UNIVERSIDADE FEDERAL FLUMINENSE
INSTITUTO DE COMPUTAÇÃO
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

EDUARDO DE SALES CAVALCANTI

BIBLIOTECA DE RECONHECIMENTO DE MARCADORES CROMÁTICOS PARA AMBIENTES INTERATIVOS

Niterói
2010

EDUARDO DE SALES CAVALCANTI

BIBLIOTECA DE RECONHECIMENTO DE MARCADORES CROMÁTICOS PARA AMBIENTES INTERATIVOS

Monografia apresentada ao Departamento de Ciência da Computação da Universidade Federal Fluminense como parte dos requisitos para obtenção do Grau de Bacharel em Ciência da Computação

Orientador: Prof D. Sc. ESTEBAN WALTER GOLZALEZ CLUA

Niterói

2010

EDUARDO DE SALES CAVALCANTI

BIBLIOTECA DE RECONHECIMENTO DE MARCADORES CROMÁTICOS PARA AMBIENTES INTERATIVOS

Monografia apresentada ao Departamento de Ciência da Computação da Universidade Federal Fluminense como parte dos requisitos para obtenção do Grau de Bacharel em Ciência da Computação

Aprovado em 08 de dezembro de 2010.

BANCA EXAMINADORA

_____

Prof D. Sc. ESTEBAN WALTER GOLZALEZ CLUA

Orientador

UFF

_____

Prof D. Sc. CRISTINA NADER VASCONCELOS

UFF

_____

Prof D. Sc. DANIELA GORSKI TREVISAN

UFF

Niterói

2010

RESUMO

Com o crescimento da tecnologia e o constante aprimoramento de usuários de computador no que diz respeito ao seu manuseio, torna-se inerente a necessidade de serem exploradas novas metodologias de interação, mais naturais e eficientes, entre homens e máquinas. Com isso, o estudo sobre interação humano-computador (IHC) vem ganhando uma projeção há tempos em segundo plano. Com o propósito de contribuir para a evolução desta ciência, a proposta deste projeto é a criação de uma biblioteca de reconhecimento de gestos, de simples implementação e baixo custo computacional, baseada em fortes conceitos de IHC e Engenharia de Software, que possibilite uma navegação mais rica através de interfaces e ambientes virtuais de Rich Internet Applications (RIA) a partir da detecção e rastreamento de objetos do mundo real.

Palavras-chave: Reconhecimento de gestos, detecção de objetos, visão computacional, interação humano-computador, aplicações ricas, usabilidade, acessibilidade.

ABSTRACT

With the technology's growth and the constant improvements of computer using, it is inherent the necessity of exploring new methods of interaction, more natural and efficient, between men and machines. Thus, the study of human-computer interaction is returning to increase its visibility after a long time in background. Aiming to contribute to this science evolution, the purpose of this project is to create a gesture recognition library with simple implementation and low processing cost, based on strong concepts about HCI and Software Engineering, using objects in real-world to enable a richer navigation through interfaces and virtual environments on RIA applications from its detection and tracking.

Keywords: Gesture recognition, object tracking, computer vision, human-computer interaction, rich applications, usability, accessibility.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

IHC	Interação Humano-Computador
GUI	Graphical User Interface
WYSIWYG	What You See Is What You Get
CAD	Computer Aided Design
RIA	Rich Internet Application
AS3	ActionScript 3
HMM	Hidden Markov Model
AR	Augmented Reality
XML	Extensible Markup Language
API	Application Programming Interface

SUMÁRIO

1. INTRODUÇÃO

Diariamente vem ao conhecimento geral a busca por formas mais eficazes de interação entre homem e máquina. Desde o estudo de cores de uma determinada janela para minimizar o cansaço durante um trabalho repetitivo no computador até o desenvolvimento de novas tecnologias para tornar a imersão das pessoas em filmes e jogos quase completa, os resultados da pesquisa e experiências em IHC têm grande influência na atual maneira como percebemos o mundo.

Em particular, as interfaces baseadas no reconhecimento de gestos vêm sendo amplamente exploradas, com aplicações em inúmeras áreas, como na indústria de videogames com a criação de dispositivos capazes de distinguir a posição do jogador, na navegação em ambientes tridimensionais de realidade virtual e até no auxílio a portadores de necessidades especiais, como deficientes auditivos, com a tradução em tempo real da linguagem de sinais e deficientes com paralisia cerebral, com a interpretação de expressões faciais usadas na composição de músicas.

Este trabalho tem por objetivo a construção de uma biblioteca de captura e reconhecimento de movimentos através de marcadores contribuindo para o aumento do interesse nesse campo de estudo. Para validar seu funcionamento serão apresentadas algumas aplicações simples utilizando o conceito de visão estereoscópica, mouse 3D e interface colaborativa, exemplificando a utilização da biblioteca em ambientes tridimensionais.

Para esclarecer os conceitos empregados no desenvolvimento do presente trabalho, o segundo capítulo aprofundará as definições sobre reconhecimento de gesto e sua contextualização com o estudo de interfaces.

No terceiro capítulo será apresentada a arquitetura proposta na elaboração do projeto sob a visão de seus principais interessados, usuários da biblioteca no que tange o desenvolvimento de aplicativos, e usuários destas aplicações.

No capítulo seguinte é explicitada a implementação da biblioteca, através da explicação detalhada dos algoritmos que a compõem.

No quinto capítulo, são apresentados alguns aplicativos que utilizam os recursos da biblioteca em questão, demonstrando seu funcionamento e a possibilidade de aplicação em diversas áreas.

Por fim, serão avaliados os resultados do uso da biblioteca, e serão apresentadas sugestões para sua melhoria em trabalhos futuros.

2. INTERFACES BASEADAS EM VISÃO COMPUTACIONAL

Este capítulo apresenta um breve histórico que contextualiza a evolução das interfaces e a busca por formas de interação mais amplas entre homens e máquinas, caracterizando o reconhecimento de gestos como uma opção significativa. Para tal também são introduzidos os conceitos de gestos e visão computacional.

2.1. EVOLUÇÃO DAS INTERFACES

O primeiro marco na construção de interfaces como as vistas atualmente foi alcançado com o projeto SketchPad desenvolvido por Sutherland (1963, p. 329-346). Apesar de sua distribuição limitada a algumas máquinas do MIT Lincoln Laboratory, as idéias que fomentaram sua implementação contribuíram para a introdução do conceito de interface gráfica (GUI) no meio computacional. A interação do sistema era baseada na manipulação direta de objetos gráficos através de uma caneta luminosa que permitia ao usuário tocá-los e manipulá-los, o que convencionaria mais tarde a ação de clique para fins de seleção, arraste e ativação de elementos.

Segundo Myers (1998, p. 44-54), o conceito “WYSIWYG” (What You See Is What You Get, ou, O que você vê é o que você toca), que forma a base da construção das interfaces atuais, é diretamente provindo de pesquisas no campo de manipulação direta de objetos, com o surgimento de aplicações como o Bravo e o Draw, um editor de texto e um programa de desenho, respectivamente.

Alguns anos mais tarde o surgimento do mouse popularizaria o uso de dispositivos de ponteiro bidimensional, como uma opção mais barata à caneta luminosa, com a comercialização do Xerox Star em 1984 (MYERS, 1984, p. 13-23), que também tornou mais acessível o conceito de múltiplas janelas nos sistemas operacionais.

Outros tipos de aplicativos que formam a base do que é conhecido sobre interfaces hoje, são os primeiros programas que utilizavam interfaces 3D e reconhecimento de gestos, a exemplo dos sistemas de desenho auxiliado por computador (CAD). Sistemas de suporte a trabalho cooperativo e, sistemas de realidade virtual e realidade aumentada, também são oriundos da década de 70 (MYERS, 1998, p.44-54).

Muito se discute sobre o grande salto que o universo computacional deu nas últimas décadas. Tecnologias cada vez mais avançadas permitem dia após dia um aumento exponencial da capacidade de processamento dos processadores e placas gráficas atuais. No entanto o ritmo da evolução dos aspectos funcionais das interfaces, a maneira como são vistas e como funciona sua interação, não acompanhou a velocidade do crescimento tecnológico, o que pode ser facilmente observado quando se reflete sobre a maneira como o ser humano permanece usando computadores diariamente. Ainda hoje, aproximadamente 40 anos após sua concepção, o mouse ainda é o principal instrumento de interação entre homem e máquina.

2.2. CONCEITOS DE VISÃO COMPUTACIONAL

Existem diversas maneiras de detectar, rastrear e reconhecer os movimentos de determinadas partes do corpo. Sistemas que são capazes de coletar esses dados e utilizá-los como entrada são conhecidos como sistemas baseados em Visão Computacional. Sistemas desse tipo capturam as informações associadas aos gestos sem a necessidade de dispositivos físicos, da mesma forma como os humanos interagem entre si, vendo e ouvindo, utilizando-se de câmeras e microfones, respectivamente (BÉRARD, 2001). Normalmente utiliza-se a mão, por sua função intuitiva nas ações envolvendo objetos, e a cabeça, já que a visão também é orientada através de sua posição, como instrumentos de interação, como exemplificado na Figura 1.

Figura 1 – Modelos de interação em interfaces tradicionais e digitais (BÉRARD, 1999).

Para que a detecção de um determinado objeto seja possível, as imagens capturadas são submetidas a técnicas de processamento de imagens cujos objetivos são destacar da forma mais conveniente o objeto alvo do restante da cena. Um fluxo natural de processamento de imagens inicia com a recepção e armazenamento das imagens da câmera, seguida de um pré-processamento dessa imagem, que consiste em melhorar sua qualidade para facilitar a análise realizada na etapa de processamento, quando os dados de interesse da imagem são recolhidos, e por fim, interpretam-se esses dados de acordo com o propósito da aplicação. Quando o resultado desta interpretação é concluído, a ação referente a esse gesto é exibida ao usuário e esse ciclo pode reiniciar com a captura da próxima imagem da câmera, e assim por diante, como ilustra a Figura 2.

Figura 2 – Ciclo de uso de tecnologia de detecção de objetos

Para cada tipo de aplicação técnicas diferentes de detecção de objetos podem ser empregadas. Alguns tipos de abordagens comuns são a segmentação e a subtração de fundo.

