

# Ferramenta para análise de informações do olhar

---

Adalberto Kazuo Kishi

Cristina Fang

Universidade De São Paulo  
Instituto de Matemática e Estatística  
Departamento de Ciência da Computação

Adalberto Kazuo Kishi

Cristina Fang

## **Ferramenta para análise de informações do olhar**

*MAC0499 Trabalho de formatura Supervisionado do Departamento de Ciência da Computação da Universidade De São Paulo para obtenção do grau de Bacharel em Ciência da Computação.*

Orientador: ***Prof. Dr. Carlos Hitoshi Morimoto***

São Paulo

17 de fevereiro de 2007

# Sumário

<b>1</b>	<b>Introdução</b>	<b>1</b>
1.1	Motivação	1
1.2	Descrição das funcionalidades do aplicativo	2
1.3	Aplicações do projeto desenvolvido	3
1.4	Perspectivas futuras	4
<b>2</b>	<b>Conceitos e Tecnologias Estudadas</b>	<b>6</b>
2.1	Atividades realizadas	7
2.2	Ferramentas Utilizadas	7
2.2.1	O rastreador de olhar	8
2.2.2	Java Advanced Imaging - JAI	8
2.2.2.1	Convolve	9
2.2.3	Java Media Framework - JMF	10
<b>3</b>	<b>O Aplicativo</b>	<b>13</b>
3.1	Dois tipos de visualização	13

3.1.1	Binária	14
3.1.2	Gaussiana	15
3.2	Classificação em Fixação ou Sacada	17
3.2.1	O Comportamento do Olhar	17
3.2.2	O Algoritmo	17
3.3	Modos de visualizar a classificação do olhar	19
3.3.1	Estático	19
3.3.2	Dinâmico sem decaimento	19
3.3.3	Dinâmico com decaimento	21
3.4	Vídeo	23
<b>4</b>	<b>Conclusão</b>	<b>24</b>
<b>5</b>	<b>Subjetiva</b>	<b>26</b>
5.1	Desafios e frustrações encontrados	26
5.2	Disciplinas relevantes	26
5.3	Interação com os membros da equipe	27
5.4	Cooperação na faculdade vs. Cooperação no trabalho	27
5.5	Agradecimentos	28

# Introdução

No fim do século XIX iniciaram os primeiros estudos do movimento dos olhos. Na Universidade de Paris, o Professor Emile Javal observou que, durante a leitura, o movimento dos olhos não era contínuo, mas realizado em séries de pequenas pausas até atingir o final de uma linha.

Os experimentos eram realizados sem auxílio de ferramentas tecnológicas. Em muitos casos especialistas observavam a olho nu o movimento dos olhos [5]. Entretanto, com o surgimento de diversas ferramentas e metodologias computacionais houve a possibilidade de aprofundar os estudos e realizar outros tipos de experimentos. Como consequência ocorreu um aumento na precisão dos resultados.

Uma das ferramentas que revolucionou o estudo do movimento dos olhos foi o rastreador de olhar (Eye Tracker), um meio de determinar a coordenada em que um indivíduo está olhando, dada uma superfície.

## 1.1 Motivação

Este projeto de conclusão de curso visa:

- Colocar em prática os conceitos e tecnologias aprendidos;
- Aprofundar conhecimentos em outras tecnologias;

- Criar uma ferramenta básica de análise de dados que utilize o rastreador de olhar já existente, e desenvolvido no Laboratório de Tecnologias para Interação (LaTIn)
- Tornar a tecnologia do rastreador acessível a outros grupos de pesquisa.

## 1.2 Descrição das funcionalidades do aplicativo

### 1. Classificação do comportamento do olhar

Dado um objeto (imagem, texto ou vídeo) o aplicativo desenvolvido recebe uma coordenada (x, y) da tela do monitor, e através de um algoritmo, a classifica como:

- Fixação: caracterizada por período sem movimentação do olho, com duração superior a 100ms.
- Sacada: caracterizada por deslocamentos rápidos do olho, com duração inferior a 100ms.

### 2. Modos de visualização do objeto

A partir da coordenada, são disponibilizados dois tipos de visualização:

- Binária: neste tipo de visualização apenas um círculo da imagem pode ser observado - com centro na coordenada observada pelo usuário. As demais regiões da tela estão escurecidas.
- Gaussiana: neste tipo de visualização, ocorre uma composição de três camadas da imagem. Cada camada possui um nível de nitidez. Quanto maior a distância da camada ao ponto observado menor a nitidez da camada.

### 3. Modos de visualização da interação com o usuário

Sacadas são representadas por linhas contínuas e fixação, por círculos. A partir da classificação da coordenada, são disponibilizados três formas gráficas de visualização dos locais observados:

- Estático: O usuário interage com objeto (texto, imagem ou vídeo) e apenas visualiza as classificações quando terminar a interação.
- Dinâmico sem decaimento: O usuário interage com o objeto e visualiza, simultaneamente, as classificações recentes.
- Dinâmica com decaimento: A visualização de cada classificação tem um tempo de permanência na tela, ou seja, somente as mais recentes permanecem na tela. (efeito *Fade out*)

### 1.3 Aplicações do projeto desenvolvido

#### 1. Auxílio na detecção de problemas de leituras em adultos e crianças.

O Aplicativo oferece um processo cognitivo para detecção de distúrbios na leitura. Por exemplo, o modo de visualização estático pode ser utilizado para analisar o padrão de leitura apresentado pelo indivíduo. Através desta análise é possível diagnosticar distúrbios como a dislexia [2].

#### 2. Estudos psicológicos e neuro-psicológicos

Por receber do usuário uma resposta gráfica frente a um estímulo visual, seja imagem, texto ou vídeo. Essa ferramenta pode ser utilizada em pesquisas cognitivas. Por exemplo, pedir que uma pessoa observe uma imagem por alguns segundos e tente memorizar a maior quantidade de detalhes da imagem. A partir das informações obtidas pode-se analisar quais são os pontos relevantes para a memorização [10].

3. Pesquisas de marketing e desenho industrial. Ao desenvolver uma propaganda ou design de um produto, pode-se apresentá-la a um grupo de usuários, analisar o efeito obtido por tais estímulos visuais e, assim, fazer as alterações necessárias para obter o impacto desejado.

4. Auxílio no design de páginas web

Medição de usabilidade: possibilidade de poder observar a percepção do usuário ao utilizar uma determinada interface. Em sítios, o aplicativo pode avaliar:

As primeiras impressões de um usuário ao acessar um sítio comercial.

- Quais textos, imagens ou vídeos despertam mais interesse.
- Se as propagandas são vistas ou ignoradas.

## 1.4 Perspectivas futuras

Devido à quantidade de aplicações que poderiam ser anexadas a esta ferramenta, são muitas as possibilidades de melhorar o projeto desenvolvido. Serão citadas algumas:

- Implementar mapas de calor (heat maps) que, dada uma imagem, utilizam a escala de cores para diferenciar em que frequências os locais foram observados. Cores quentes <sup>1</sup> para frequências maiores e cores frias, para as menores.
- Criação de um identificador de padrões, uma ferramenta que analise observações de usuários diferentes e verifique se pertencem a um mesmo padrão.

