam uma sensibilidade satisfatória para essa movimentação.

Uma vez que a posição anterior é fundamental para calcular o deslocamento, é necessário, dessa forma, definir uma posição inicial para o cursor do mouse. Dessa forma, sempre que a houver a execução da detecção do olho e extração do *template*, a posição do o cursor do mouse será definida como sendo o centro da tela.

Com a posição do mouse inicializada, é assegurado que sempre existirá um estado anterior, sendo possível o cálculo da posição. O posicionamento do mouse pode ser representado pelas Equações 35 e 36, (DINIZ, 2012).

$$C_x^i = \begin{cases} W_w/2, se\ i = 0; \\ C_x^{i-1} + \left(O_x^i - O_x^{i-1}\right) \cdot K, se\ i > 0 \end{cases} \quad (35)$$

$$C_y^i = \begin{cases} W_h/2, se\ i = 0; \\ C_y^{i-1} + \left(O_y^i - O_y^{i-1}\right) \cdot K, se\ i > 0 \end{cases} \quad (36)$$

A Equação 35 calcula a posição do cursor no eixo x, representada pela variável C_x^i, e a Equação 36 calcula a posição no eixo y, representada pela variável C_y^i, a partir da localização do olho representada pelas coordenadas O_x^i e O_y^i, utilizando como referência também a posição anterior (O_x^{i-1} e O_y^{i-1}). Onde i equivale à iteração atual contabilizada a partir da detecção automática do olho. W_w e W_h correspondem às dimensões da tela do monitor, largura e altura, respectivamente. K é a constante de proporção mencionada anteriormente, sendo necessária para que as movimentações "milimétricas" do olho tenham uma resposta mais perceptível no cursor do mouse.

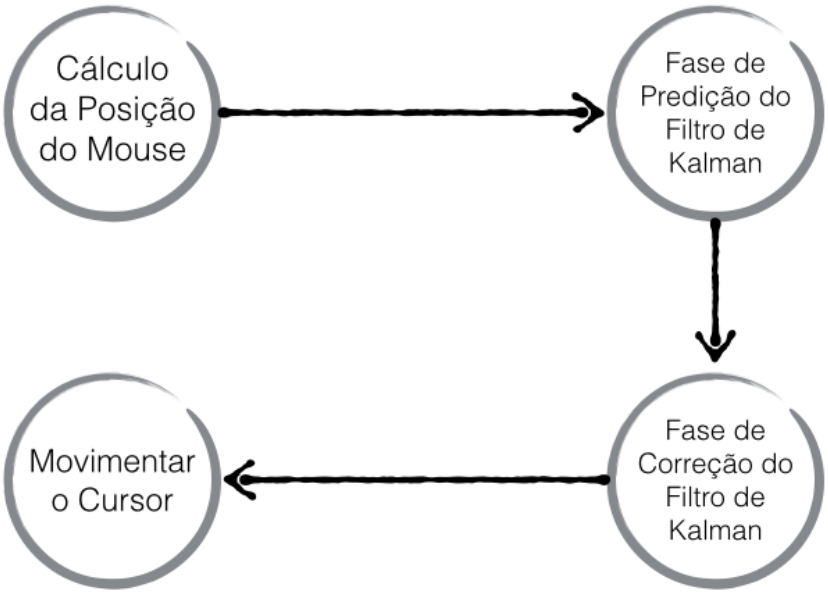
O objetivo do uso de uma constante K é tornar a movimentação do cursor proporcionalmente maior do que movimento da íris no decorrer do vídeo. No entanto, ocasiona uma imprecisão considerável na movimentação do cursor, principalmente se seu valor for alto. O tratamento dessa imprecisão será discutido posteriormente.

Além disso, esse cálculo é afetado pela movimentação da face, uma vez que quando ela se movimenta, o olho do usuário muda de posição.

3.5.1 Filtro de Kalman – Otimização de posicionamento do cursor

O posicionamento do cursor baseado no deslocamento do olho é multiplicado por uma constante K (Equações 35 e 36), faz com que a movimentação do cursor reproduza movimentos que, muitas das vezes, são imprecisos e trêmulos, pois derivam de erros que na etapa de rastreamento são visivelmente desprezíveis, mas que nessa etapa são significativos. Dessa forma, é necessária uma otimização do movimento do cursor para corrigir problemas de posicionamento esperado. Os passos podem ser representados pela Figura 21.

Figura 21 – Diagrama de otimização do posicionamento do cursor



Primeiramente, o rastreamento do olho é impreciso devido à própria natureza das imagens de vídeo que, em geral, apresentam ruídos e intervalos nas capturas dos *frames*, reduzindo precisão do *Template Matching*.

Em segundo lugar, o deslocamento do cursor deve ser proporcionalmente maior do que o do objeto de interesse. A íris do olho, por exemplo, move-se "milimetricamente". Isso implica em um erro de posicionamento do cursor significantemente maior do que o erro no rastreamento do objeto de interesse.

Devido a esses fatores, o Filtro de Kalman (Seção 2.5) é utilizado como alternativa para o controle e redução dessas instabilidades. Através da sua aplicação, pressupõe-se que haverá uma melhoria significativa no processo de transposição dos movimentos do olho para o cursor do *mouse*.

Para aplicação do Filtro de Kalman é necessário modelar o sistema. Inicialmente, define-se o estado do cursor do mouse. Este pode ser representado por quatro dimensões. Dentre essas dimensões C_x e C_y referem-se, respectivamente, a posição do cursor nos eixos x e y, assim como C_{v_x} e C_{v_y} indicam, respectivamente, à velocidade do cursor nos eixos x e y. Possuindo, a variável de medição, duas dimensões, C_x e C_y.