A segmentação consiste na extração de um objeto da imagem a partir de uma ou mais características predominantes. Propriedades como a cor, a geometria e a textura, são exemplos de atributos que podem ser usados no processo de segmentação. Todos os algoritmos de segmentação convergem em duas decisões importantes, os critérios significativos para um bom particionamento da imagem e o método usado para alcançá-los (SHI; MALIK, 2000, p. 888-905). Segundo Britto (1997), particularmente a cor é um atributo bastante usado nos algoritmos de segmentação por seu caráter discriminatório, mas apesar de sua popularidade, as cores de uma imagem de câmera são sempre suscetíveis a mudanças de iluminação,

dificultando o processo de detecção. Algumas técnicas conhecidas de detecção de objetos baseados em segmentação são o Mean-Shift (COMANICIU; MEER, 2002, p.603-619), que usa a comparação de imagens consecutivas e os picos dos histogramas destas imagens para determinar um mapa de confiança, reconhecendo assim o objeto, e o método de Cortes Normalizados (SHI; MALIK, 2000, p. 888-905), que propõe a segmentação de imagens a partir do particionamento de grafos.

Técnicas baseadas em subtração de fundo comparam a imagem capturada a cada *frame* da câmera com um modelo de fundo conhecido previamente (FERREIRA, 2007, p. 10). Uma mudança significativa nos *pixels* da imagem provavelmente denota a movimentação de algum objeto naquela região. Dois exemplos de abordagens de subtração de fundo são a Mistura de Gaussianas (STAUFFER; GRIMSON, 2000, p.747-767), baseada na persistência e variância de gaussianas que representam cada *pixel*, e Eigenbackground (OLIVER; ROSARIO; PENTLAND, 2000, p. 831-843), que determina um novo modelo espacial que descreve as variações na aparência da imagem, como diferença de iluminação, por exemplo, para facilitar a detecção do objeto alvo.

O próximo passo é analisar se houve de fato alterações em alguma característica deste objeto, como a alteração da posição dos *pixels* referentes ao mesmo, analisado a partir de um método de subtração ou ainda da extração de um padrão de posição de um objeto usando modelos ocultos de Markov (HMM), como proposto por Eickeler (1997), e traduzir o novo estado do objeto em uma ação correspondente esperada pelo dispositivo funcional, um componente de interface de um aplicativo, por exemplo.

A representação dos objetos depende do interesse da análise. Por exemplo, em um sistema de controle de tráfego seria interessante contabilizar a quantidade de veículos que atravessam um determinado trecho em horários diferentes do dia. Para tal poderia ser feito o reconhecimento do contorno de cada veículo, o que provavelmente possibilitaria uma análise mais completa a respeito dos tipos de veículos que trafegam nessa rodovia, mas se apenas um retângulo contornasse as cores desses veículos, e incrementasse um contador a cada novo retângulo criado, o sistema já teria as informações quantitativas desejadas.

Algumas restrições apresentadas no processo de visão computacional são a incapacidade de fazer um processamento em tempo real de todas as propriedades dos gestos mapeados, a necessidade do uso de equipamentos caros (câmeras infravermelhas, câmeras 3D, câmeras sensíveis à temperatura) e de difícil acesso no Brasil, restrições relacionadas à iluminação do ambiente, restrições relacionadas a condições do fundo da imagem a ser processada, uso de

objetos que facilitem a detecção dos gestos, como luvas coloridas e outros tipos de marcadores, e restrições quanto à velocidade máxima do gesto.

Do ponto de vista computacional, as restrições mencionadas acima são apenas dificuldades no desafio que é a criação de algoritmos eficazes com resultados satisfatórios de reconhecimento de gestos, entretanto, é justamente uma dessas restrições que pode prevenir problemas possivelmente mais graves, como a diminuição da acessibilidade e da usabilidade aos usuários.

O uso de marcadores pode ajudar a solucionar esses problemas quando usados como instrumentos de navegação por tornar funcionamento de algoritmos de reconhecimento extremamente mais simples, bastando que alguma das características físicas do marcador se distinga facilmente da cena. Por outro lado, o reconhecimento de gestos da mão, que são encontrados mais freqüência na literatura, requer algoritmos mais complexos e máquinas mais poderosas. Um algoritmo de reconhecimento com uma latência alta, ou seja, com tempo de resposta aos movimentos do usuário lenta, claramente afeta a usabilidade do sistema, e o uso do mesmo sistema em um ambiente sem iluminação provavelmente seria inviável.

É com base nessa idéia, de diminuir a complexidade dos algoritmos empregados no reconhecimento, e conseqüentemente seu tempo de processamento, de forma a não diminuir significativamente a qualidade dos resultados alcançados, e em questões básicas de IHC, que este projeto é proposto.

2.3. A INFLUÊNCIA DE GESTOS COMO FORMA DE INTERAÇÃO

O ser humano se utiliza de gestos para interagir com os outros naturalmente. Seja para sinalizar algo, para externar seus sentimentos num dado momento, ou para facilitar a compreensão em um diálogo. Em verdade toda comunicação entre duas pessoas está intrinsecamente relacionada a articulações gestuais. É com base nessa habilidade inata que usuários têm em se comunicar a partir de movimentos que surge o interesse de replicar esse modelo para o meio computacional.

Para definir o que de fato significa um gesto seria necessário listar tudo que é possível exprimir através dele: sensações, indicações, opiniões, idéias, e todas as suas demais interpretações. De maneira a simplificar sua definição e torná-la mais adequada ao campo computacional, somente serão considerados os aspectos perceptíveis à visão. Kurtenbach (1990) define os gestos como sendo movimentos do corpo que contém informação, uma definição simples que sintetiza os atributos necessários para este estudo.

Basicamente, a todo gesto estão associadas características como posição, orientação e alterações físicas na parte do corpo que está executando aquele movimento. Se a maneira como o homem usa as mãos para realizar qualquer tarefa for analisada ao longo de um espaço de tempo definido fica claro que a cada novo instante estas três variáveis, por assim dizer, têm seus valores alterados, ainda que com pequenas variações.

Vale ressaltar que, para a análise deste trabalho, ações como pressionar um mouse ou qualquer outro dispositivo não devem consideradas como gestos, já que apesar de haver a movimentação do dedo até o objeto para que a ação seja possível, o ato de pressionar em si não é significativo para o computador enquanto movimento de um corpo.

2.4. INTERFACES BASEADAS EM GESTOS

Um tipo de aplicação muito encontrado na literatura quando se fala em modelos de navegação baseados em gestos são os sistemas de realidade virtual. Para sistemas dessa categoria esse modelo de interação é interessante, pois existe uma associação bastante clara com a forma como é realizado o manuseio de objetos no mundo real, sendo assim, uma forma mais natural de navegação seria a que mais se aproxima da realidade.

Apesar de o ser humano trabalhar com grande facilidade no mundo real, manipulando objetos de formas variadas, não é simples para um programa traduzir essa simplicidade para o mundo computacional. Historicamente, sistemas de navegação em ambientes tridimensionais baseados em gestos fazem uso freqüente de complexos arranjos de câmeras como o Media Room (BOLT, 1984) ou de luvas especiais contendo sensores de movimento, como a CyberGlove (KRAMER, 1989) para promover a navegabilidade de seus sistemas. Mais recentemente a indústria dos videogames percebeu a grande valia de sistemas de navegação mais naturais para seus consoles. Exemplos como o WiiMote, o controle do Nintendo Wii baseada na tecnologia de acelerômetros, e mais recentemente o Kinect da Microsoft, que é capaz de determinar a profundidade de um ambiente através do mapeamento da reflexão de raios infravermelhos, mostram que apesar de solucionarem o problema de rastreamento de gestos de maneira extremamente eficiente e inovadora ainda precisam de dispositivos complexos para sua implementação. Um ótimo exemplo de aplicação de reconhecimento de gestos é o projeto G-Stalt de Zigelbaum (2010), que permite aos usuários manipular um ambiente tridimensional repleto de vídeos.

Outra questão importante a ser levada em consideração é como o advento da internet altera a forma com que as pessoas interagem. O que outrora era inimaginável, como o envio e compartilhamento de dados através de grandes distâncias em tempos curtos, ou ainda mais

importantes, a centralização desses dados em um ambiente acessível a um número imenso de pessoas, agora se torna parte do cotidiano dos usuários de computador. Com isso é necessário o exercício de um novo conceito de interfaces, que possibilitem o manuseio desses recursos da mesma maneira como são acessados, por muitas pessoas simultaneamente, são as chamadas interfaces colaborativas. Apesar de extremamente importante ao estudo de IHC e o modo como essa ciência avança, neste estudo o objeto de interesse das interfaces colaborativas é a sua verossimilhança com a realidade. No mundo real o tipo de interação mais comum é a interação entre pessoas. Seja concorrendo ou se ajudando, a questão é que essa interação é necessária para obtenção de resultados melhores nas atividades requisitadas, e a transposição dessa possibilidade para o universo computacional se torna fundamental.

Existem ainda interações que usam objetos criados virtualmente como suporte ao mundo real. São as aplicações baseadas em realidade aumentada (AR), que lidam com a interatividade de forma conceitualmente oposta à realidade virtual. Enquanto os ambientes virtuais levam os usuários para dentro de mundo virtual, a realidade aumentada traz a semântica virtual para o mundo real, como explica Azuma (1997), potencializando o senso de imersão.

Para tal, interfaces em AR devem ser projetadas de alguma forma no mundo real. Comumente são usados dispositivos com câmeras montados na cabeça ou, mais recentemente, projetores portáteis, que criam a sobreposição necessária entre os mundos de forma aos objetos virtuais se tornarem tangíveis como em um projeto em desenvolvimento no MIT chamado SixthSense (MINSTRY, 2009), que usa um dispositivo simples, anexando um projetor portátil, uma *webcam* e um dispositivo móvel de processamento de imagens, para projetar objetos interativos em ambientes reais, e torna-os manipuláveis através de reconhecimento de gestos.

3. ELABORAÇÃO DO PROJETO

Antes de uma análise mais minuciosa a respeito dos aspectos arquiteturais e modelagem é importante ressaltar o real propósito deste projeto: a criação de uma biblioteca simples de reconhecimento de gestos capaz de usar qualquer objeto do mundo real como dispositivo físico de entrada de comandos a aplicações ricas de internet (RIA) e sites. Para a construção da mesma foi usada a linguagem ActionScript3 (AS3), amplamente usada na construção de RIAs, para a plataforma Flash, podendo assim rodar em qualquer navegador com o *plug-in* Flash Player, em sua versão 9 ou superior, instalado em qualquer sistema operacional, fornecendo assim grande portabilidade e escalabilidade no uso da biblioteca.

Para melhor compreensão do funcionamento da biblioteca deve ser analisada a sua utilização no prisma dos dois *stakeholders* relacionados ao projeto: os desenvolvedores que usarão a biblioteca em seus sistemas e os usuários desses sistemas.

3.1. VISÃO DO DESENVOLVEDOR

A organização da arquitetura adotada, apresentada na Figura 3 possibilita que desenvolvedores utilizem a biblioteca de maneira simples e independente dos demais módulos da sua aplicação.