---

<sup>1</sup>Cores quentes são aquelas que transmitem sensação de calor, associadas ao sol ao fogo (vermelho, laranja, amarelo) e cores frias transmitem sensação de frio, associadas ao céu, à água, ao gelo às árvores (azul, violeta, verde)



- Desenvolver um editor de texto, como sugerido pelo Prof. Dr. Carlos Hitoshi Morimoto, que como o editor descrito ao longo desta monografia, tenha um paradigma diferente. Que apresente características que contornem as desvantagens apresentadas por uma interface que se utilize apenas do rastreador de olhar.

## Conceitos e Tecnologias Estudadas

Alguns artigos contribuíram no desenvolvimento deste projeto. Inicialmente, o *Eye gaze tracking techniques for interactive applications* [3] apresentou a tecnologia do rastreador de olhar e suas técnicas mais utilizadas.

Os primeiros rastreadores tinham usabilidade comprometida, devido à mobilidade limitada do usuário e à necessidade de calibrar a ferramenta constantemente (o processo de calibração é descrito na seção *Ferramentas Utilizadas - O rastreador de olhar*). Entretanto, como visto no artigo acima, já são conhecidas técnicas para contornar os problemas de usabilidade apontados.

Contudo pode-se notar outra dificuldade, interagir com uma interface utilizando o olhar. O artigo (Magic) Pointing [12] mostra uma técnica utilizada para otimizar essa interação. Quando um cursor, controlado pelo olhar, está nas proximidades de um objeto, este o atrai para si. Assim, os pontos relevantes da interface são facilmente atingidos.

Em seguida houve um contato com aplicações que estão sendo desenvolvidas atualmente. Dentre elas, editores de texto: com teclados virtuais ou com um paradigma mais adequado, como o editor descrito no artigo *Artificial intelligence: Fast hands-free writing by gaze direction* [11], que utiliza modelos estatísticos para prever quais letras têm maior probabilidade de serem escolhidas baseado nas escolhas anteriores. Além de editores de texto, existem mapas de calor (*heat maps*), que dada uma imagem, indicam quais os locais foram mais observados, ou seja, utilizam a escala de cores para diferenciar em que frequências os locais foram observados.

## 2.1 Atividades realizadas

Após a leitura dos artigos, foram realizadas as seguintes tarefas:

1. Familiarização com as Interfaces de Programação de Aplicativos(API) *Java Advanced Imaging (JAI)* e *Java Media Framework(JMF)*;
2. Implementação dos filtros binário e gaussiano para a imagem, utilizando o mouse ao invés do rastreador, e obtendo as coordenadas de um arquivo texto;
3. Implementação de um reprodutor de vídeo (*player*);
4. Familiarização com o rastreador de olhar;
5. Implementação de um algoritmo para classificar os tipos de movimento do olhar: sacada e fixação;
6. Implementação de uma interface de testes, simples, para o item anterior;
7. Integração do que foi realizado nos itens dois e três com o rastreador de olhar, ou seja substituição do mouse pelo rastreador, para obter as coordenadas observadas;
8. Implementação dos modos de visualização estático, dinâmico com e sem decaimento;
9. Implementação das funcionalidades listadas para vídeo.

## 2.2 Ferramentas Utilizadas

Enumeradas as tarefas realizadas, a seguir serão descritas as ferramentas e tecnologias estudadas.

### 2.2.1 O rastreador de olhar

O rastreador de olhar é composto de uma câmera e dois conjuntos de luzes infra vermelhas (leds). O primeiro conjunto de luz, é posicionado ao redor da lente da câmera, e gera uma imagem clara do olho do usuário; o segundo, posicionado nos quatro cantos do monitor, e gera uma imagem escura. A partir dessas imagens, e do brilho ocular, obtém-se a posição em que o usuário está olhando.

Para que essa ferramenta funcione devidamente, ocorre um processo de calibração, onde o usuário olha para alguns pontos na tela do computador. E a ferramenta se adapta às características particulares do olho de cada usuário.

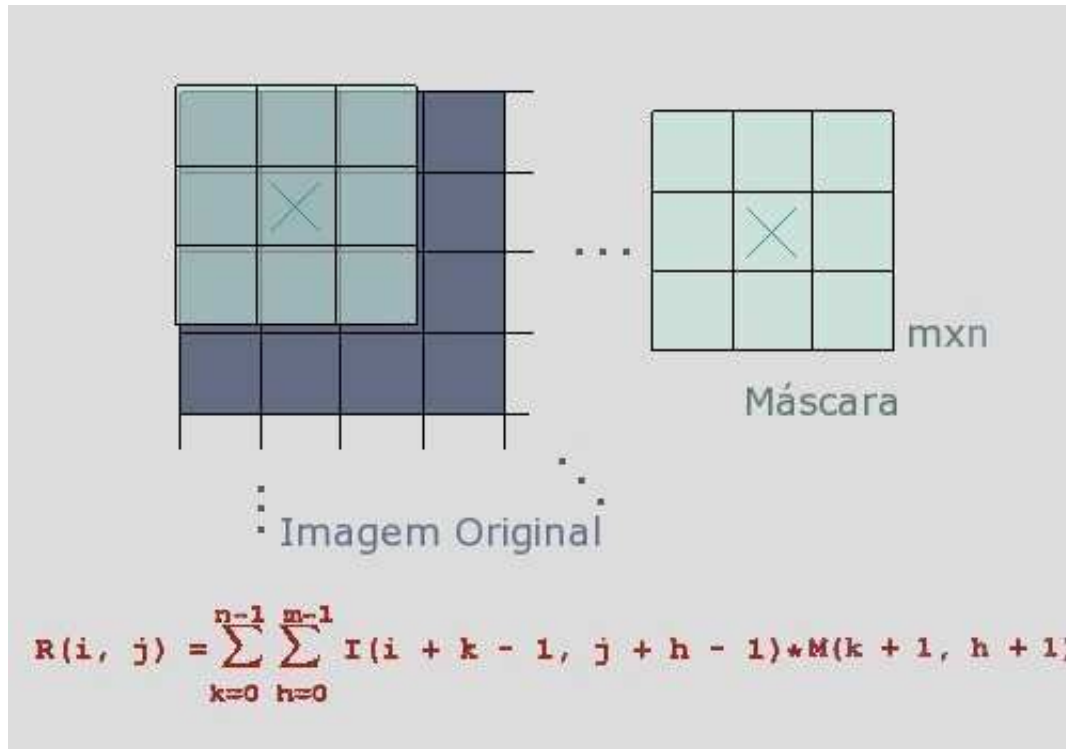
### 2.2.2 Java Advanced Imaging - JAI

JAI (*Java Advanced Imaging*) é uma API , ou conjunto de classes, que permite a inclusão de diversas rotinas para processamento de imagens em aplicações (Java). O poder e a flexibilidade do JAI, aliados à facilidade de programação em Java, permitem uma rápida criação de aplicações para processamento de imagens. A convolução (*convolve*), o histograma (*histogram*), inversão(*invert*), a conversão de cores(*colorconvert*) são uma pequena amostra da variedade de operações que o JAI é capaz de realizar.