Para que se possa definir os x_k das próximas iterações, ou seja, as estimações para as próximas posições nos próximos *frames*, é preciso determinar primeiro se há entradas de controle ou eventos aleatórios. Neste caso, não há nenhum controle externo. Existiria um controle externo caso fosse criada alguma forma de ajuste da sensibilidade de cursor, como é implementado nos sistemas operacionais, mas esse não é o caso.

Resta, portanto, apenas definir a matriz de transição F. Ela será uma matriz quadrada $4x4$. Será quadrada pois a mesma será multiplicada pelo estado anterior, configurando parte do próximo estado, e será $4x4$ pois o estado modelado possui 4 dimensões.

Sabendo que a modelagem do estado é dada por:

$$x_k = \begin{bmatrix} C_x \\ C_y \\ C_{v_x} \\ C_{v_y} \end{bmatrix} \tag{37}$$

E que, considerando que a diferenciação no tempo dt entre *frames* é de uma unidade de tempo, a matriz é representada por:

$$F = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{38}$$

Por fim, é necessário definir z_k. No plano da tela, a única observação realizada diretamente é a próxima posição do objeto. Nesse sentido, pode-se dizer que z_k conterá apenas as posições e, portanto, será representado por uma matriz $2x1$ que contém as próximas coordenadas calculadas para a localização cursor. Assim, tem-se:

$$z_k = \begin{bmatrix} C_x \\ C_y \end{bmatrix}_k \tag{39}$$

A implementação do Filtro de Kalman utilizada nesse trabalho foi feita através da biblioteca OpenCV. Os parâmetros de ruído são um tanto complicados de serem

determinados, já que não é uma tarefa simples prever o ruído presente ambiente a ser realizado as medições. Esses valores podem ser inferidos empiricamente. Porém, foi inicialmente utilizado os mesmo valores apresentados por Bradski & Kaehler (2008) em exemplos similares e que apresentaram resultados satisfatórios. Sabendo disso, as matrizes referentes a ruído de processo e ruído de medição foram inicializadas com, respectivamente, os valores 10^{-5}e 10^{-1} em suas diagonais. Quanto melhor estimados esses valores, melhor a estimação obtida pelo filtro de Kalman.

Os parâmetros de modelagem deduzidos até agora são suficientes para que o algoritmo seja inicializado e tenha funcionamento satisfatório.

Em nível de código, o funcionamento do Filtro de Kalman terá duas etapas: fase de predição e a fase de correção. A primeira tenta estimar, a partir do estado anterior, o estado atual. Já a segunda fase faz uma correção do estado atual a partir da leitura realizada, que requer como parâmetro a coordenada atual onde o cursor deverá estar, baseado no cálculo do posicionamento de cursor.

A melhoria do Filtro de Kalman é adicionada após a fase do cálculo da posição do cursor e antes da movimentação do mesmo. Utilizando da fase de predição para otimizar a posição do cursor, e o novo valor obtido é então utilizado para mover o cursor.

4 TESTES E RESULTADOS

Durante desenvolvimento desse trabalho foi implementado um protótipo (nos sistemas operacionais Windows e MAC OS X) a fim de testar a metodologia aqui proposta, fazendo assim uma análise da mesma, sugerindo especificações e ajustes onde necessário. Foram utilizados uma webcam de 2.0 megapixels no sistema Windows, como também a FaceTime HD Camera 720p no sistema OS X. Alguns vídeos foram realizados mostrando a utilização desse protótipo, como mostra (REIS, 2014)[8]. A seguir serão mostrados os testes realizados em cada etapa, individualmente.

4.1 Resultado do Pré-processamento

Conforme dito na Seção 3.1, o pré-processamento foi composto por técnicas triviais de processamento de imagem. A Figura 22 exibe (a) um frame extraído do vídeo, onde (b) o frame convertido para a escala de cinza, que após ser espelhado (c) é utilizado a equalização de histograma (Seção 2.1.2) para obter-se (d).

Figure 22 – Resultado do Pré-processamento

[8] Resultado publicado no YouTube. Disponível em: <http://www.youtube.com/watch?v=KPM7CPMHHyo>. Acesso em: 10 de Janeiro de 2014

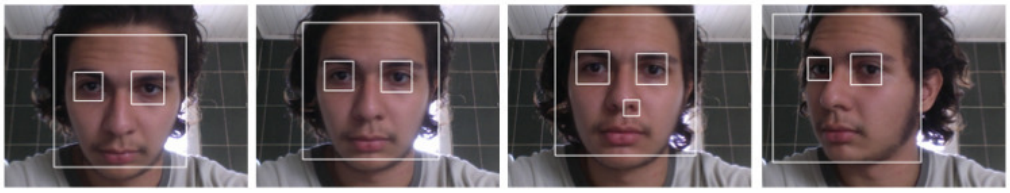
4.2 Resultado da Captura do Olho

A Sessão 3.2, descreve a captura do olho realizada que utiliza de um classificador em cascata que utiliza as características Haar-*like*. A partir daí é definido como olho uma janela de 64x64 com o centro no local indicado pelo classificador. Foi utilizado funções do OpenCV referentes a esse classificador, assim como os arquivos de treinamento fornecidos por ela: “haarcascade_frontalface_alt2.xml” e “haarcascade_eye.xml".