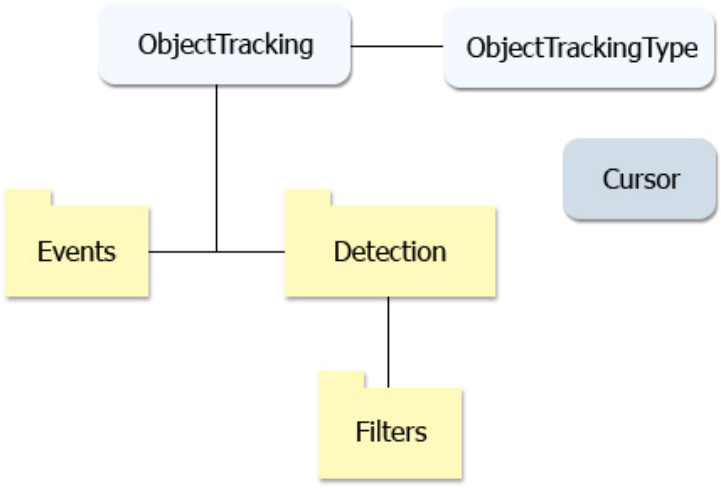

Figura 3 – Visão geral da arquitetura

A classe ObjectTracking, explicitada na Figura 4, é a classe de interface entre o desenvolvedor e a biblioteca. Nela estão contidos os elementos necessários para que um usuário final possa interagir com a aplicação, como a captura das imagens da *webcam*, vídeos de controle e exibição, e métodos de controle da biblioteca, sendo assim, instanciando esta classe o desenvolvedor terá um acesso transparente às funcionalidades disponíveis.

Figura 4 – Classe ObjectTracking

A classe Cursor, mostrada com mais detalhes na Figura 5, responsável por fornecer *feedbacks* aos usuários finais do sistema. Ela renderiza na tela a área correspondente ao objeto selecionado, e contem métodos que fornecem a sua posição no eixo (x, y), bem como sua profundidade relativa, referente ao eixo z.

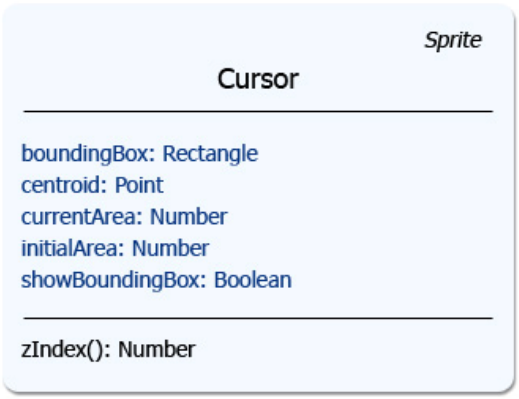

Figura 5 – Classe Cursor

O módulo que contém os métodos de detecção de objetos, bem como os filtros usados no processamento das imagens, é independente da classe ObjectTracking, possibilitando futuras extensões e aprimoramentos nos métodos usados facilmente. Para tornar mais simples a inclusão de novos métodos esse módulo se comunica com a classe principal a partir de disparo de eventos, que será explicado com mais detalhes em seguida.

A fim de exemplificar a facilidade da adição de um novo método de detecção, suponha que um desenvolvedor deseje criar uma classe que implemente a técnica Mean-Shift (COMANICIU; MEER, 2002, p.603-619). Para que a integração seja feita, basta essa classe estender da classe EventDispatcher, que permite o disparo de eventos aos objetos, e implementar a interface IDetectionMethod, que assegura que os métodos Start, responsável pela inicialização das estruturas da técnica em questão e pela sinalização de que a classe está pronta para ser usada, Detect, responsável pela sinalização de que uma instância de detecção foi iniciada, e Execute, onde a técnica deve ser de fato codificada, serão implementados.

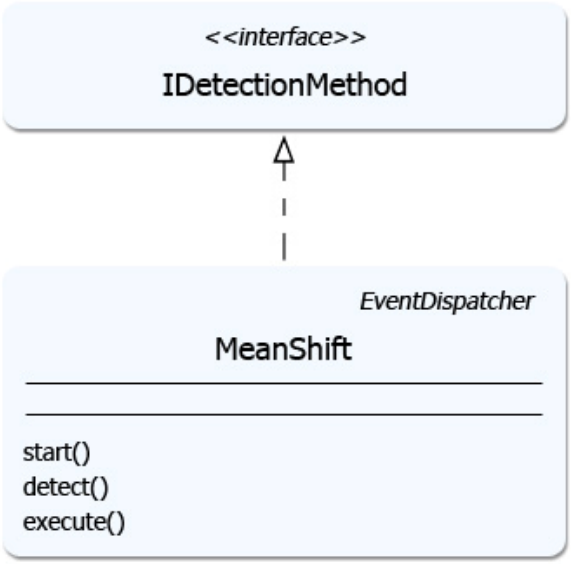

Figura 6 – Integração de uma classe com a interface IDetecionMethod

Detalhes de implementação das classes apresentadas acima, bem como seu uso, serão explicitados no capítulo 4 e 5.

O módulo de detecção, onde estão presentes as classes responsáveis pelo reconhecimento dos objetos, se comunica com a classe ObjectTracking via troca de eventos. Isso possibilita maior flexibilidade e o encapsulamento da biblioteca devido à independência de conhecimento de questões de implementação das classes internas para seu uso. Quando alguma classe de detecção é instanciada o objeto que a mantém é configurado de maneira a esperar por três eventos distintos, referentes à inicialização da classe de detecção, ao início do processo de detecção e ao término do processo. Estes eventos agem da seguinte forma:

- TrackingEvent.DETECTION_INIT: Este evento é disparado quando o construtor da classe de detecção termina de configurar todas as estruturas necessárias para a sua execução. Quando a ObjectTracking recebe este evento pode adicionar ao Stage, onde os objetos são renderizados no Flash, uma cópia da imagem durante o processo de análise, com a certeza de que estas estruturas já estão plenamente criadas. Além disso, a função que é chamada a partir do evento recebido pode solicitar o início do processo de detecção.
- TrackingEvent.DETECTION_START: Este evento é disparado quando a técnica de detecção é iniciada. A principal importância deste evento é sinalizar à classe ObjectTracking que o *buffer* de imagens não deve ser atualizado até que a detecção esteja completa.

- TrackingEvent.DETECTION_COMPLETE: Este evento é disparado quando o processo de análise da imagem e detecção do objeto é terminado. A classe ObjectTracking pode então atualizar todos os cursores referentes aos objetos encontrados, e iniciar o próximo ciclo de detecção com próxima imagem do *buffer* .

Esses três eventos são usados apenas internamente às classes da biblioteca, sendo assim, são manipuláveis apenas por desenvolvedores de novos métodos de detecção. No entanto, foram criados alguns eventos auxiliares que são disparados a partir da classe ObjectTracking, para refletir a execução dos eventos internos, importantes para o desenvolvedor de aplicações que utilizem a biblioteca, são eles: o ObjectTrackingEvent.INIT e o ObjectTrackingEvent.COMPLETE, referentes à inicialização da detecção e ao término do processo, respectivamente, e o ObjectTrackingEvent.CAMERA_ACTIVE, referente à inicialização da câmera.

3.2. VISÃO DO USUÁRIO

Um usuário que deseje executar um sistema RIA com esta biblioteca precisará ter o *plug-in* Flash Player, em sua versão 9.0 ou superior, instalado em seu navegador. O *plug-in* está disponível para download na página do fabricante. As configurações de máquina para sua execução ficarão dependentes da aplicação implementada.

Durante a execução de uma aplicação, o usuário apenas precisará exibir de um objeto qualquer, preferencialmente com uma cor destoante das demais cores da cena, em frente a uma *webcam*. A cor do objeto será capturada e servirá de referência para os métodos de detecção. A biblioteca fornece funcionalidades para que sejam exibidos *feedbacks* em relação à cor selecionada e posição atual do objeto, como será mostrado através de uma aplicação desenvolvida para projeto no capítulo 5. Uma possível interação entre um usuário, um aplicativo e a biblioteca é exemplificado na Figura 7.

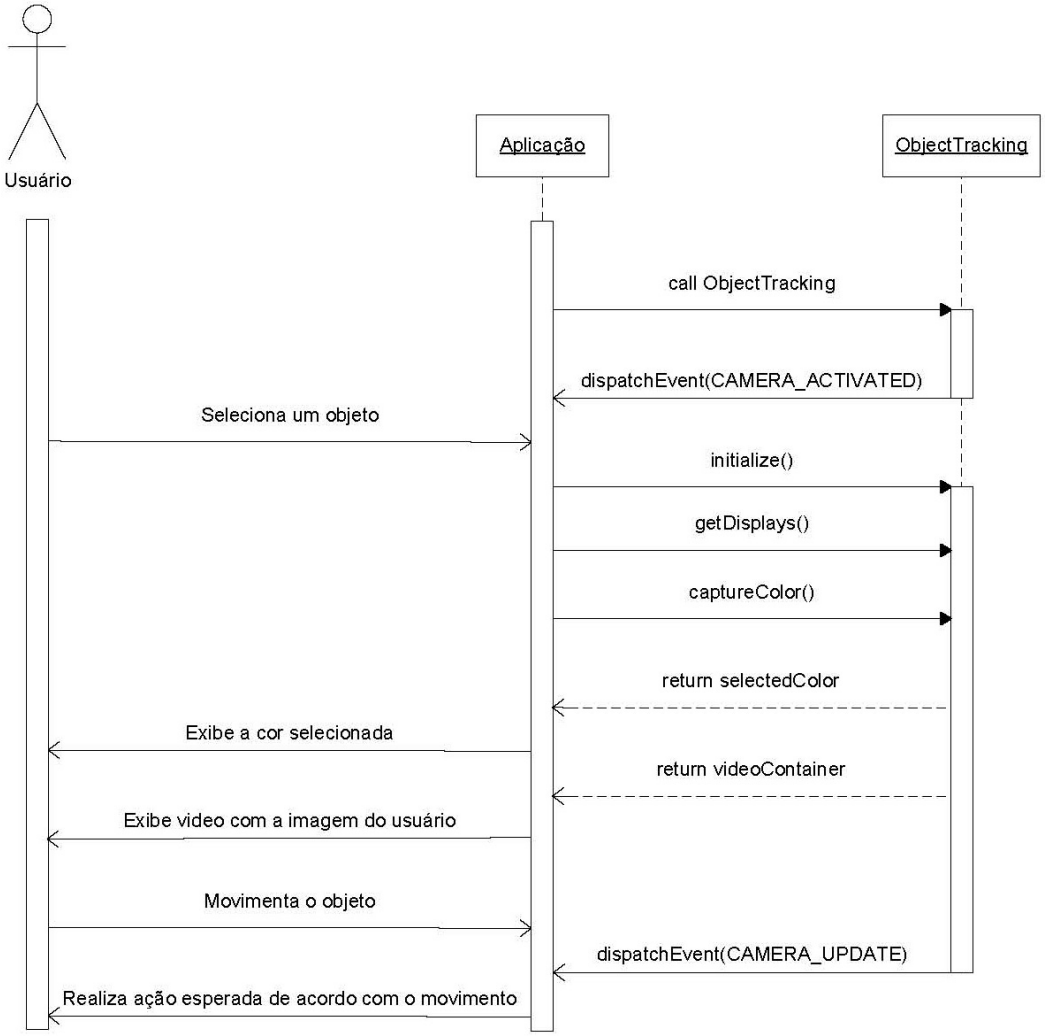

Figura 7 – Diagrama de sequência de uma possível interação do sistema

4.