O JAI é uma API extensível o que permite que novas operações sejam adicionadas de forma a se tornarem parte nativa do JAI. Esta ferramenta oferece muitas vantagens tais como flexibilidade, eficiência, orientação a objetos, independência de plataforma e além disso pode ser obtido gratuitamente e sem restrições de distribuição (no site da Sun Microsystems).

## 2.2.2.1 Convolve

A operação de convolução *convolve* foi utilizada no modo de visualização gaussiano para gerar as imagens desfocadas.



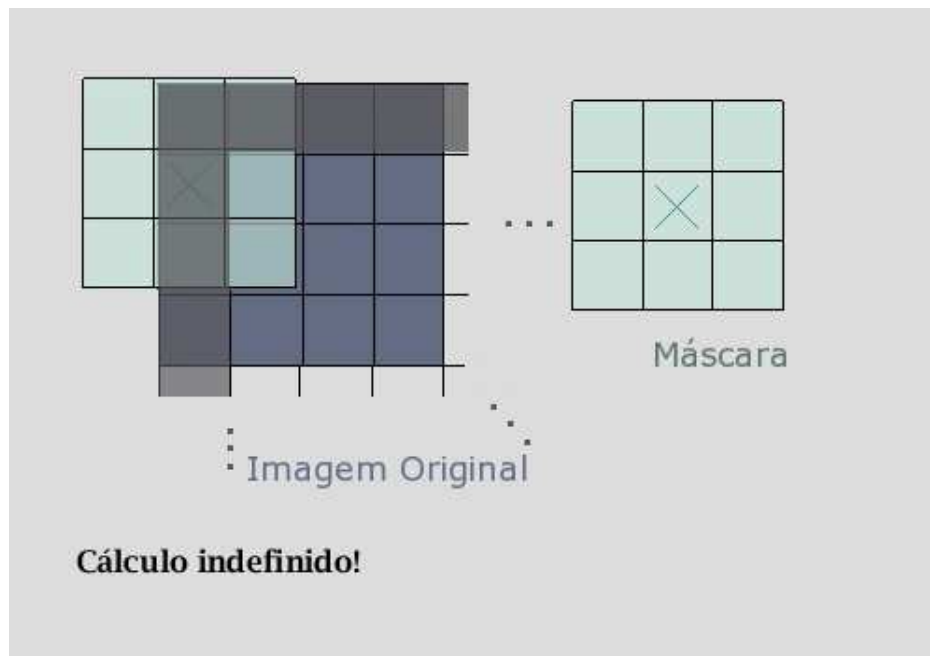
**Figura 2.1** Convolução

Convolução é uma operação espacial comum a muitas operações de processamentos de imagens. É aplicada a uma imagem através de uma máscara, a qual determina o efeito resultante.

A convolução é composta de uma série de iterações. Em cada iteração a máscara é sobreposta à um grupo de pixels da imagem. Cada pixel da imagem(fonte) é multiplicado pelo pixel que o sobrepõe, e estes produtos são somados para obter o pixel resultante. Esse procedimento é repetido para todos os pixels da imagem, gerando o efeito de suavização (*smoothing*). A figura 2.1 mostra uma das iterações do procedimento descrito acima.

A convolução, como qualquer operação que utiliza pixels vizinhos para calcular o pixel resultante, não possui definição para os pixels da borda da imagem.

Seja uma imagem  $N \times N$  e uma máscara  $M \times M$  (em pixels). Aplicando a convolução, será obtida uma imagem resultante com  $4(N-1) * \lfloor M/2 \rfloor$  pixels indefinidos, os quais são representados pela cor preta. A figura 2.2 ilustra essa situação.



**Figura 2.2** Convolução

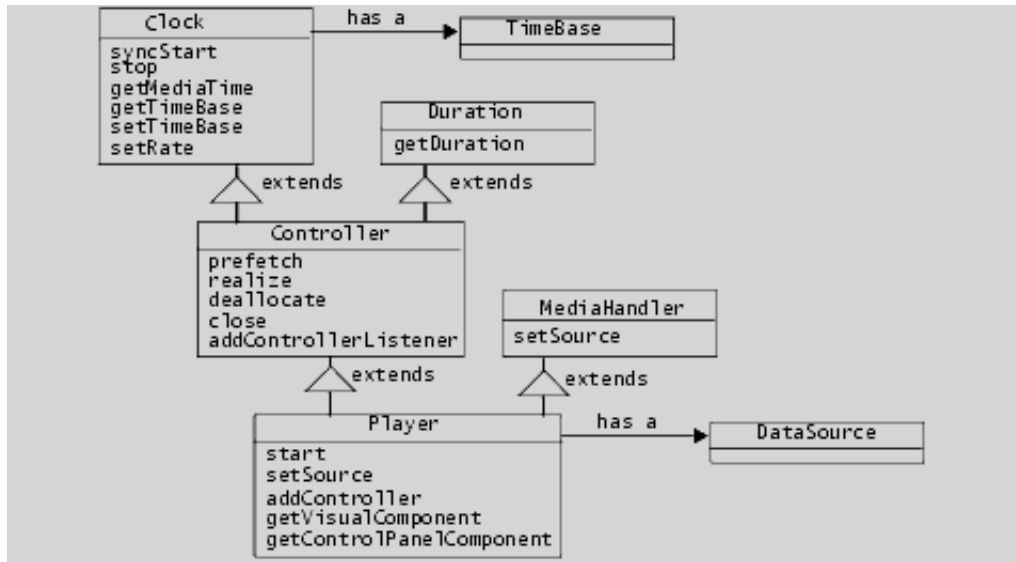
O tamanho da máscara não pode ter dimensão maior que o da imagem fonte.

### 2.2.3 Java Media Framework - JMF

O Java Media Framework (JMF) é uma API Java que permite adicionar áudio, vídeo e outras funcionalidades de mídia em aplicações e applets. Este pacote oferece capacidades de transmissão, reprodução e criação de streams, codificação e decodificação de múltiplos formatos. A versão JMF 2.0 API foi desenvolvida pela Sun Microsystems, Inc. e IBM Corporation.

Essa API foi utilizada no desenvolvimento de um reprodutor de vídeo e suas interações, como aplicações de filtros e modo de visualização estático.

Um reprodutor de vídeo (*Player*) processa os fluxos de dados de mídia e os renderiza, entretanto não fornece controle sobre o processamento ou renderização desses dados.