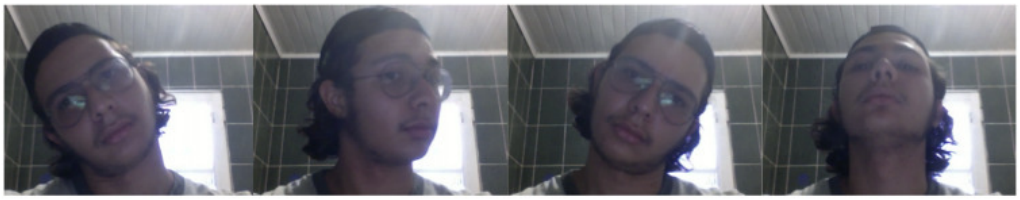
A Figura 23 mostra os resultados dessa etapa, onde a detecção de olho foi precedida da detecção de face. Observamos na Figura 23 (a) que essa técnica apresenta alguns falsos positivos (resultados errados classificados como sendo verdadeiros) na detecção dos olhos. Esse método retorna seus resultados em uma lista. Como na metodologia proposta é utilizado apenas um olho dos olhos encontrados para fazer o rastreamento, apenas o primeiro é utilizado e os demais elementos presentes no resultado da detecção dos olhos são descartados. Na maioria dos casos, os falsos positivos não afetam as próximas etapas, pois geralmente são descartados por não serem o primeiro elemento dessa lista.

A Figura 23 (b), mostra os casos em que houve uma falha na detecção de faces, consequentemente a detecção de olhos não foi realizada.

Figure 23 – Resultado da captura dos olho precedida da detecção de face



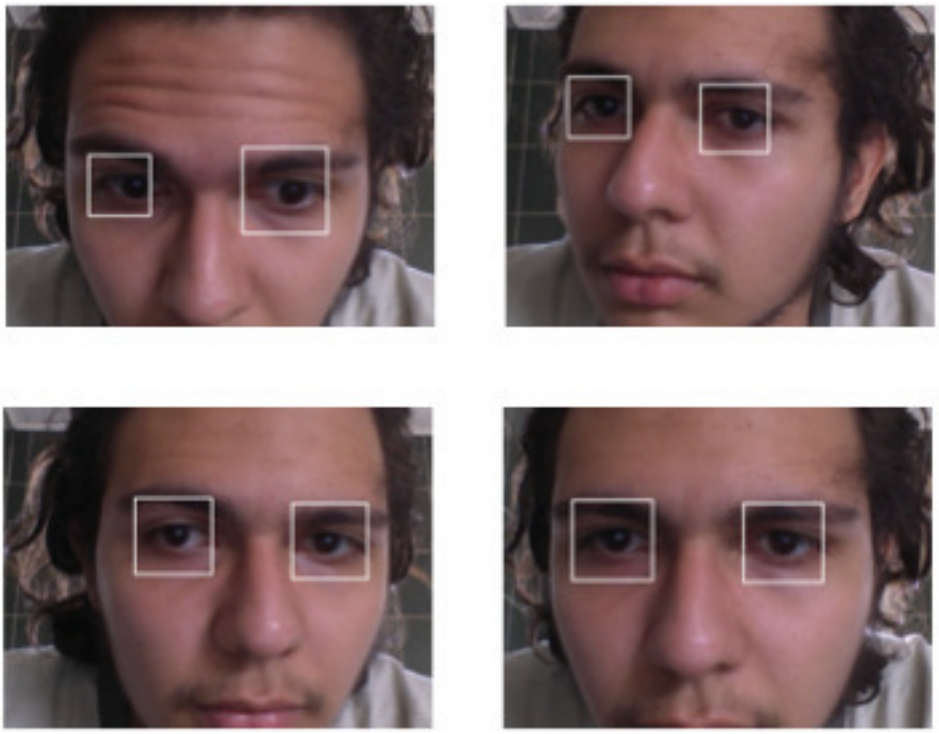
(a) Casos em que a detecção dos olhos teve sucesso



(b) Casos em que a detecção dos olhos falhou

A fim de se obter imagens do olho com melhor resolução e, consequentemente, uma maior precisão, foram realizados testes onde o rosto do usuário estava bem próximo da tela do computador. Contudo, em casos onde o usuário está muito próximo à tela, fazendo com que na imagem captura se tenha apenas parte do rosto, a detecção dos olhos é afetada por depender diretamente da detecção facial, que nesse caso não seria possível. Pela finalidade da metodologia, pressupõe-se que o rosto usuário está a frente da tela. Por isso também foram realizados testes com a detecção de olho sem ser precedida da detecção facial, conforme a Figura 24 .