4 IMPLEMENTAÇÃO

A fim de contextualizar de forma mais apropriada à construção das classes da biblioteca, este capítulo será organizado de maneira seqüencial em relação as suas atividades.

4.1. CONFIGURAÇÃO DO AMBIENTE

Como foi dito anteriormente, para iniciar a utilização da biblioteca a classe ObjectTracking deve ser instanciada. O construtor desta classe criará então os recursos necessários para a interação possa acontecer.

Em um sistema baseado em Visão Computacional os elementos básicos de manipulação são as imagens capturadas a cada instante por uma determinada câmera, sendo assim, primeiramente é selecionada a *webcam* que servirá de input para os gestos. Atualmente a biblioteca fornece suporte à utilização de apenas uma câmera, apesar de existirem métodos de reconhecimento de gestos e padrões baseados em múltiplas câmeras, como o SPfinder (AZARBAYJANI, 1996), uma abordagem que obtém modelos tridimensionais a partir do uso de câmeras estéreo.

A partir do momento que a câmera é detectada dois vídeos são criados: um vídeo de controle e outro de visualização. O vídeo de controle é criado para que o processamento das imagens aconteça a partir de um *buffer* de vídeo independente daquele que será usado como interface para o usuário. Possíveis aplicações de filtros e transformações sobre as imagens do *buffer* do vídeo exibido, bem como no próprio vídeo, para auxiliar os métodos de detecção poderiam prejudicar o desempenho do *feedback* ao usuário. Além disso, uma pequena transformação de *flip* horizontal é efetuada em ambos os vídeos, para eliminar o efeito de enantiomorfismo, a simetria gerada pela captura da imagem na câmera. Para concluir a configuração, a câmera é configurada para atualizar o vídeo a uma taxa de 30 frames por

segundo, mas esse valor pode ser alterado pelo desenvolvedor se desejado. O Quadro 1 mostra este processo.

```
//Seleção da câmera
var cam:Camera = Camera.getCamera();

if (cam != null)
{
	/**
	 * Inicialização da câmera:
	 * O evento ActivityEvent.ACTIVITY é capturado quando a câmera está pronta para ser usada.
	 * Configuração do tamanho das imagens capturadas, frameRate e qualidade da captura
	 **/
	cam.addEventListener(ActivityEvent.ACTIVITY, cameraActivityHandler);
	cam.setMode(_width, _height, 30, false);
	cam.setQuality(0, 70);


      //Criação do video de controle e flip horizontal
	_controlVideo = new Video(_width, _height);
	_controlVideo.attachCamera(cam);
	_controlVideo.x = _width;
	_controlVideo.scaleX = -1;

	/**
	 * Adição do vídeo de controle no container de vídeos, usado para facilitar a manipulação da interface que contém os vídeos.
	 **/
	_videoContainer.addChild(_controlVideo);

	//Caso o desenvolvedor queira, o vídeo de display para o usuário pode não ser exibido.
	if (_showCamera)
	{
		//Criação do video de display, usado como interface do usuário
	_displayVideo = new Video(_width, _height);
		_displayVideo.attachCamera(cam);
		_displayVideo.x = _width;
		_displayVideo.scaleX = -1;

		_videoContainer.addChild(_displayVideo);

		_trackingVideoArray.push( { camera:cam, showVideo:_displayVideo, controlVideo:_controlVideo } );
	}
	else
	{
		_controlVideo.alpha = 0;
		_trackingVideoArray.push( { camera:cam, showVideo:null, controlVideo:_controlVideo } );
	}
}
```

Quadro 1 - Código de configuração do ambiente do ObjectTracking

4.2. PROCESSAMENTO DO VIDEO

Esta é a etapa mais importante do projeto. É o processamento do *buffer* do vídeo de controle que possibilita o reconhecimento dos objetos.

A idéia inicial do desenvolvimento desta biblioteca era de implementar um método que fosse capaz de reconhecer objetos de fácil compreensão e baixo custo computacional. O resultado dessa pesquisa é a implementação de duas técnicas distintas de detecção de objetos representadas pelas classes SingleSegmentation e BlobDetection, detalhadas em seguida.

Para processar as imagens do *stream* da *webcam*, o conteúdo destes frames é associado a objetos do tipo BitmapData, um vetor de dados referentes a *pixels*. Cada *pixel* é representado por um inteiro de 32 bits, divididos em 8 bits referentes à transparência e aos três canais de cor (ARGB). A classe BitmapData também implementa alguns métodos que possibilitam uma manipulação direta dos *pixels*.

4.2.1. SINGLE SEGMENTATION

O objetivo desta classe é fornecer ao desenvolvedor de uma aplicação o uso de cores como forma de interação. A idéia inicial da construção deste algoritmo é possibilitar a utilização de qualquer objeto como instrumento de interação, já que, obviamente, todos os objetos têm uma cor. Outro motivo é que, como exemplificado no capítulo 3, existem diversos algoritmos de detecção de padrões através de cores consagrados, entretanto, implementações simples e de baixo custo computacional são dificilmente encontradas. Tendo em vista que os algoritmos presentes nesta biblioteca devem processar uma sequência de imagens capturadas em uma freqüência relativamente alta, o resultado da análise do *buffer* deve ser dado rapidamente.

O algoritmo proposto faz uma série de simplificações em relação aos métodos clássicos, desde a escolha de filtros mais simples usados na etapa de pré-processamento, até os critérios adotados para a extração das informações de cor do objeto. A idéia básica do algoritmo é tornar as imagens do vídeo mais homogêneas, reduzindo o seu número de cores, e detectar todos os *pixels* de uma determinada cor presentes nelas.

Essa abordagem simplista traz algumas restrições em detrimento ao tempo de resposta alcançado, como a necessidade da seleção de uma cor de destaque na cena, já que todos os *pixels* dessa cor serão caracterizados como um único objeto, e a necessidade de o usuário estar em um ambiente com pouca variação de iluminação, que alteraria a cor desse objeto.

Para reduzir a gama de cores da imagem de entrada do algoritmo, é criada uma nova paleta de cores que reproduz o efeito de posterização, comumente percebido em impressões de baixa qualidade. A simulação desse efeito é feita deslocando-se os bits de cada componente do ARGB para esquerda, ignorando aqueles que extrapolarem o limite de 8 bits reservados a cada canal, e preenchendo o bits mais a direita com 0, perdendo assim

informações de cor dos bits que estouraram. Para melhor compreensão dos resultados desse filtro, a Figura 8 mostra um comparativo entre uma imagem capturada normalmente e outra após a transformação, bem como seus histogramas, e o Quadro 2 exibe sua codificação.

```
_redArray = [];
_greenArray = [];
_blueArray = [];
_alphaArray = [];

var div:Number = 256 / levels;

for (var i:uint = 0; i < 256; i++)
{
      var value:uint = Math.floor(i / div) * div;

      _alphaArray[i] = value << 24;
      _redArray[i] = value << 16;
      _greenArray[i] = value << 8;
      _blueArray[i] = value;
}
```

Quadro 2 - Código de criação de paleta de cores reduzida

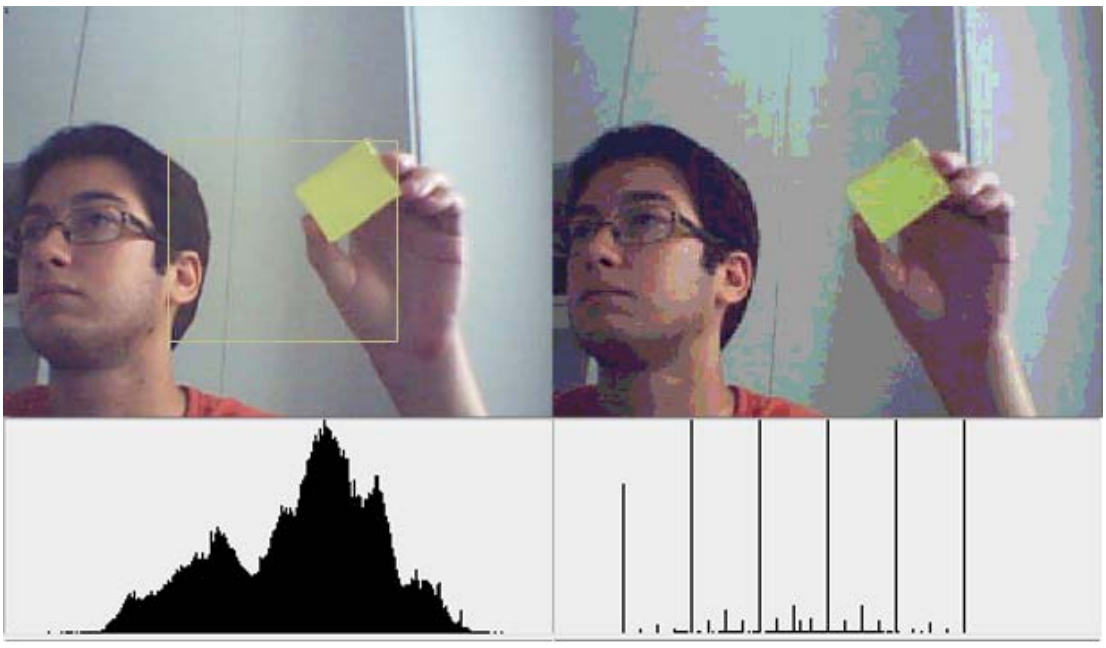

Figura 8 – Resultados da segmentação do amarelo sem o uso de filtros

Com a imagem mais planificada e a diminuição da variação de cores, e partindo do princípio que a cor escolhida é única na cena, torna-se trivial perceber que todos os *pixels* da cor escolhida fazem parte do objeto desejado. O método armazena a cor do *pixel* selecionado pelo usuário e cria um objeto do tipo Cursor a partir do retângulo que inscreve todos os *pixels* da cor armazenada.