**Figura 2.3** Reprodutor de vídeo do JMF.

Existem seis estados possíveis: dois primários, *Started* ou *Stopped*, definidos pela interface *Clock*; e para facilitar o gerenciamento dos dados, o estado *Stopped* foi dividido em cinco outros estados: *Unrealized*, *Realizing*, *Realized*, *Prefetching* e *Prefetched*. Figura 2.4

Para realizar a reprodução de um vídeo o *player* passa por todos os estados:

1. *Unrealized*

Nesse estado, o *Player* é instanciado e não possui informações sobre a mídia que irá reproduzir. Ao chamar *realize* ocorre a mudança para o estado *Realizing*;

2. *Realizing*

Determina os recursos necessários para a reprodução dos dados de mídia. Esses recursos (exceto os de uso exclusivo) são adquiridos apenas uma vez.

### 3. *Realized*

Neste estado, o *Player* possui conhecimento de como renderizar os dados e pode fornecer componentes e controles visuais.

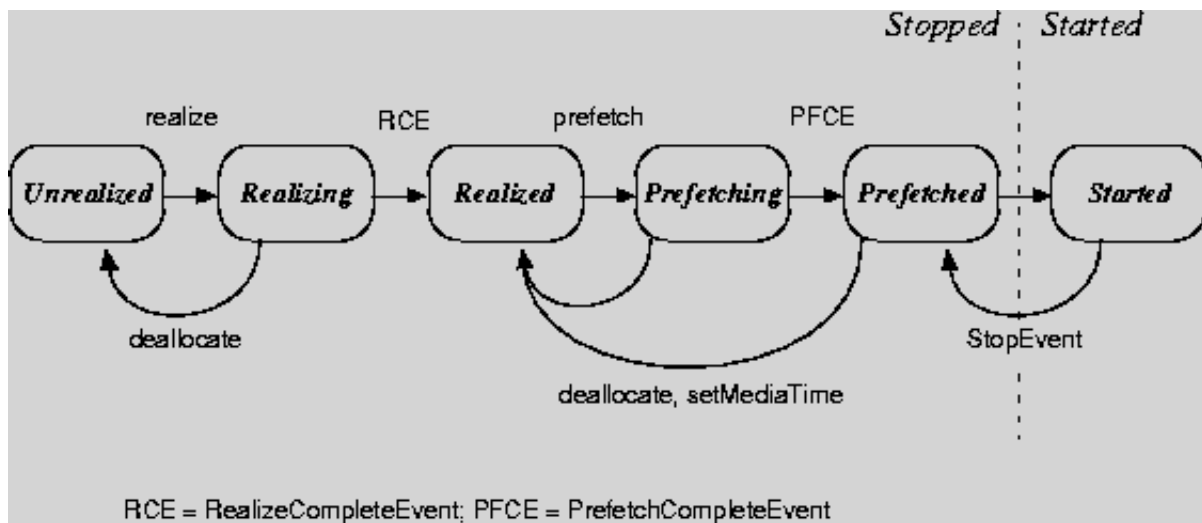
### 4. *Prefetching*

Pré-leitura dos dados de mídia e a obtenção dos recursos de uso-exclusivo (recursos limitados que podem ser utilizados por apenas um *Player*).

### 5. *Prefetched*

O *Player* está pronto para iniciar. Quando *start* é chamado o *Player* passa para o estado *Started*.

### 6. *Started* Reprodução do vídeo.



**Figura 2.4** Autômato representando os estados do JMF

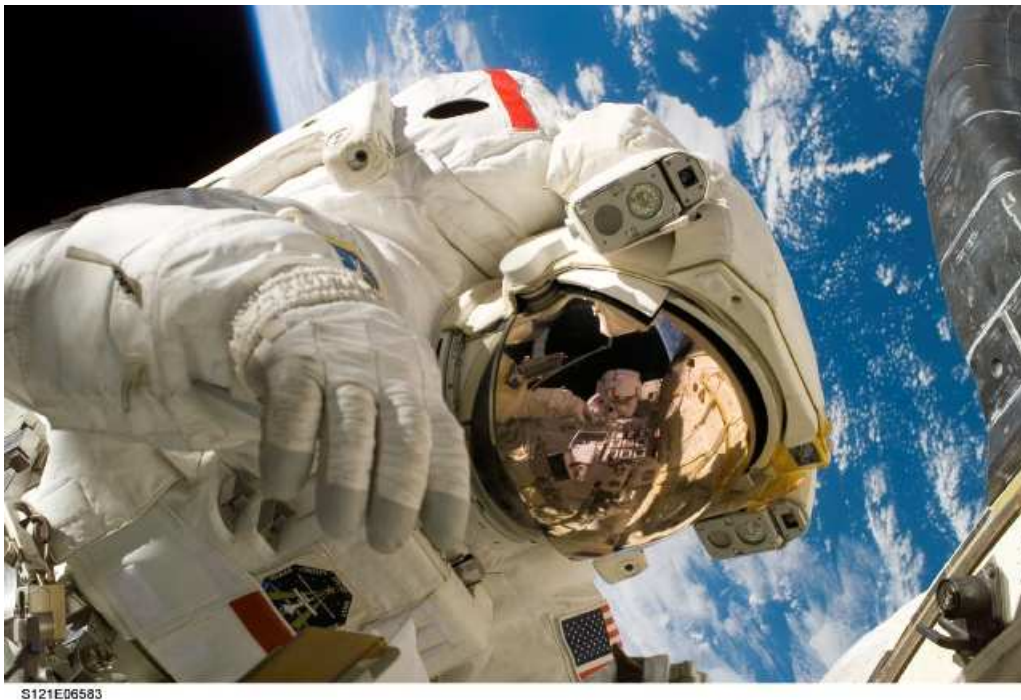


## CAPÍTULO 3

# O Aplicativo

Descrição detalhada do aplicativo.

### 3.1 Dois tipos de visualização



**Figura 3.1** Imagem original.

Inicialmente, o usuário escolhe uma imagem (figura 3.1) ou vídeo, e pode optar por dois tipos de visualização: Binária (figura 3.2) e Gaussiana (figura 3.4). Em ambos os tipos de

visualização evidencia-se a coordenada para onde o usuário está olhando.