Figure 24 - Detecção dos olhos



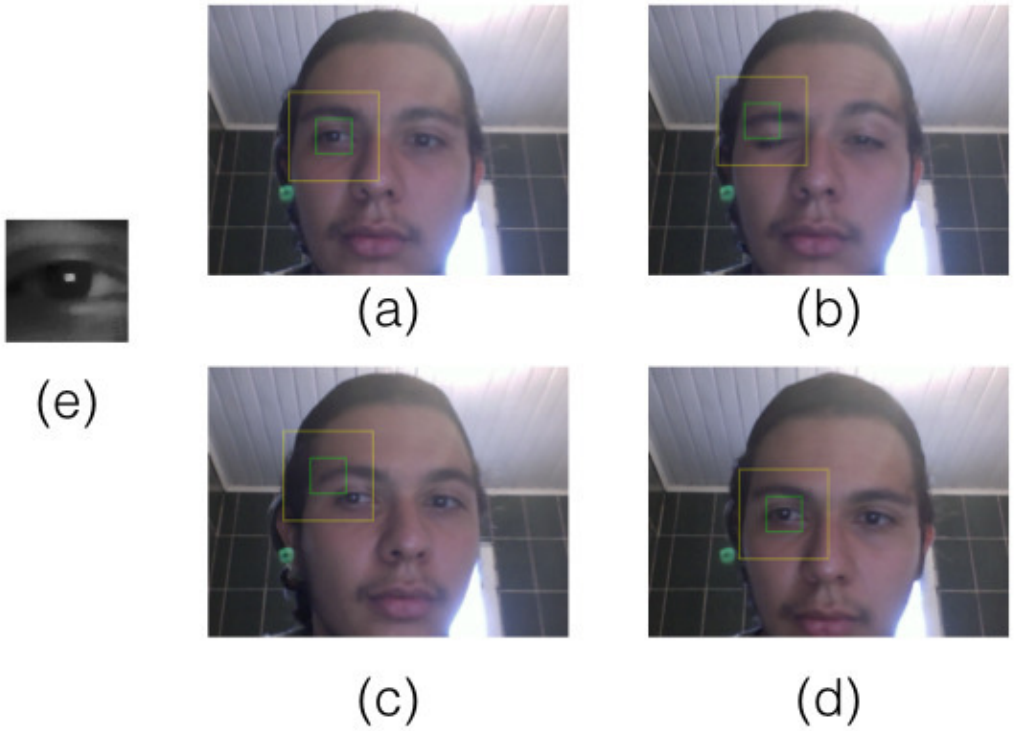
4.3 Resultado do Rastreamento do Olho

Como proposto na Seção 3.3, o rastreamento do olho se deu através do *Tempalte Matching* e da limitação da região de busca. Sendo o *template* utilizado definido na etapa anterior de detecção do olho. A Figura 25 contém os resultados obtidos, em que cada

dimensão da região de busca utilizada nesse caso é de 2,5 vezes maior do que a respectiva dimensão no olho encontrado na detecção.

Comparando as Figuras 25 (a) e 25 (b) é possível observar que quando ocorre o piscar do olho, por ter-se um *template* estático e uma repentina mudança no olho, ocorre um indesejado *matching* (casamento) com uma região da área vizinha similar ao *template*. Em alguns casos, após a abertura do olho o rastreamento pode se "perder" mais nessa região, por ter um valor ainda aceitável de similaridade, e a região de busca não abranger o olho por completo, como é o caso de (c); porém, se a região de busca passar a abranger novamente o olho, o rastreamento deve encontrar novamente o olho, como em (d). Para a finalidade da metodologia, não é viável que o olho se perca. O limiar de similaridade aceitável estabelecido foi de 0,25. E quando não satisfeito esse limite, a etapa de detecção do olho será novamente executada.

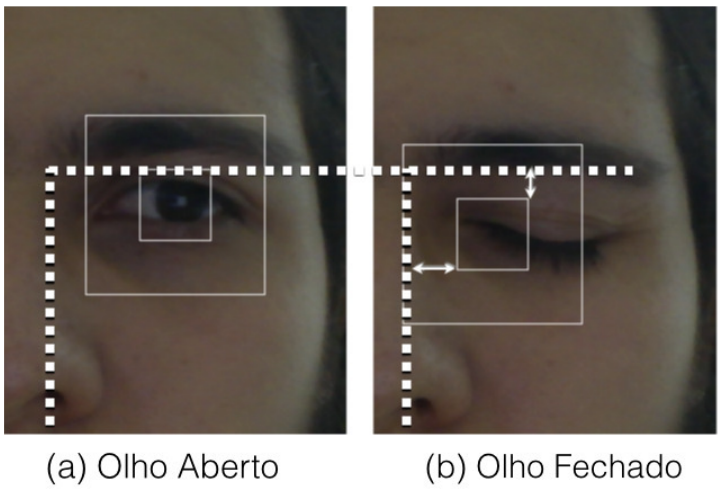
Figure 25 – Resultado do Rastreamento usando *Template Matching*



Em alguns casos notou-se também movimentos indesejados ocasionados pela piscada involuntária do olho, como é o caso da Figura 26. O *template* gerado na detecção do olho, idealmente, é uma imagem do olho, onde a íris está no centro. Considerando os frames

anteriores e posteriores ao pisco, o centro da parte visível do olho não coincidirá mais com o centro do olho propriamente dito. Então, nesses frames, o melhor *matching* será provavelmente obtido centralizando o *template* sobre parte visível da íris, não sendo assim mais o centro da região ocular, essa diferença. Que é interpretada pela etapa seguinte de movimentação do mouse.