Até então, estava sendo ignorada qualquer necessidade de pré-processamento, como aplicação de filtros e transformações nas imagens, e por seguinte garantindo uma taxa de resposta bastante eficiente, entretanto, a qualidade dos resultados não era a desejável. Durante a captura das imagens alguns ruídos adicionados pela câmera eram renderizados com a mesma cor selecionada dependendo da iluminação do ambiente. Nessa circunstância, o retângulo que serve de base para a construção do cursor era incoerente com o objeto que se desejava seguir.

A primeira solução encontrada foi aplicar um filtro de desfoque gaussiano, amplamente usado o processamento de imagens para redução de ruídos. O filtro consiste na aplicação de uma função estatística em x e y relativas a cada *pixel* da imagem, dada pela fórmula a seguir, onde σ é o desvio padrão da distribuição normal (VLIET; YOUNG; VERBEEK, 1998, p. 509-514).

$$G(x,y) = \frac{1}{\sqrt{2\pi\sigma}} e^{\frac{[(x]^2+y^2)}{\sigma^2}}$$

Na busca por algoritmos mais simples de desfoque, o que trouxe os resultados mais satisfatórios foi o BoxBlur, onde cada *pixel* na imagem resultante tem o valor igual à média dos valores dos *pixels* vizinhos, como mostrado na Figura 9. De acordo com a Adobe, pelo teorema do limite central pode-se demonstrar que aplicando-se três vezes este algoritmo o resultado final se aproxima bastante da distribuição gaussiana, com uma diferença de aproximadamente 3%.

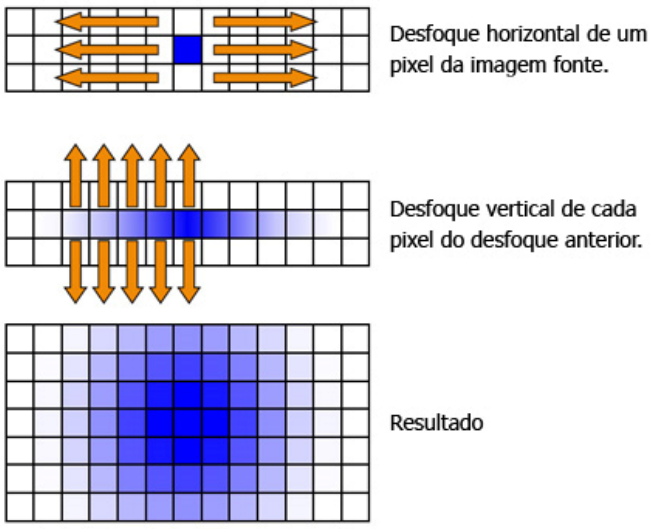

Figura 9 – Funcionamento do algoritmo BoxBlur

Com a adição deste desfoque durante a fase de pré-processamento foram atingidos os resultados mostrados a seguir, na Figura 10, e demonstrados nas aplicações apresentadas no capítulo 5.

Figura 10 – Segmentação com o uso do filtro de desfoque

4.2.2. BLOB DETECTION

Com motivação semelhante, mas de forma complementar à classe SimpleSegmentation, esta classe possibilita a detecção de objetos através do deslocamento dos *pixel* durante fluxo de imagens. Com a construção dessa classe a biblioteca passa a abranger uma lista mais extensa de aplicações de reconhecimento de gestos.

Este algoritmo se baseia na subtração de fundo para a percepção de movimentos em frente à câmera. Abordagens de subtração de fundo são formas de segmentação de objetos a partir da comparação entre a imagem que está sendo observada no momento com uma estimativa desta imagem caso esta não contenha o objeto de interesse. Ao ser feita a subtração dos dois planos a imagem resultante contém apenas o objeto em questão. Para tal, normalmente aplicativos que utilizem essa técnica precisam de certo treinamento, correspondente a captura da imagem que servirá de base de comparação, o *background*.

Apesar de menos determinístico, assim como na segmentação feita puramente através da cor, essa técnica apresenta também restrições em relação à iluminação do ambiente de uso. Em um ambiente escuro, se o usuário movimentar objetos também escuros, a percepção do movimento pode ficar comprometida pela falta de contraste entre a cena e o objeto. Em contrapartida, por realizar o reconhecimento através dos movimentos na cena, não importa se existem elementos de cores semelhantes no mesmo plano.

Para implementar o algoritmo são usados 3 objetos BitmapData que guardam as informações dos *pixels* relativas a cada operação realizada, sendo um relativo a imagem atual,

de primeiro plano, uma relativa a imagem anterior, que será usada como base de comparação no segundo plano, aproveitando o fluxo de imagens para dispensar a etapa de configuração do *background*, e o terceiro responsável por armazenar o resultado da subtração entre os dois anteriores. O Quadro 3 ilustra como é realizada esta subtração.

```
var gray:ColorMatrix = new ColorMatrix();
gray.adjustSaturation(0);
var grayscaleFilter = new ColorMatrixFilter(gray.matrix);

_done = _now.clone();
_done.draw(_old, new Matrix(), null, BlendMode.DIFFERENCE);

_done.applyFilter(_done, _done.rect, pt, grayscaleFilter);

_done.threshold(_done, _done.rect, pt, "<=", 0xFF111111,
FLOOD_FILL_COLOR, BASE_COLOR);
_done.threshold(_done, _done.rect, pt, "!=", FLOOD_FILL_COLOR,
BASE_COLOR, 0x00FFFFFF);

_done.applyFilter(_done, _done.rect, pt, new BlurFilter(10, 10, 1));


_out.draw(_done, new Matrix());
_old = _now.clone();
```

Quadro 3 - Rotina de subtração de fundo

Primeiramente é feita uma cópia da imagem atual do *buffer* para o BitmapData de resultado, preservando a integridade da imagem original. Em seguida é feita a subtração do clone da imagem atual e a imagem base através da função draw da classe BitmapData, que desenha o conteúdo deste objeto em um objeto IBitmapDrawable, usando o modo de sobreposição por diferença, que compara as cores constituintes do objeto de exibição com as cores do seu fundo e subtrai seus valores dependendo de qual tiver o maior brilho, de modo a sempre obter um valor positivo. Esta transformação é mostrada na Figura 11.

Figura 11 – Segmentação por subtração de fundo sem a aplicação de threshold

A imagem resultado dessa subtração passa então por uma transformação de RGB para tons de cinza. Essa transformação é realizada a fim de atenuar os efeitos da variação de luz no ambiente, bem como diminuir a quantidade de informação associada a cada *pixel*, já que seus valores de vermelho, verde e azul tornam-se iguais. Grande parte das abordagens de reconhecimento de gestos usa espaços de cor diferentes do RGB, pelo fato de este ser um espaço de cor vetorial (BRITTO, 1997), não se assemelhando a maneira como o ser humano enxerga, optando assim por esquemas de cor como o HSI, que oferece a manipulação de características como a cor, a saturação e a iluminação sobre o *pixel*. Para passar a imagem de entrada para a escala de cinzas, a saturação da imagem foi reduzida a zero, usando as funções de mudança contraste da biblioteca ColorMatrix, re-implementada por (SKINNER, 2001), que possibilita a manipulação das características espaciais das imagens sem a necessidade de transformar o espaço de RGB para HSI discretamente. A Figura 12 mostra a distribuição das cores nos espaços RGB e HSI.

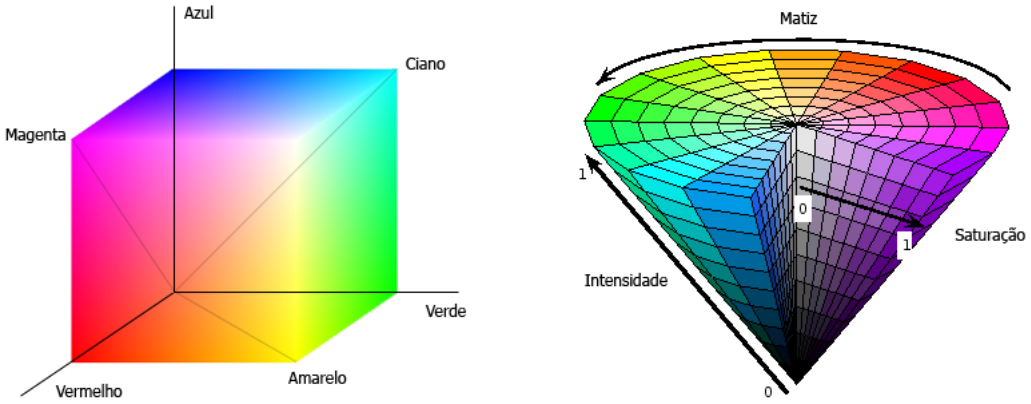

Figura 12 – Representação dos espaços de cor RGB e HSI

Em seguida o resultado é submetido a um processo de limiarização, mais conhecido como *thresholding*, onde são filtradas apenas as cores que respeitem os limites atribuídos. Nos filtros de limiarização vistos normalmente, os pontos com cores abaixo da faixa de corte são convertidos para pretos, os acima para brancos. Já o método threshold da classe BitmapData permite que seja atribuída uma cor aos *pixels* que passaram no filtro, possibilitando assim transformar a imagem de entrada em uma distribuição binária de quaisquer cores, exemplificada pela Figura 13, facilitando posteriormente o processo de seleção dos *blobs*. Após a limiarização, um filtro de desfoque é aplicado à imagem, para minimizar os ruídos, como detalhado no item 4.2.1. Por fim, a imagem anterior do *buffer*, que servia como base de comparação, é substituída pela atual, de maneira a possibilitar a dinamicidade do processo,

caracterizando este método como uma estratégia adaptativa de subtração de fundo (FERREIRA, 2007, p. 13-14).

Figura 13 – Segmentação por subtração de fundo e threshold

Terminada a etapa de subtração de fundo, é dado início ao reconhecimento dos possíveis *blobs*, pontos ou regiões com grande contraste em relação ao resto da imagem resultante. A estratégia de destaque dos *blobs* consiste em determinar a área onde possivelmente podem existir *blobs*, e em seguida fazer uma varredura dessa área procurando-os.

Determinar a área de possíveis ocorrências de *blobs* é simples: como observado na Figura 13, os *pixels* de cor branca são relativos às áreas onde o movimento é detectado, então basta criar um retângulo englobando todos os *pixels* brancos e a área resultante é garantidamente a maior área possível de ocorrência.