### 3.1.1 Binária

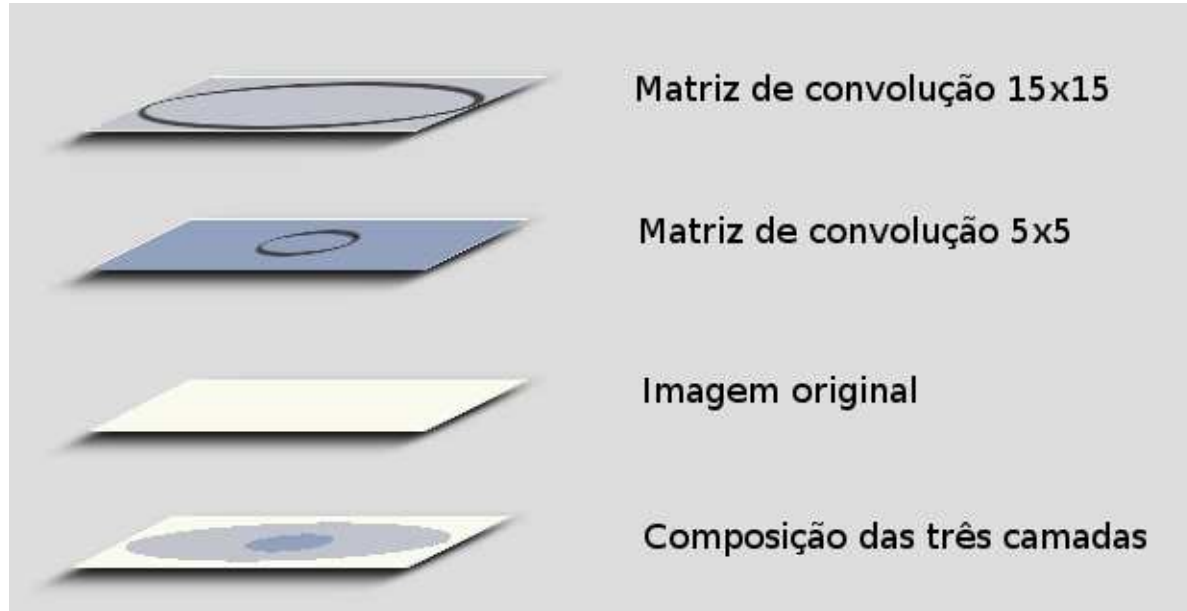


**Figura 3.2** Utilizando o tipo de visualização binário

Neste tipo de visualização, apenas um círculo da imagem pode ser observado e as demais regiões da tela estão escuras, onde o centro do círculo é a coordenada detectada pelo rastreador. (Efeito Lanterna)

Inicialmente a imagem resultante recebe a cor preta. Em seguida, para a coordenada observada, obtém-se um círculo preenchido com a textura da imagem original. Este círculo é sobreposto na imagem resultante.

### 3.1.2 Gaussiana



**Figura 3.3** Utilizando o tipo de visualização gaussiano

Neste tipo de visualização, é utilizada a operação *convolve* (convolução), que pode ser utilizada para suavizar imagens ou diminuir a nitidez da imagem.

Os dois efeitos são determinados por uma matriz. Quanto maior os valores e o tamanho da matriz, menor a nitidez da imagem.

Dada uma imagem original A, são geradas duas outras imagens B e C com níveis de nitidez distintos. O grau de nitidez diminui de A para C.

Buffers são utilizados para armazenar as três imagens:

- Imagem A, é a imagem original (buffer A);
- Imagem B, é a imagem original convoluída (buffer B).
- Imagem C, é a imagem original convoluída, entretanto com uma máscara de tamanho

maior do que a utilizada no caso anterior (buffer C).



**Figura 3.4** Utilizando o tipo de visualização gaussiano

A criação e o armazenamento dessas imagens ocorrem somente quando o usuário abre uma imagem. As demais interações são feitas utilizando os buffers criados. E no caso, o modo de visualização gaussiano é obtido através de uma composição das três imagens:

- A tela é preenchida com o buffer C, imagem menos nítida;
- Sobreposição de um círculo do buffer B;
- Sobreposição de um círculo, de tamanho menor, do buffer A (a imagem original).

Como mencionado anteriormente, ocorrerá uma perda devido à convolução utilizar pixels vizinhos para determinar cada pixel resultante. Como consequência a imagem diminui de tamanho quando esse modo de visualização é utilizado.

Na figura 3.3, ilustramos as passagens descritas acima.

## **3.2 Classificação em Fixação ou Sacada**

### **3.2.1 O Comportamento do Olhar**

Como foi visto, devido a natureza do olho e seus atributos, pode-se classificar o tipo de olhar em:

- Fixação: caracterizada por período sem movimentação do olho, com duração superior a 100ms.

- Sacada: caracterizada por deslocamentos rápidos do olho, com duração inferior a 100ms.

Um indivíduo não é capaz de deslocar o olhar continuamente. Pode-se notar a ocorrência de uma série de pequenas oscilações, devido à natureza do olhar. Assim a distinção entre sacada e fixação não é imediata e foi necessário desenvolver um algoritmo para classificar o comportamento do olhar.

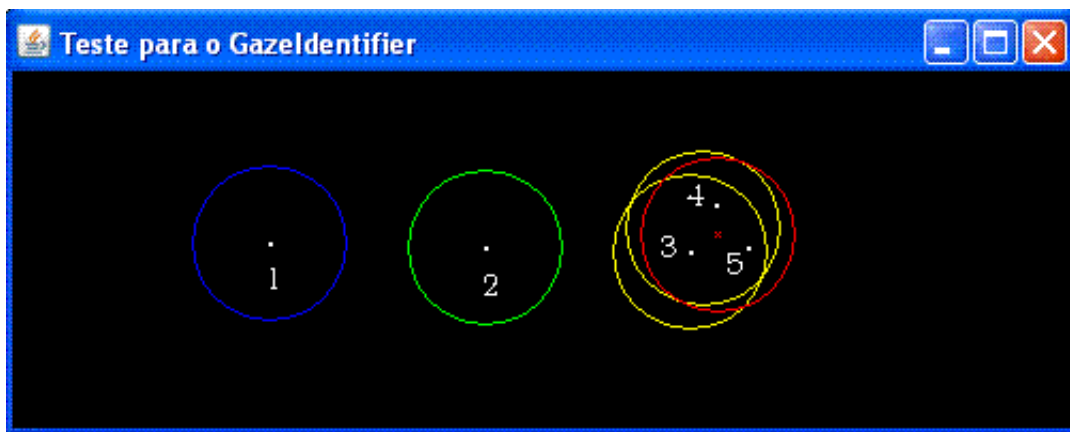
### **3.2.2 O Algoritmo**

A ocorrência de uma fixação será caracterizada por oscilações delimitadas por uma certa região. Essa região é determinada com base na margem de erro do rastreador de olhar.

O aplicativo recebe as coordenadas em tempo real. Para determinar se ocorreu uma fixação, analisa a coordenada recebida com as mais recentes. Os candidatos a fixações são os subconjuntos disjuntos dessa amostra de coordenadas recentes e que pertençam a uma mesma

região. Na figura 3.5:

- As coordenadas rastreadas estão representadas por pontos brancos e suas seqüências no tempo, por números;
- Os candidatos a fixação estão representados por círculos coloridos (azul, verde e amarelo), onde o centro é a média das coordenadas que estão na mesma região. <sup>1</sup> Note que quando a coordenada três é recebida a candidata é ela mesma. Entretanto com a coordenada quatro, a possível fixação é deslocada para o ponto equidistante a ambas. O mesmo ocorre quando a coordenada cinco é recebida.<sup>2</sup>
- A fixação obtida pelo algoritmo está representada em vermelho, por um x e um círculo com centro nesse x.