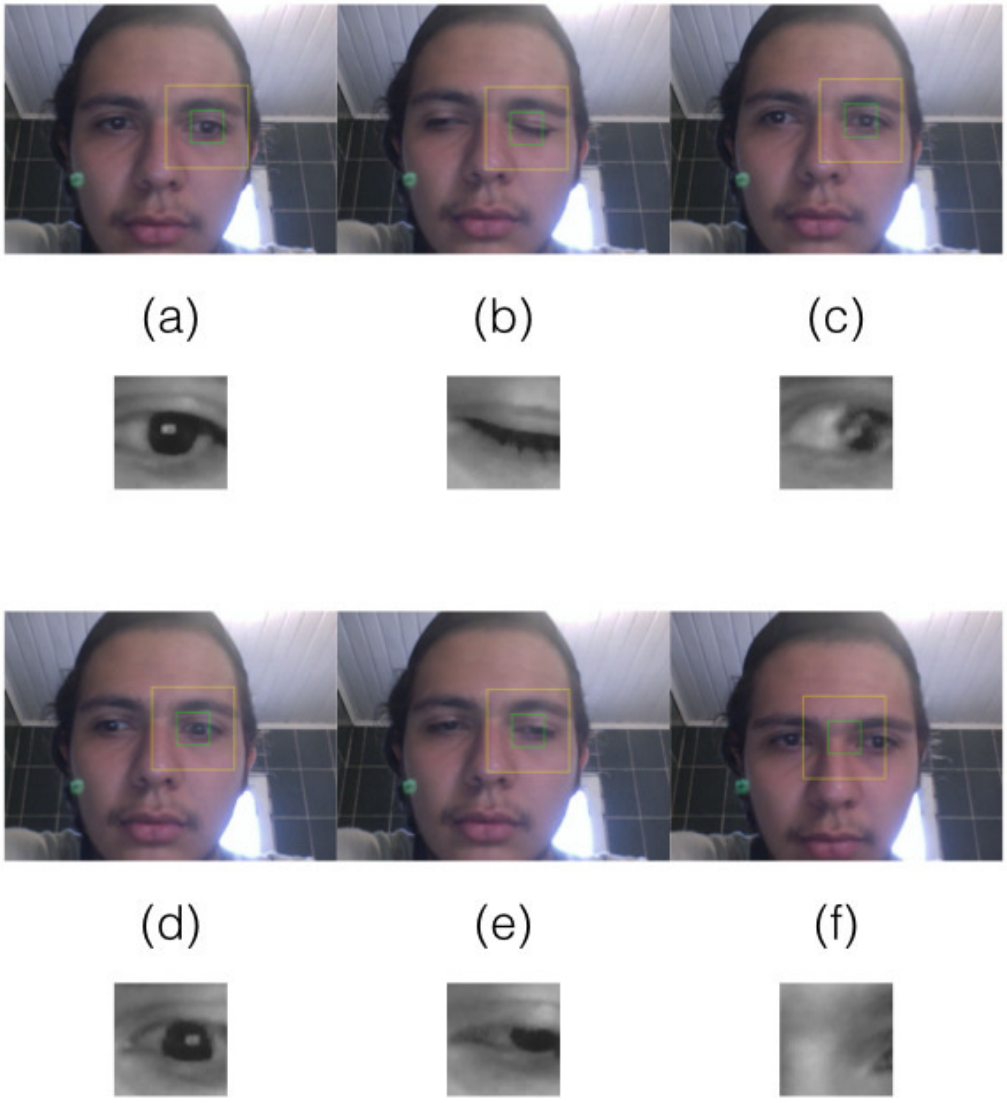
Figure 26 – Movimentação gerada pelo piscar do olho



Fonte: Elaborada pelo autor.

O Template Matching Adaptativo (Seção 2.3.1) também foi testado, e alguns dos resultados podem ser visualizados na Figura 27, que exibe alguns frames durante o rastreamento e a respectiva textura do *template* utilizada naquele momento. Pode-se observar que da Figura 27 (a) à Figura 27(e), apesar da mudança gerada repentinamente no objeto (no caso o olho) devido ao piscar, o rastreamento não foi tão afetado quanto no caso de manter-se o *template* estático. Todavia, foi observado que uma vez que o rastreamento se perde, Figura 27 (f), é improvável voltar ao objeto de interesse, nesse caso olho. Pois houve uma defasagem no *template*, uma vez que a textura do *template* muda para algo indesejado, como por exemplo a pele, passando aquela região a ser rastreada. Sendo pouco provável retornar ao olho. Entretanto o *Template Matching* Adaptativo apresentou uma melhor performance se tratando em modificações sofridas pelo objeto.

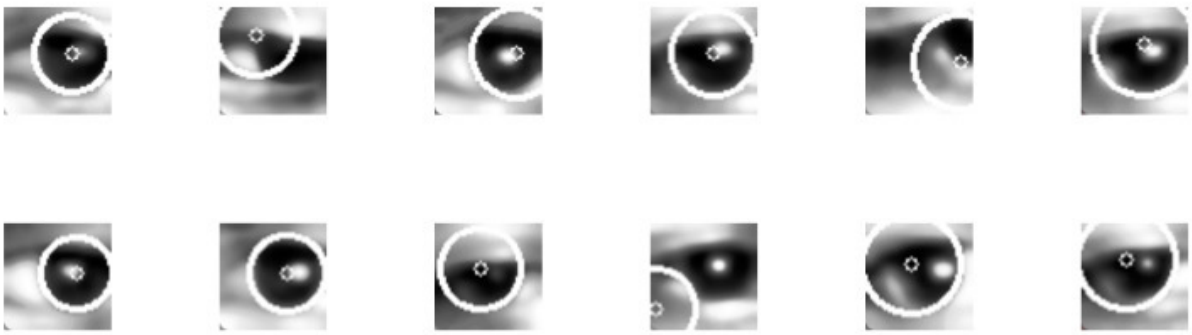
Figure 27 – Resultado do Rastreamento usando *Template Matching* Adaptativo