Em seguida é examinada a primeira coluna dessa área buscando a cor de base, neste caso o branco. Quando um *pixel* branco é encontrado, é marcada a forma constituída pelos *pixels* vizinhos com uma cor arbitrária. Para isso é usada a função floodFill da classe BitmapData, que preenche todos os *pixels* vizinhos ao nó desejado, desde que tenham a mesma cor deste nó, com outra cor escolhida. A partir dessas marcações criam-se novos retângulos envolvendo estas formas. Caso os retângulos se adéqüem às restrições de tamanho definidas pelo desenvolvedor, então um *blob* foi encontrado, como mostrado na Figura 14. O Quadro 4 mostra a implementação deste processo.

```
while (true)
{
      mainRect = _done.getColorBoundsRect(BASE_COLOR, BASE_COLOR);
      if (mainRect.isEmpty()) break;

      var x:int = mainRect.x;

      for (var y:uint=mainRect.y; y < mainRect.y+mainRect.height; y++)
```

```
        {
                if (_done.getPixel32(x, y) == BASE_COLOR)
                {
                        _done.floodFill(x, y, 0x00ff00);

                        blobRect = _done.getColorBoundsRect(0xFFFFFF, 0x00ff00);

                        if ((blobRect.width * blobRect.height)  > 300)
                        {
                                addCursor(blobRect);
                        }

                        break;
                }
        }
}
```

Quadro 4 - Rotina de detecção de blobs

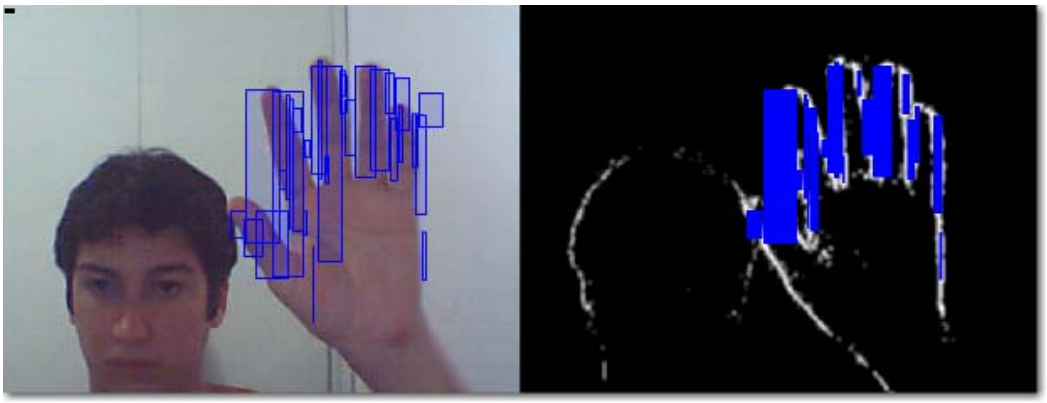

Figura 14 – Detecção de blobs

4.3. CURSORES

Esta classe foi criada para servir de orientação do reconhecimento do objeto selecionado pelo usuário. Basicamente um cursor é constituído de um *bouding box*, um retângulo que marca os pontos extremos dos objetos detectados. Quando criado, é atribuído a cada cursor um identificador único e uma cor igual à cor do objeto selecionado, de maneira a possibilitar sua distinção e a coexistência de múltiplos cursores em um mesmo ambiente.

A partir da *bouding box* são calculadas as posições do cursor nos três eixos, sendo o seu centróide responsável pela localização no eixo (x,y) e o zIndex, calculado a partir da razão entre a área inicial da *bouding box* e sua área atual, responsável pela localização relativa ao eixo z. A implementação do cursor é apresentado no Quadro 5.

Vale lembrar que a biblioteca suporta a criação de múltiplos cursores simultaneamente durante a mesma execução de uma aplicação, possibilitando o seu uso em sistemas onde se façam necessários os conceitos de interfaces colaborativas.

```
_boundingBox.graphics.clear();
_boundingBox.graphics.beginFill(_color, .1);
_boundingBox.graphics.lineStyle(3, _color);
_boundingBox.graphics.drawRect(rect.x,rect.y,rect.width,rect.height);

_centroid = new Point(rect.x+(rect.width/2),rect.y+(rect.height/2));

_currentArea = rect.width * rect.height;
_zindex = _currentArea / _initialArea;
```

Quadro 5 - Código de criação e atualização de um cursor

5. APLICAÇÕES DESENVOLVIDAS E RESULTADOS

Com a finalidade de testar a biblioteca ao longo do seu desenvolvimento foram criadas algumas aplicações que usam tipos de interação diversos, que ajudam a demonstrar seu funcionamento, versatilidade e potencial. Essas aplicações são apresentadas abaixo.

Para fins de comparação, e testes em ambientes diversos, as aplicações foram executadas em dois computadores diferentes. A configuração básica dos dois é dada na Tabela 1, a seguir.

Tabela 1 – Configuração dos ambientes de teste

Ambiente 1		**Ambiente 2**	
Processador	Intel Core 2 Duo E8400 3.0 GHz	Processador	Pentium Dual Core T4500 2.3 GHz
Vídeo	NVIDIA GeForce 8600 GTS	Vídeo	Mobile Intel 4 Series Express
Memória	2.0 Gb	Memória	2.0 Gb

5.1. APLICAÇÃO 1 – NAVEGAÇÃO ANÁGLIFA

Esta aplicação consiste na visualização de um espaço tridimensional onde é renderizada uma série de cubos aleatoriamente. A movimentação através desse campo é feita através da biblioteca proposta, sendo a translação no eixo (x,y) feita através do centróide do cursor e a movimentação pelo eixo z através da propriedade zIndex do cursor. A aplicação foi desenvolvida usando a biblioteca PaperVision3D, uma *engine* 3D para Flash.

A aplicação funciona usando um sistema de *widgets*, podendo executar dentro de si diversas aplicações. Foi feita desta maneira, pois eram necessários testes de validação do funcionamento da biblioteca de navegação, e da biblioteca de estereosopia, exemplificada a

seguir, e para tal foram desenvolvidos mais de um mini-aplicativo. É mantida uma lista de aplicativos, armazenados em um arquivo XML contendo dados como nome, localização em disco e ícone, que é carregada no início da aplicação pai. A partir dessa lista é criado um menu lateral contendo os ícones relativos a todas as aplicações carregadas, além de uma área que exibe a cor selecionada pelo usuário para o rastreamento. No canto superior esquerdo é mantido o vídeo da câmera do usuário, exibindo o conjunto de cursores referentes às cores escolhidas, e uma representação de como o vídeo está sendo interpretado, habilitado no modo de debug do sistema, para verificação do funcionamento do método, como mostrado na Figura 16.

Uma forma simples de aumentar a sensação de imersão dos usuários em ambientes de realidade virtual é o uso de estereoscopia. Bem como o uso de formas mais naturais de navegação são importantes em ambientes tridimensionais, a utilização da estereoscopia é interessante já que a visão humana é baseada nela, com aquisição de duas imagens ligeiramente diferentes do mundo a cada momento, possibilitando assim o senso de profundidade (PERSIANO, 2004).

Como um dos objetivos desta biblioteca é fornecer ao usuário uma experiência mais rica de navegação, a primeira aplicação desenvolvida usa o conceito de estereoscopia, criada através de anáglifos, para maximizar a imersão dos usuários na simulação da navegação através deste campo tridimensional. Um anáglifo é uma composição de duas imagens com discrepância suficiente para criar uma sensação de profundidade, como mostrado na Figura 15, com seus canais de cor ligeiramente separados e sobrepostos, usando o princípio da quebra de fusão e complementaridade de cores (PERSIANO, 2004).

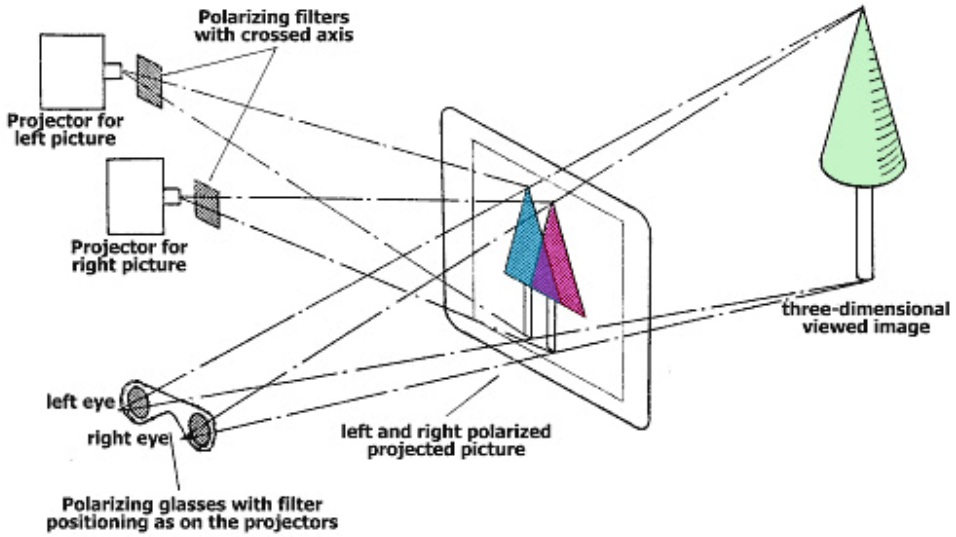

Figura 15 – Representação do funcionamento de modelos anáglifos

Para criar a anaglifia da aplicação foi desenvolvida uma biblioteca de estereoscopia que renderiza as cenas em 3D usando duas câmeras, cada uma representando um olho. As câmeras são separadas entre si de maneira a simular a distância entre as retinas humanas e sofrem transformações de cor para que sejam filtradas pelas lentes dos óculos 3D e sobrepostas usando-se um filtro Multiply, da própria Adobe, que multiplica as cores das imagens. Para esta aplicação foram usados os filtros ciano e magenta, por serem os mais freqüentemente encontrados.

A Figura 16 ilustra o uso de aplicação a partir da seleção de uma cor presente nos óculos 3D, aumentando ainda mais a sensação de imersão no ambiente. A Tabela 2 exibe os resultados dos testes nos diferentes ambientes.