**Figura 3.5** Teste do Algoritmo de Classificação

<sup>1</sup>Duas coordenadas estão na mesma região quando os círculos com centro nas mesmas se interceptam

<sup>2</sup>Nesse exemplo, a amostra tem tamanho cinco, assim três pontos em uma mesma região são suficientes para determinar uma fixação

### 3.3 Modos de visualizar a classificação do olhar

Dado o algoritmo para determinar fixações e sacadas, foram criadas formas de visualizar como foi classificado o olhar do usuário.

Linhas são utilizadas para indicar a ocorrência de sacadas; e pontos, para indicar fixações.

Em todos os modos de visualização, o usuário inicialmente abre uma imagem, opta por um dos modos, e recebe um *feedback* gráfico das classificações feitas pelo aplicativo.

Nas próximas seções descreveremos os modos de visualização que o aplicativo oferece ao usuário.

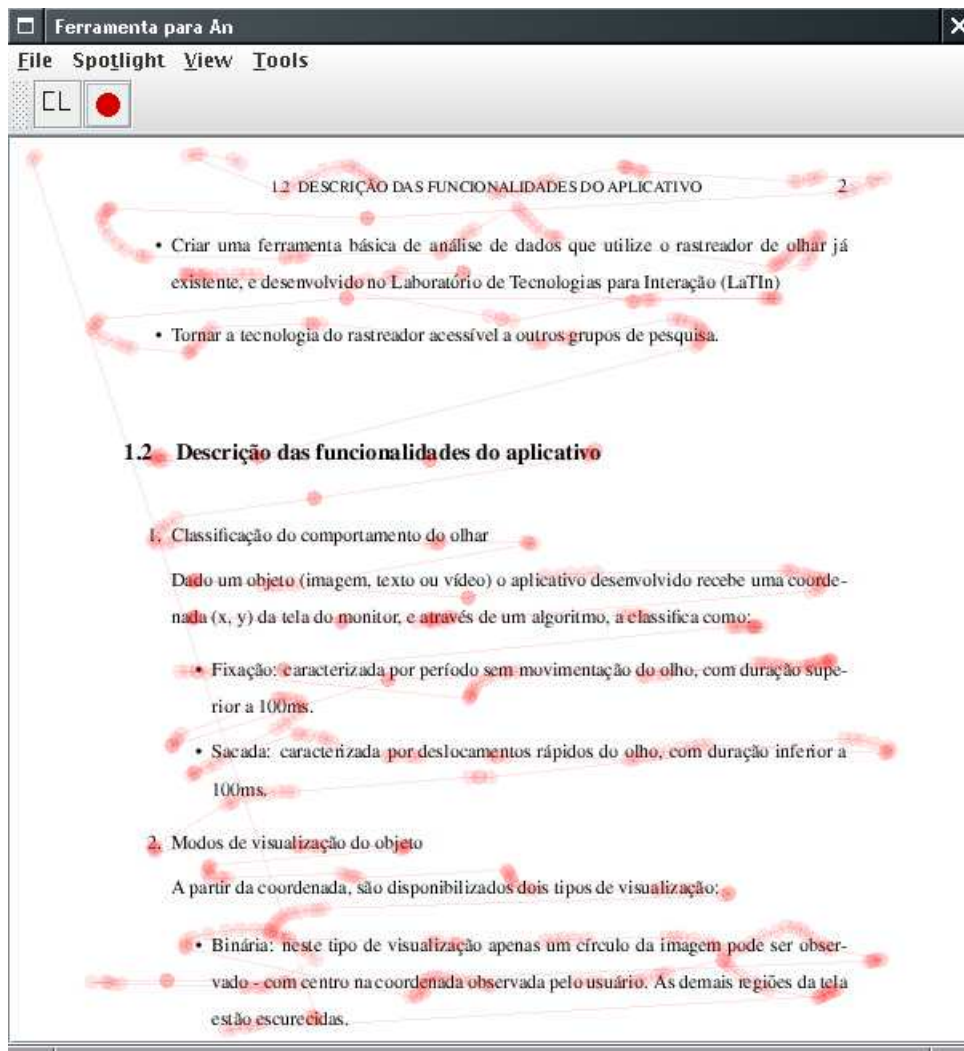
#### 3.3.1 Estático

O usuário começa a gravar as coordenadas observadas e apenas quando termina de gravar, observa todas as classificações obtidas até o momento.

Apenas para o estático as coordenadas são gravadas em um arquivo texto, e quando o usuário opta por visualizá-las, as coordenadas do arquivo texto são processadas e obtém-se o efeito desejado, ou seja, as classificações obtidas pelo algoritmo acima são apresentadas na tela, linhas indicando sacadas e pontos indicando fixações.

#### 3.3.2 Dinâmico sem decaimento

O usuário começa a gravar as coordenadas observadas e, em tempo real, vai observando as classificações obtidas pelo aplicativo. É dita sem decaimento, pois as coordenadas são mostradas ao usuários e continuam na tela até que o usuário realize outra operação (figura 3.6).