4.4 Resultado da detecção do clique

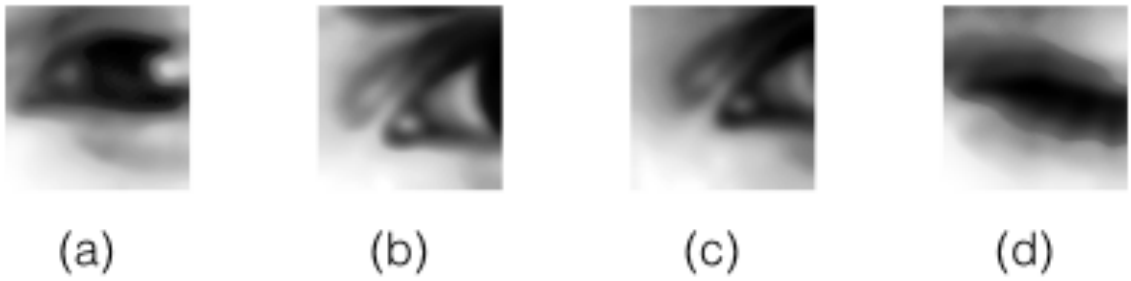
A detecção de clique (Seção 3.4) utiliza a função do OpenCV referente à Transformada de Hough (Seção 2.4) para detecção de círculos. A partir de diversos testes, foi obtido um resultado satisfatório, Figura 28, para a detecção da íris utilizando os parâmetros 64, 360, 16 e 24, sendo respectivamente: a distância mínima entre círculos (levando em consideração a posição do centro); o limiar para função interna de detecção de bordas de bordas utilizando filtro de Canny; o raio mínimo e o raio máximo.

Figure 28 – Detecção de íris usando transformada de Hough



Devido à parametrização, houveram falhas na detecção (falsos negativos) em que o círculo da íris não pode ser encontrado. Na Figura 29, por exemplo, (a) é um falso negativo. Em (b) e (c) não foi possível encontrar o *template* pois o mesmo foi defasado para o canto do olho. Apenas em (d) o olho encontra-se de fato fechado. Conforme descrito na Seção 3.4, se por sete *frames* consecutivos não for detectado um círculo é considerado então que houve um clique, e o mesmo é mantido pressionado até que haja pelo menos dois *frames* consecutivos que possuam um círculo.

Figure 29 - Falhas na detecção de íris



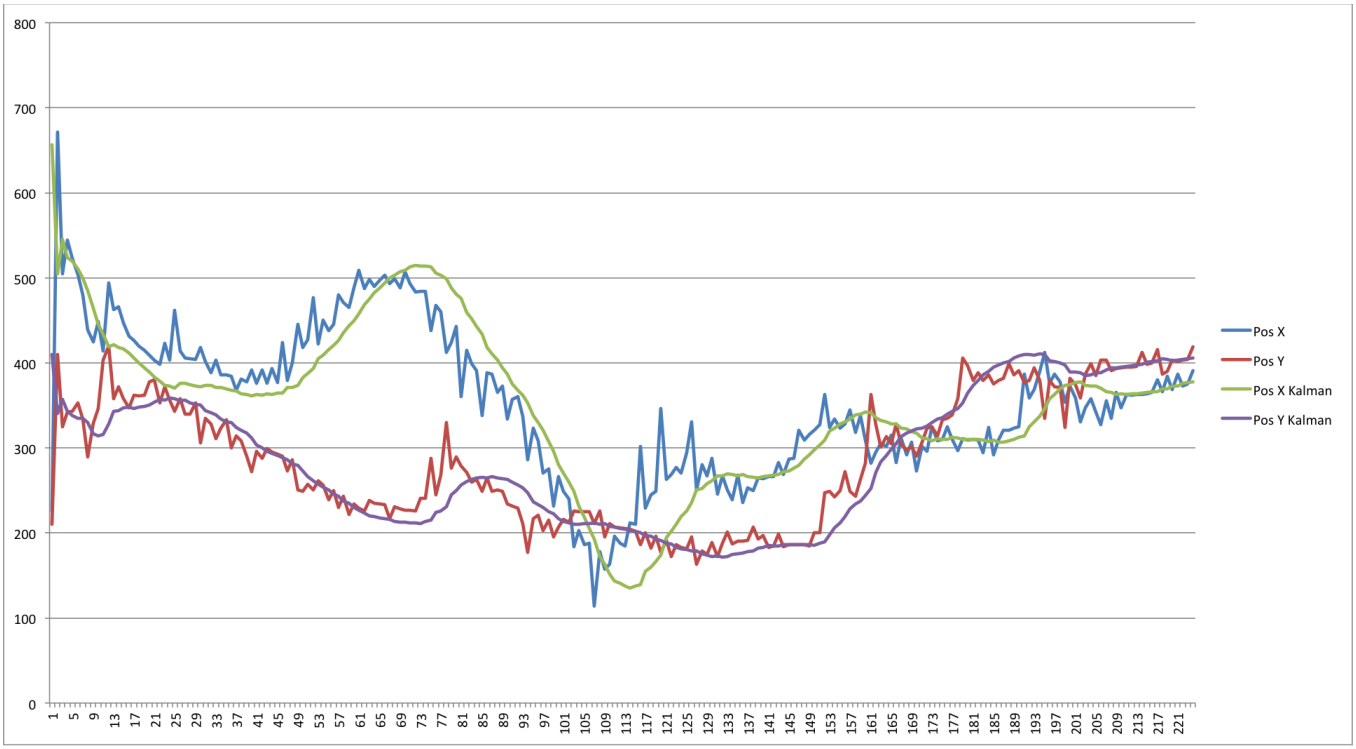
4.5 Resultado do posicionamento do mouse

Conforme a Seção 3.5, o posicionamento do mouse é realizado com base na localização do olho obtida a partir do rastreamento. Porém, quando não se tem uma câmera de alta resolução o rastreamento pode apresentar ruído, dado a sensação que a região marcada está “tremendo” durante o rastreamento. Em alguns casos essa imprecisão pode

ser mínima, mas quando transposto esse movimento para o cursor do mouse, tal imprecisão passa a ser perceptível, e até indesejável.