Tabela 2 – Resultados de performance do Aplicativo 1

Ambiente 1		**Ambiente 2**	
FPS	15-20 fps	**FPS**	9-10 fps
Memória Ocupada	~26mb	**Memória Ocupada**	~13mb

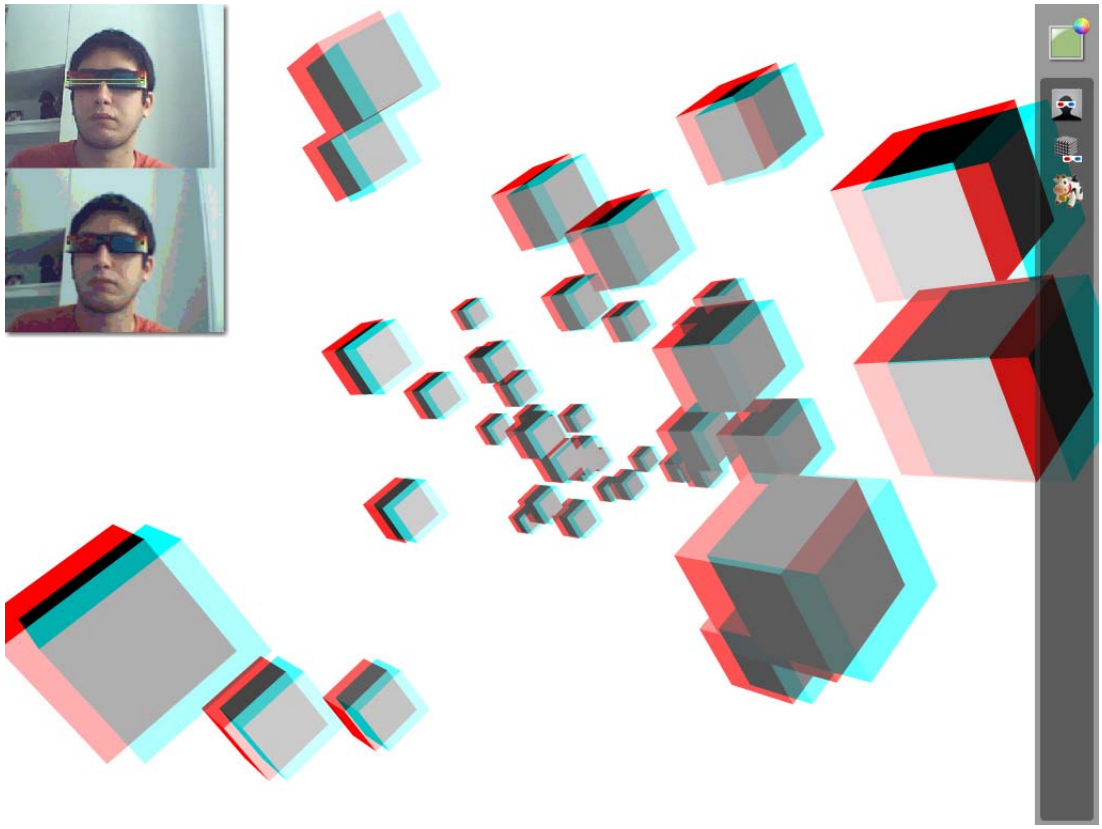

Figura 16 – Interface da navegação em ambiente anáglifo

5.2. APLICAÇÃO 2 – DESENHO 3D

Esta aplicação é um exemplo simples de como os cursores da biblioteca podem ser usados como ponteiros, semelhantes aos do mouse convencional. O usuário interage com a cena da seguinte forma:

Os movimentos no eixo (x,y) do mouse guiam o desenho da fita, que se movimenta em direção ao ponteiro do mouse. Ao mesmo tempo, o globo central, a âncora de iluminação e a fita que está sendo desenhada giram com a velocidade variando em relação à profundidade do mouse. Para a implementação do desenho em forma de fita foi usada a API Ribbon3D, que utiliza o Papervision3D e curvas de Bezier para possibilitar estas formas. A Figura 16 mostra a aplicação em uso. A Tabela 3 exibe os resultados dos testes nos diferentes ambientes.

Tabela 3 – Resultados de performance do Aplicativo 2

Ambiente 1		**Ambiente 2**	
FPS	31-35 fps	**FPS**	15-24 fps
Memória Ocupada	~32mb	**Memória Ocupada**	~23mb

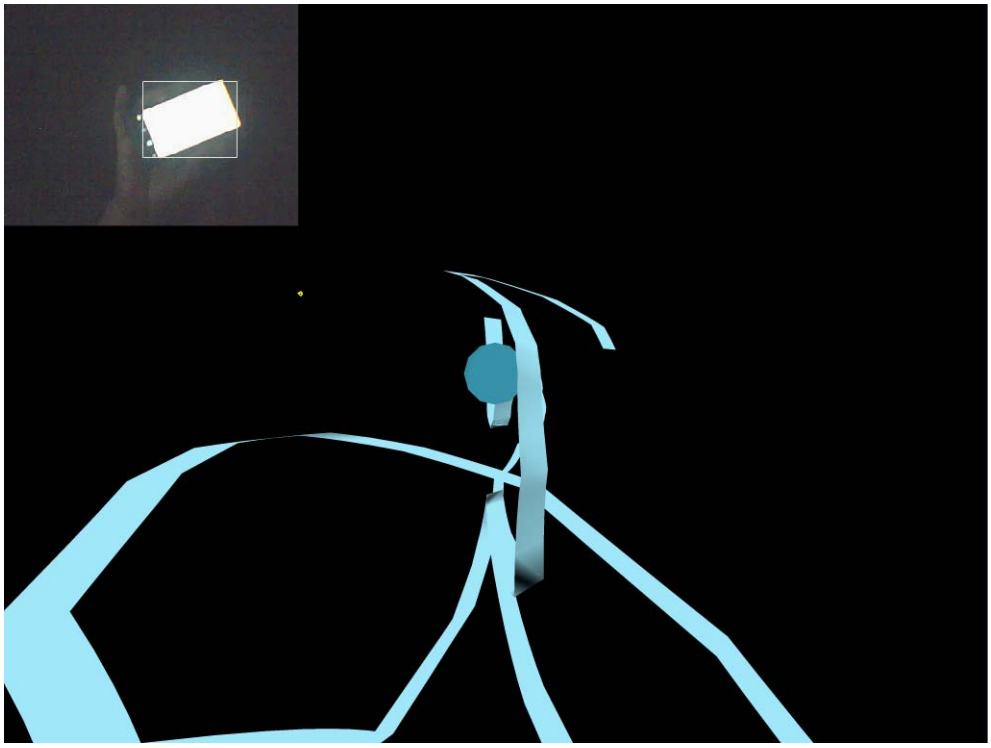

Figura 17 – Uso da aplicação de desenho em 3D

5.3. APLICAÇÃO 3 – OBJETOS INTERATIVOS

Esta aplicação mostra como é possível o uso do cursor como instrumento de interação com objetos tridimensionais a partir da simulação de eventos de mouse como o clique. Neste exemplo também fica evidente a eficiência da biblioteca em aplicação que necessitem de mouse 3D.

O autor dessa aplicação a projetou uma interação com objetos usando o mouse. Os blocos reagem ao pressionar o mouse e podem ser soltos ao soltar o botão do mesmo. A intenção deste exemplo é demonstrar como é simples adaptar um código já existente a esta biblioteca.

Para tornar essa aplicação compatível com a navegação através de gestos foi necessário instanciar a classe base da biblioteca, a ObjectTracking, e adicionar o vídeo de feedback da própria biblioteca no *stage* para que o usuário possa selecionar um objeto e tenha uma visualização acerca da localização do cursor. O aplicativo fica então aguardando o clique do mouse no objeto da câmera. Ao selecionar uma cor, o rastreamento é iniciado. Como o aplicativo responderá aos eventos do cursor da biblioteca, deve-se adicionar um *listener* ao objeto ObjectTracking que aguarde um evento ObjectTrackingEvent.INIT, garantindo assim que as estruturas do método de detecção, bem como o cursor, foram inicializados corretamente.

A ativação do evento de clique é determinada pelo aumento da área do cursor. Quando esta área atinge uma escala 1.2x maior que área original o evento é disparado. Como as primitivas ficam estacionadas no piso da tela, usar o centróide do cursor como ponteiro não era uma boa estratégia, pois com a diminuição da área do cursor durante a saída do objeto rastreado da tela não era possível garantir que a área se mantivesse do tamanho mínimo para a captura, e o centróide atingisse o objeto. Foi necessário deslocar o ponteiro um pouco para baixo do centróide, de maneira a solucionar o problema.

Um exemplo do código que explicita o uso da biblioteca é mostrado no item 9.1 do apêndice. Além disso, a Figura 17 mostra a execução da aplicação. A Tabela 4 exibe os resultados dos testes nos diferentes ambientes.

Tabela 4 – Resultados de performance do Aplicativo 3

Ambiente 1		**Ambiente 2**	
FPS	26 fps	**FPS**	13 fps
Memória Ocupada	~26mb	**Memória Ocupada**	~13mb

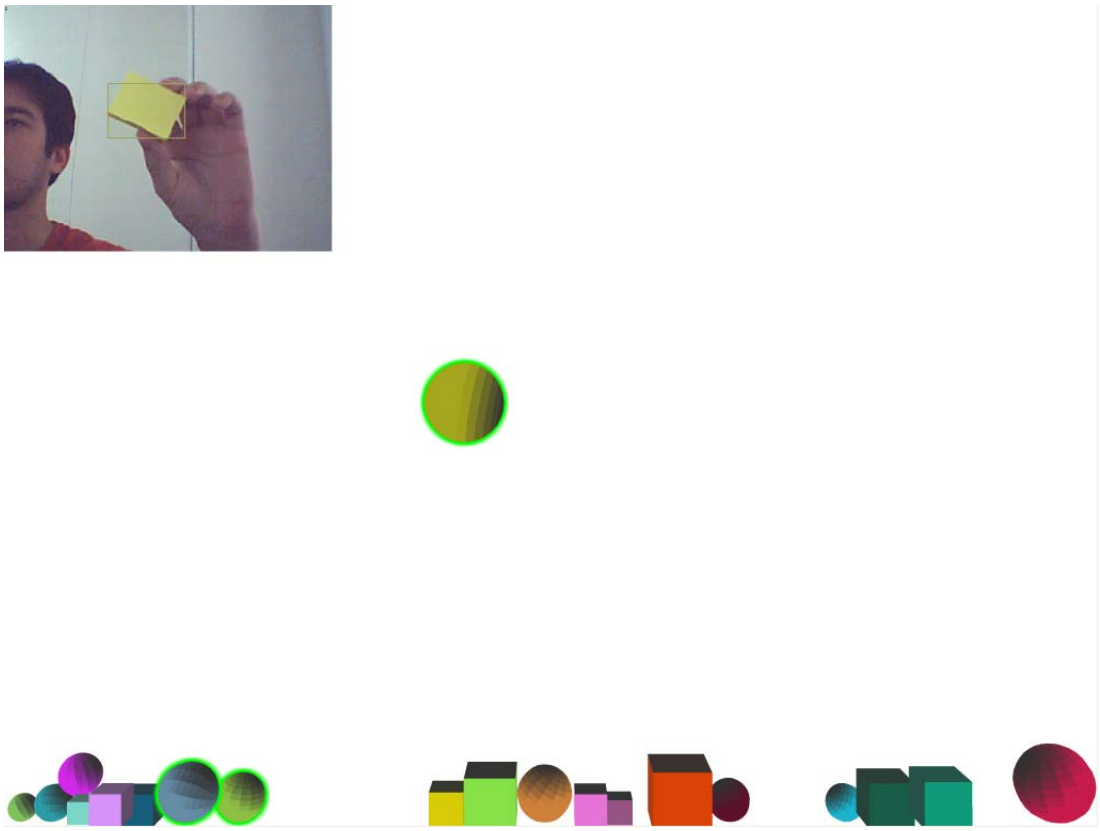

Figura 18 – Exemplo da manipulação de objetos virtuais

5.4. APLICAÇÃO 4 – PONG

Para demonstrar o uso de múltiplos objetos sendo reconhecidos e identificados simultaneamente pela biblioteca foi criada uma versão do jogo Pong, um simulador do jogo de tênis de mesa, onde os jogadores devem acertar a bolinha de modo a não permitir que seu adversário consiga rebater.