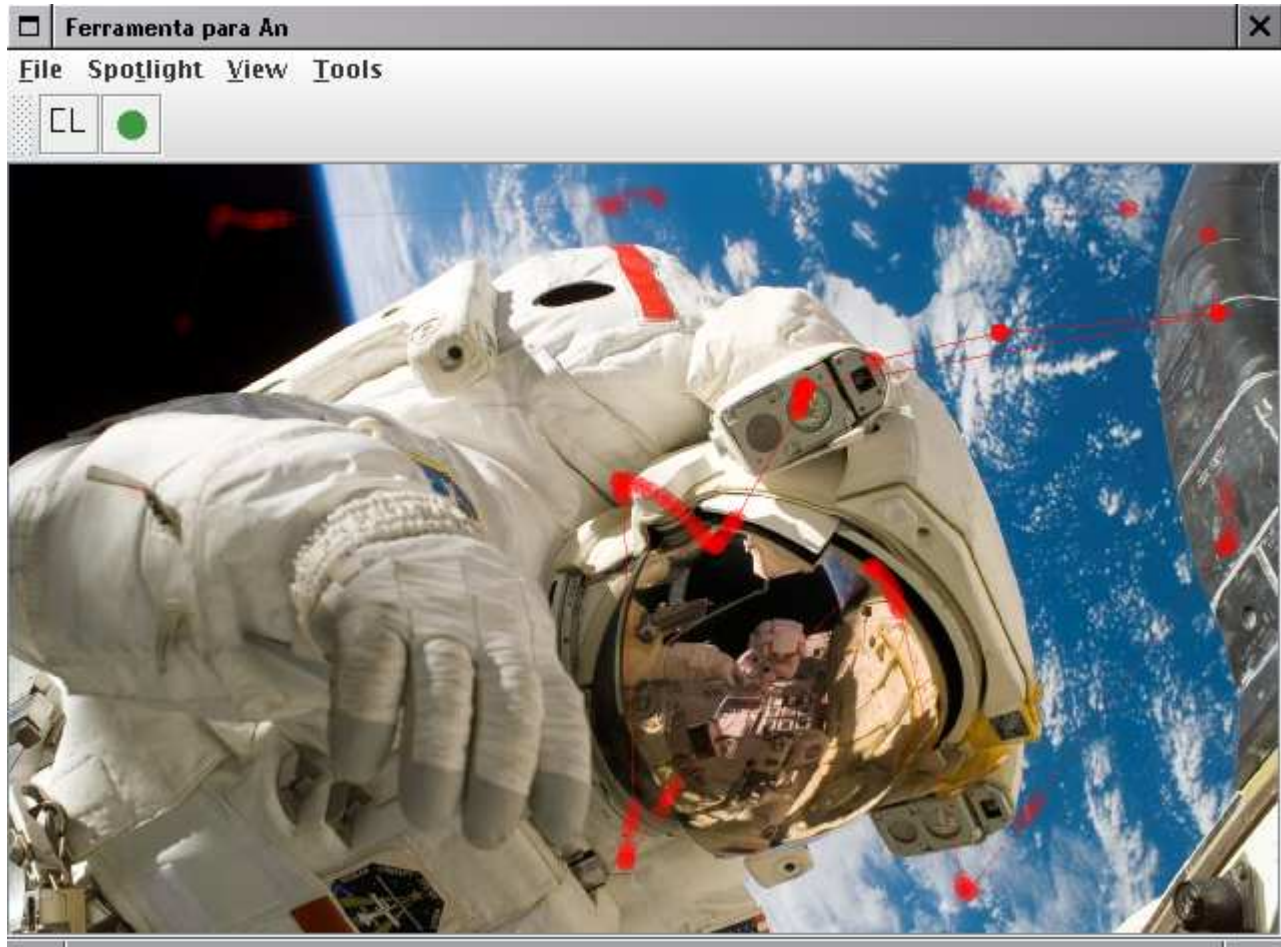
**Figura 3.6** Dinâmico sem decaimento

As fixações e sacadas são armazenadas em uma máscara com o tamanho da imagem. Um buffer é utilizado para armazenar o tipo de visualização(binário, gaussiano ou normal) aplicado ao objeto original (imagem ou frame atual do vídeo). Para atingir o efeito desejado, o buffer é desenhado na tela juntamente com a máscara sobreposta.



### 3.3.3 Dinâmico com decaimento

Assim como o **Dinâmico sem decaimento**, o Dinâmico com decaimento, realiza o *feed-back* gráfico, em tempo real. É dita com decaimento, pois cada ponto ou linha tem um tempo de vida para ser mostrado ao usuário, ou ainda, os pontos e linhas mais antigos vão desaparecendo a medida que novos vão surgindo (*fade out*).



**Figura 3.7** Dinâmico com decaimento (Fade out)

Neste modo também é utilizado um buffer para armazenar o tipo de visualização aplicado ao objeto. Para obter o decaimento é utilizada uma fila FIFO(*First-in-First-out*), contendo as fixações mais recentes. Para obter a imagem resultante, a fila é percorrida atribuindo graus de



**Figura 3.8** Composições dos modos de visualização com filtros

transparências (*alfa*): maior para fixações mais antigas, menor para as mais recentes <sup>3</sup>. Assim obtém-se o efeito *fade out*, que pode ser observado na figura 3.7

Existe também a possibilidade de fazer composições dos filtros (gaussiano e binário) com os modos de visualização apresentados acima. A figura 3.8 mostra a composição do filtro gaussiano com o modo de visualização Dinâmico com decaimento;

---

<sup>3</sup>O alfa é inversamente proporcional ao grau de transparência e, portanto, menor alfa para fixações mais antigas, maior para as mais recentes

## 3.4 Vídeo

O JMF, mencionado no capítulo anterior, foi utilizado para desenvolver um reprodutor de vídeo e suas interações, como aplicação dos filtros e do modo estático. Foi escolhido o JMF 2.1.1e por fornecer uma solução eficiente no desenvolvimento de aplicações de mídia.

Apesar do projeto apresentar uma necessidade pontual de JMF frente a diversidade de funcionalidades que esta ferramenta apresenta, foi necessário estudar alguns conceitos fundamentais. Alguns foram mencionados no capítulo anterior, e outros serão discutidos a seguir.

Como visto, um reprodutor de vídeos possui seis estados: Unrealizing, Realizing, Realized, Prefetching, Prefetched e Started. No estado Prefetched é possível ter acesso aos frames de um vídeo com o auxílio de duas interfaces do JMF:

- `FrameGrabbingControl`

Captura o frame a partir de um stream de vídeo e o armazena em um buffer, facilitando a manipulação.

- `FramePositioningControl`

Fornece precisão no controle do posicionamento de um frame em reprodutores e processadores de vídeos. Com esta interface é possível armazenar o frame em um buffer e posteriormente transformá-lo em um `Image` (classe do awt).

Durante o desenvolvimento observou-se que capturar um frame em um buffer consumia tempo excessivo no processamento, comprometendo a sincronização do vídeo. A solução encontrada foi armazenar as imagens em disco rígido e recuperá-las sob demanda de uma thread sincronizada.

Para a convolução foi necessário um armazenamento inicial pois a operação de convolução interferia na sincronização do vídeo.

## Conclusão

Com o desenvolvimento desse projeto, foi obtida a experiência de trabalhar em equipe com algumas diferenças em relação aos trabalhos realizados ao longo do curso: realizar decisões quanto à eficiência versus o consumo de memória e realizar um projeto de longa duração.

### 1. Ferramenta para análise do rastreador de olhar

Pesquisadores descobriram no olhar um bom modo de obter informações (feedback) de um usuário. Trazendo, assim, benefícios a diversas áreas, dentre elas marketing, ciência da computação (inteligência artificial, interação homem computador) e psicologia.

A ferramenta desenvolvida fornece aos usuários uma interface para o rastreador de olhar. Tornando-o acessível a qualquer grupo de pesquisa, não sendo necessário conhecimentos computacionais avançados. Os usuários podem facilmente observar as interações que estão ocorrendo com o rastreador e fazer os experimentos apropriados. É possível fazer análises dos elementos de imagens (paisagens, textos, objetos) ou de vídeos.

### 2. Rastreador de olhar

Inicialmente, utilizar um rastreador de olhar era incomodo devido à pouca mobilidade que este proporcionava ao usuário, com suportes para queixo, capacetes com câmeras, suportes com visores, entre outros. Entretanto os avanços tecnológicos trouxeram mais comodidade e o uso passou a ser menos invasivo e hostil. E embora tenham ocorrido grandes avanços, ainda não se tornou mais interessante para a maioria dos usuários substituir o mouse por um rastreador de olhar. Contudo, para pessoas com deficiências motoras tornou-se uma alternativa interessante.