O Gráfico 1 apresenta os valores $Pos\ X$ e $Pos\ Y$, como sendo as coordenadas do cursor descritos respectivamente por C_x^i e C_y^i nas Equações 35 e 26. É evidente, observando os picos, a presença de ruídos nos valores obtidos. Após aplicar a melhoria do filtro de Kalman (Seção 3.5.1), obtem-se uma suavização na movimentação do cursor, gerando valores corrigidos para C_x^i e C_y^i, apresentados no Gráfico, respectivamente por $Pos\ X\ Kalman$ e $Pos\ Y\ Kalman$.

Gráfico 1 – Melhoria do posicionamento do cursor utilizando filtro de Kalman



5 CONCLUSÃO

Esse trabalho teve por objetivo desenvolver uma metodologia computacional para auxiliar pessoas portadoras de necessidades especiais no controle de movimentação do cursor do mouse através do rastreamento do olho em vídeo.

A metodologia computacional desenvolvida teve como base a quatro técnicas principais: Classificador em cascata, *Template Matching*, Transformada de Hough e Filtro de Kalman. Utilizadas respectivamente para:

1. A detecção automática dos olho.
2. O rastreamento do olho, sendo possível, como exposto nesse trabalho, inferir o posicionamento do cursor.
3. A detecção do pisco, determinando assim se houve ou não a intenção de acionar o clique do mouse, por parte do usuário.
4. A suavização da posição estimada pelo rastreamento. Promovendo assim ao usuário uma maior precisão na movimentação do cursor.

Como resultado, foi construída uma ferramenta que permite ao usuário movimentar o cursor com uma boa precisão sem o auxilio das mãos, realizando também cliques na tela através do piscar.

5.1 Avaliação do Trabalho

Esse projeto ainda encontra-se em estágio de conceptualização a respeito de como facilitar o uso de computadores por pessoas com necessidades especiais utilizando de recursos já presente na maioria dos computadores, tais como webcams. Porém, apesar desse trabalho ainda se encontrar em fase inicial, já apresenta uma boa contribuição nessa área. Os resultados obtidos mostram que a metodologia proposta é de fato capaz prover ao usuário um controle alternativo do cursor do mouse a partir do rastreamento do olho, sincronizando a movimentação do olho com o posicionamento do cursor; assim como simular o clique através do piscar do olho rastreado.

É notado também que a movimentação do cursor do mouse não depende somente da movimentação ocular, mas por ter como referencial posição do olho referente a origem da

imagem, movimentos de rotação e translação da cabeça também afetam e auxiliam no posicionamento do cursor.

Também foi observado que movimentos bruscos na imagem, ou modificações sofridas na própria região que se está rastreando podem ocasionar uma defasagem do *template*, prejudicando assim o rastreamento. O piscar do olho por exemplo, em alguns casos gera uma defasagem no *template*. A utilização do *Template Matching* Adaptativo trás uma melhoria considerável nesse aspecto.

A utilização do filtro de Kalman trás uma contribuição muito importante, pois com sua utilização a movimentação do cursor se torna suave, consequentemente dando mais precisão à movimentação.

A aplicação do Classificador em cascada para a detecção automática dos olhos, assim como a Transformada de Hough para detecção do clique, trouxe uma inovação a esse projeto, se comparado ao trabalho anterior realizado por Diniz (2012). Uma vez que o usuário portador de necessidade especial não necessitaria mais de um auxilio para informar manualmente a posição do cursor, como também possui mais um controle além da movimentação do cursor – o acionamento do clique direito do mouse por meio do piscar. Observou-se também que aumentando proximidade do usuário da tela do computador, consequentemente da webcam, obteve-se melhores resultados em relação à detecção de pisco. Uma vez que estando mais próximo, a região referente ao olho teria uma melhor qualidade.

5.2 Dificuldades Encontradas

A seguir será descrito algumas dificuldades encontradas neste trabalho, a partir das quais podem ser sugeridos trabalhos futuros e novas soluções.

1. **Ocorrência de falsos positivos na detecção dos olhos.** A eficiência da detecção dos olhos, realizadas tanto a partir da detecção de face quanto diretamente sobre a imagem como um todo, é evidente. Porém como citado na Seção 4.2, a utilização de um desses falsos positivos nas etapas seguintes à detecção dos olhos gera efeitos não esperados. Em alguns casos resultando na

perda de controle do cursor, podendo o mesmo a realizar movimentos e cliques bruscos e aleatórios.

2. **Movimentos do cursor realizados pelo piscar do olho.** Foi discutido na Seção 4.3, tendo como exemplo a Figura 26. Esse efeito é presente tanto no *Template Matching* convencional, quanto no *Template Matching* Adaptativo, porem o último tem esse feito de forma reduzida.
3. **Defasagem do *template*.** Explicada também na Seção 4.3. Diferentemente do caso anterior, esse efeito acontece apenas quando utilizado *templates* adaptativos. Por ser adaptativo, pode acontecer de o *template* modificar-se para algo que não é o olho, e passar a rastrear essa região.
4. **Cliques indesejados.** Geralmente ocorre quando em conjunto com o problema anterior, a região rastreada não necessariamente possui um círculo, portanto pode acontecer de haver cliques aleatórios, ou o clique manter-se pressionado.