Uma desvantagem do rastreador é possuir uma margem de erro ao obter as coordenadas. Assim, em um sistema, em que o mouse foi substituído pelo rastreador, precisaria ter uma interface com elementos maiores, ou seja, os locais onde o usuário clicaria com o mouse precisariam ser aumentados de tal forma que abrangessem essa margem de erro. Assim, uma grande área útil da tela seria desperdiçada.

### 3. API's

Com o desenvolvimento deste projeto foi possível adquirir conhecimentos de algumas funcionalidades das Interfaces de Programação de Aplicativos (API) utilizadas: JAI e JMF.

Essas bibliotecas contribuíram muito no projeto devido à facilidade de manipulação de imagens e vídeos que as mesmas proporcionam.

- O JAI fornece inúmeras abstrações de operações para manipular imagens (convolve, histogram, invert, colorconvert). Em consequência, oferece ao desenvolvedor facilidade e eficiência na execução de efeitos em imagens, como os obtidos neste projeto (gaussiano).
- O JMF apresentou boas soluções para a manipulação de frames em tempo real, apesar da última atualização ter ocorrido em 2004.

## Subjetiva

### 5.1 Desafios e frustrações encontrados

Um dos desafios encontrados foi não ter cursado disciplinas que favoreceriam o desenvolvimento do aplicativo. Tais como Visão e Processamento de Imagens. Ou ainda estar cursando Computação Gráfica durante o desenvolvimento do projeto, quando teria sido melhor ter cursado antes.

Corrida contra o tempo, um ano é pouco para escolher um projeto, estudar as tecnologias e ferramentas que o projeto irá necessitar e desenvolvê-lo.

### 5.2 Disciplinas relevantes

Disciplinas que considero que foram importantes para o desenvolvimento do projeto:

1. MAC0122 - Princípios de Desenvolvimento de Algoritmos
2. MAC0211 - Laboratório de Programação
3. MAC0414 - Linguagens Formais e Autômatos
4. MAC0446 - Princípios de Interação Homem-Computador
5. MAC0338 - Análise de Algoritmos

6. MAC0438 - Programação Concorrente

7. MAC0420 - Introdução à Computação Gráfica

### **5.3 Interação com os membros da equipe**

Desenvolvi o projeto com o Adalberto Kazuo Kishi. Já havíamos desenvolvido, anteriormente, outros projetos menores (exercícios-programa). E apesar, de nunca termos desenvolvido projetos tão grandes, e com uma duração tão extensa, não tive surpresa alguma em relação à nossa interação. Sempre muito paciente, e com muita boa vontade, pude aprender muito, tanto na parte profissional quanto na pessoal, trabalho em equipe.

### **5.4 Cooperação na faculdade vs. Cooperação no trabalho**

Talvez devido à dificuldade do curso, a união sempre esteve muito presente durante cada disciplina cursada.

E em muitos casos, colegas fizeram papel de grandes professores. Cada um aprende no seu ritmo, e têm ritmos diferentes para aprender disciplinas diferentes. Assim, os que aprendiam mais rápido certa disciplina transmitiam seus conhecimentos. E em outras, recebiam conhecimentos dos colegas.

Mas posso dizer que no trabalho também tive sorte, sempre houve muita cooperação e uma constante troca de conhecimentos. Realmente trabalhamos em equipe, e tem sido uma ótima experiência profissional.

## 5.5 Agradecimentos

Gostaria de agradecer,

**ao Adalberto Kazuo Kishi**, com quem tive o privilégio e a satisfação de desenvolver esse projeto. Demonstrou muita paciência e boa vontade, passando muitos fins de semana comigo desenvolvendo o projeto;

**ao Prof. Dr. Carlos Hitoshi Morimoto**, que se colocou a nossa total disposição. Sempre muito participativo e atencioso, com muitas idéias interessantes e inovadoras, e sempre com muita paciência frente a nossa inexperiência. Nos forneceu todo o material necessário para desenvolver o projeto e nos auxiliou muito com as nossas reuniões semanais;

**ao Flávio Coutinho**, desenvolvedor do rastreador de olhar que utilizamos, também se colocou a nossa total disposição. E nos ajudou muito na integração do rastreador de olhar com o nosso aplicativo;

**aos familiares e amigos** que sempre nos apoiam nos momentos difíceis, e que talvez tornassem inviável a realização de qualquer projeto se não estivessem presentes nos momentos certos. Sou muito grata a todos os meus amigos que me ajudaram ao longo do curso, em especial ao Rafael Antônio Cosentino, um grande amigo. Aprendi muito com eles não só profissionalmente como pessoalmente.

**à Profa. Dra. Nami Kobayashi e ao Prof. Dr. Marcelo Gomes de Queiroz**, que foram mais do que professores, foram uma lição de vida! Certamente ambos se preocupam com seus



alunos mesmo fora da sala de aula.

# Referências Bibliográficas

- [1] Just M. A. and Carpenter P. A. Eye fixations and cognitive processes. *Cognitive Psychology* 8 441-480, 1976.
- [2] Salles J. F. and Parente M. A. M. P. Cognitive processes involved in children's word reading: relations with reading comprehension and reading time. *Psicol. Reflex. Crit., Porto Alegre*, v. 15, n. 2, 2002.
- [3] Morimoto C. H and Mimica M. R. M. Eye gaze tracking techniques for interactive applications. *Computer Vision and Image Understanding* 98, 2005.
- [4] Nielsen J. Noncommand user interfaces. comm. *ACM* 36 (4) 822-839. Available at <http://www.useit.com/papers/noncommand.html>, 1993.
- [5] Paulson E. J. and Goodman K. S. Influential studies in eye-movement research. 1999-2000.
- [6] Jacob R. J. K. The use of eye movements in human computer interaction techniques: what you look is what you get. *ACM Transactions on Information Systems* 9(3), 152-169, 1991.
- [7] Jacob R. J. K. Eye movement-based human computer interaction techniques: Toward noncommand interfaces. *Advances in Human Computer Interaction*, 1993.
- [8] K. Rayner. Eye movements in reading and information processing: 20 years of research. *Psychology Bulletins*, 1998.

- [9] D.D. Salvucci and J. R. Anderson. Automated eye-movement protocol analysis. *Human Interaction*, 2001.
- [10] N. A. Stillings. Cognitive psychology: The architecture of the mind. In N. A. Stillings, M. H. Feinstein, J. L. Garfield, E. L. Rissland, D. A. Rosenbaum, S. E. Weisler, and L. Baker-Ward, editors, *Cognitive Science: An Introduction*, pages 17–63. MIT Press, Cambridge, MA, 1987.
- [11] David J. Ward and David J. C. MacKay. Artificial intelligence: Fast hands-free writing by gaze direction. *Nature*, 2002.
- [12] S. Morimoto C. e Ihde S. Zhai. Manual and gaze input cascade (magic) pointing. *CHI'99*, 2462013253, 1999. <http://www.almaden.ibm.com/u/zhai/papers/magic/magic.pdf>.