5.3 Considerações Gerais e Trabalhos Futuros

Apesar dos impasses, o protótipo possui uma boa precisão e, como observado nos Testes e Resultados, já possibilita uma certa usabilidade para pessoas portadoras de necessidades especiais, sendo, portanto, uma alternativa promissora.

Esse trabalho possui um potencial para a adição de novas funcionalidades como o duplo clique, ou o acionamento do clique com o botão direito do mouse. Como também apresenta uma metodologia flexível o suficiente para se fazer alterações visando superar as dificuldades encontradas.

Visando-se trabalhos futuros, é sugerido abaixo algumas melhorias, a fim de obter-se êxito em meio as dificuldades:

1. **Seleção inteligente dos candidatos a olhos.** A metodologia proposta utiliza a detecção facial, delimitando a região para a detecção dos olhos. É possível então utilizar de padrões na geometria da face para estimar a região dos olhos, como apresentado por (JING, et al., 2010).

2. **Realizar o rastreamento da região entre ambos os olhos, assim como da região entre eles.** Assim, como alternativa, poderia ser utilizado o rastreamento da região central para a movimentação do cursor. Consequentemente, evitando que o piscar do olho afete o posicionamento do mouse. Além disso, a movimentação do cursor atualmente se dá, em termos práticos, pela translação e rotação da face, onde o olho é utilizado como ponto de referencia. A movimentação do cursor utilizando puramente a movimentação ocular não seria interessante do ponto de vista do usuário, pois supondo uma movimentação do cursor perfeitamente síncrona com o movimento da íris, isso seria algo indesejável durante a leitura de um documento, por exemplo. Caso realizado o rastreamento dessa região entre os olhos, isso não descartaria a necessidade de se rastrear o olho, visto que o mesmo ainda é necessário para a realização do clique.
3. **Rastreamento de ambos os olhos.** Tendo a finalidade de controlar o clique referente ao botão esquerdo e direto do mouse, o rastreamento de ambos os olhos permitiria associar cada olho a, respectivamente, um desses cliques.

6 REFERÊNCIAS

Akamai. "**The State of Internet.**" 2013. http://www.akamai.com/dl/documents/akamai_soti_q213.pdf?WT.mc_id=soti_Q213 (acesso em 4 de Janeiro de 2014).

BRADSKI, Gary, e Adrian KAEHLER. ***Learning OpenCV.*** Vol. 1. O'Reilly Media, Inc., 2008.

CHAVEZ, Roger Fredy Larico, Yuzo IANO, e Vicente Idalberto B. SABLON. "**Processo de Reconhecimento de Íris Humana: Localização rápida de Iris.**" *Revista do Instituto Nacional de Telecomunicações*, Novembro de 2006.

DINIZ, Pedro Henrique Bandeira. "**Controle de cursor do mouse atraves dos olhos para pessoas com limitação fisica.**" Monografia apresentada ao curso de Ciência da Computação da Universidade Federal do Maranhão para obtenção do grau de Bacharel em Ciência da Computação. (UNIVERSIDADE FEDERAL DO MARANHÃO - UFMA) 2012.

IBGE. "**Censo Demográfico.**" 2010. ftp://ftp.ibge.gov.br/Censos/Censo_Demografico_2010/Caracteristicas_Gerais_Religiao_Deficiencia/caracteristicas_religiao_deficiencia.pdf (acesso em 4 de Janeiro de 2014).

—. "**Censo Demográfico.**" 2000. http://www.ibge.gov.br/home/estatistica/populacao/censo2000/populacao/deficiencia_Censo2000.pdf (acesso em 4 de Janeiro de 2014).

JING, Zhang, Liu ZHIXING, e Zhuo LI. "**An Adaptative Tempalte Eye Location Based on Gabor Transform Method.**" *2010 International Conference on Computer Design and Applications (ICCDA).* (IEEE) 5 (2010): 178-181.

MARQUES FILHO, Ogê, e Hugo VIEIRA NETO. ***Processamento Digital de Imagens.*** Rio de Janeiro: Brasport, 1999.

MEIRELLES, Fernando S. "**24ª Pesquisa Anual do Uso de TI.**" FGV-EAESP. 2013. http://eaesp.fgvsp.br/sites/eaesp.fgvsp.br/files/arquivos/gvpesqti2013noticias.pdf (acesso em 4 de Janeiro de 2014).

PENG, Kun, Liming CHEN, Su RUAN, e Georgy KUKHAREV. "**A Robust Algorithm for Eye Detection on Gray Intensity Face without Spectacles.**" *Journal of Computer Science & Technology* 5 (October 2005): 127-132.

REIS, Eduardo de Jesus Coelho. **"Controle do Mouse com os Olhos - Mouse Eye Control".** http://www.youtube.com/watch?v=KPM7CPMHHyo (acesso em 10 de Janeiro de 2014). 2014.

VIOLA, Paul, e Michael JONES. "**Rapid object detection using a boosted cascade of simple features.**" *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on.* (IEEE) I (2001): 511-518.

—. "**Robust real-time object detection.**" *International Journal of Computer Vision* 2 (2004): 137-154.