Esta versão da aplicação aguarda até que dois jogadores, representados pelos dois blocos, tenham seus objetos de reconhecimento escolhidos e mapeados pela biblioteca. Feito isso o cenário é renderizado e cada objeto no mundo real passa a controlar um bloco dentro do jogo.

A Figura 19 mostra a aplicação em uso. A Tabela 5 exibe os resultados dos testes nos diferentes ambientes.

Tabela 5 – Resultados de performance do Aplicativo 4

Ambiente 1		**Ambiente 2**	
FPS	33 fps	**FPS**	28 fps
Memória Ocupada	~21mb	**Memória Ocupada**	~21mb

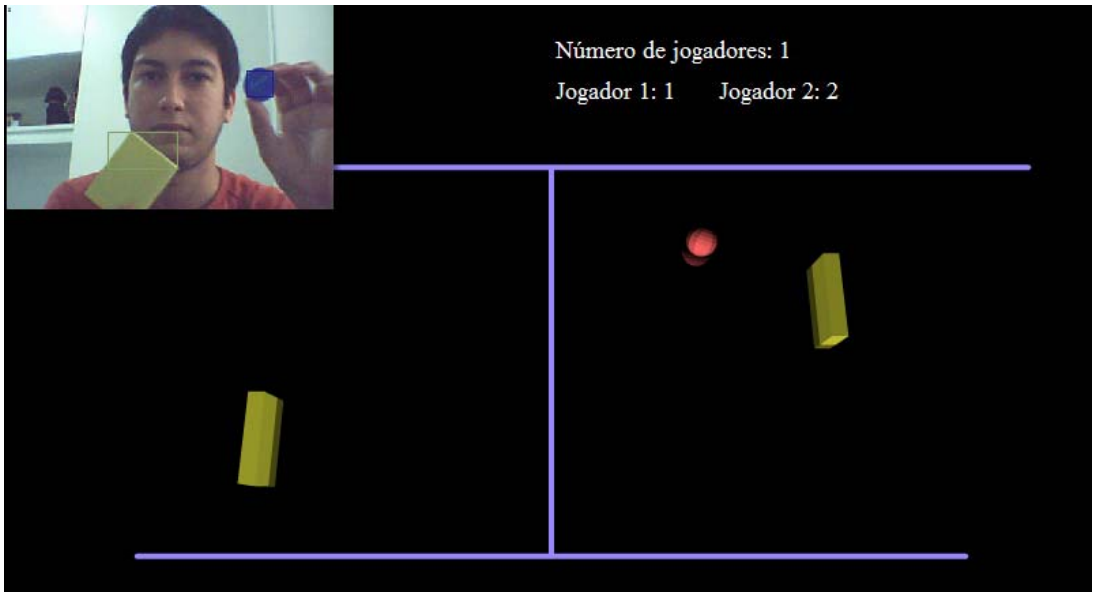

Figura 19 – Exemplo do jogo Pong. Usabilidade em interfaces colaborativas

6. CONCLUSÃO

Com o desenvolvimento desta biblioteca é mostrada a possibilidade de criar maneiras inovadoras de interação com o uso de objetos simples, sem a necessidade da obtenção de dispositivos mais complexos, ampliando assim a usabilidade e a acessibilidade da tecnologia vista atualmente.

Alguns algoritmos de reconhecimento analisados tiveram de ser simplificado devido a algumas limitações do player do Flash, como o uso de memória, sem grandes perdas na qualidade dos resultados obtidos, graças a uma gerência eficaz dos recursos disponíveis. Além disso, não é descartada a possibilidade de sua implementação de forma mais robusta, com um cuidado um pouco maior em relação ao uso de memória, aplicando-se referências fracas nas funções de eventos e desalocando de forma mais inteligente os objetos criados a cada instante nas funções de detecção. Apesar dessas limitações, acredita-se que a grande difusão da ferramenta de implementação, bem como do seu plug-in de visualização, unidos a facilidade da adaptação da navegabilidade dos aplicativos atuais permitam que a biblioteca seja bastante usada na criação de sistemas RIA com interações mais inteligentes na internet.

A capacidade de inclusão de novos métodos de reconhecimento sem maiores esforços de implementação para a comunidade também é uma característica marcante no uso da biblioteca, possibilitando seu crescimento e adaptação às novas tecnologias que estão por vir.

Com isso, conclui-se que os objetivos iniciais da construção da biblioteca foram atingidos. Flexibilidade, acessibilidade e eficiência fazem da biblioteca uma colaboração satisfatória à comunidade.

7. TRABALHOS FUTUROS

Inicialmente a intenção de continuidade desse projeto é focada na inclusão de novas técnicas de processamento de imagens e reconhecimento de gestos. Existem algoritmos como o CamShift, que também é capaz de fazer a detecção de objetos com baixo custo de processamento, mas ainda reconhece a rotação dos objetos, que não é reconhecida por nenhum dos algoritmos implementados até então. Além do CamShift, existem maneiras mais eficazes de reduzir as restrições em relação à luminosidade, como a escolha de outros espaços de cor, e fragmentação dos *pixels* das imagens captadas pela câmera.

Outro trabalho desejável é a construção de um site em Flash que use de forma mais completa e criativa todos os recursos que uma biblioteca como essa podem fornecer, como navegações em profundidade por conteúdos semelhantes, organização de conteúdos fotográficos a partir de seus matizes, rastreamento de objetos em vídeos na internet, e tantas outras aplicações.

Além disso, concentrar esforços na melhoria da arquitetura e, por conseguinte, do desempenho da biblioteca, tratando melhor de recursos como a memória, e reforçar o uso de tecnologias como o PixelBender, uma API de processamento de imagens e vídeos, também da Adobe, que possibilitam um processamento mais eficiente de conteúdos de vídeo no Flash.

8. OBRAS CITADAS

ADOBE. ActionScript 3.0 Language and Components Reference. Disponível em: <http://livedocs.adobe.com/flash/9.0/ActionScriptLangRefV3/index.html?all-index-Symbols.html&index-list.html> Acesso em: 10 de novembro de 2010.

AZARBAYJANI, A.; WREN, C.; PENTLAND, A. Real-Time 3D Tracking of the human body. 1996. In: IMAGE'COM, 1996. Anais… v. 1.

AZARI, Hossein; CHENG, Irene; BASU, Anup. Stereo 3DMouse Cursor: AMethod for Interaction with 3D Objects in a Stereoscopic Virtual 3D Space. 2009 Department of Computing Science, University of Alberta. 2009.

AZUMA, Ronald T. A Survey of Augmented Reality. Teleoperators and Virtual Environments. 1997. v. 6, nº 4, p. 355-385.

BÉRARD, F. Computer Vision for the Strongly Coupled Human-Computer Interaction. 1999. Tese (Doutorado) - Université Joseph Fourier, Grenoble 1999.

BÉRARD F.; HARDENBERG C. Bare-Hand Human-Computer Interaction. In: ACM WORKSHOP ON PERCEPTIVE USER INTERFACES, 2001, Orlando, Florida, USA. Anais… 2001.

BRITTO, Alceu Souza. Segmentação por cor utilizando a Transformada de Karhunen-Loève. 1995.Dissertação (Mestrado em Informática Industrial), Pós-Graduação em Engenharia

Elétrica e Informática Industrial, Centro Federal de Educação Tecnológica do Paraná. 1995.

BRITTO, Alceu Souza; BORGES, Díbio S. L.; FACON, Jacques. Técnica Interativa da Segmentação por Cor: Aplicação à Quantificação da Mucina. In: XXIV SEMISH 97, BRAZILIAN SOFTWARE & HARDWARE SEMINARS, 1997, Brasília.

CODEJOCKEY: Papervision3D Anaglyph Scene. Disponível em <http://codejockeyscorner.blogspot.com/2009/04/papervision3d-anaglyph-scene.html> Acesso em: 23 de novembro de 2010.

COMANICIU, D.; MEER, P. Mean shift: A robust approach toward feature space analysis. IEEE Transaction on Pattern Analysis and Machine Intelligence. 2002. v. 24, nº 5, p. 603–619.

FERREIRA, Mauricio A. Lage. Detecção de Movimento através de Subtração de Fundo para Vigilância Eletrônica Automática. 2007. Dissertação (Mestrado em Informática) - Pontifícia Universidade Católica. 2007.

KURTENBACH G.; HULTEEN E. A. Gestures in Human-Computer Interaction. In: B. Laurel (ed.): The Art of Human-Computer Interaction. 1990.

MISTRY, Parnav; MAES, P. *SixthSense – A Wearable Gestural Interface*. In: SIGGRAPH Asia 2009. 2009. Anais... Sketch. Yokohama, Japão. 2009.

MYERS, B. A. A Brief History of Human Computer Interaction Technology." ACM interactions. 1998. v. 5, nº 2, p. 44-54.

MYERS, Brad A. The User Interface for Sapphire. IEEE Computer Graphics and Applications, 1984. v. 4, nº 12, p. 13-23.

OLIVER, N.; ROSARIO, B.; PENTLAND, A. A bayesian computer vision system for modeling human interactions. 2000. IEEE Transaction on Pattern Analysis and Machine Intelligence. v. 22, nº 8, p. 831–843.

PAPERVISION3D: Open Source realtime 3D engine for Flash. Disponível em: <http://code.google.com/p/papervision3d/> Acesso em: 23 de novembro de 2010.

PERSIANO, Ronaldo. Técnicas de Visualização 3D no Tratamento Ortóptico. 2004. Disponível em: <http://vergencia.w3br.com/artigos/3d/3d.htm> Acessado em: 19 de novembro de 2010.

SHI, J; MALIK, J. Normalized cuts and image segmentation. IEEE Transaction on Pattern Analysis and Machine Intelligence. 2000. v. 22, nº 8, p. 888–905.

SKINNER, Grant: ColorMatrix Class in AS3. Disponível em: <http://www.gskinner.com/blog/archives/2007/12/colormatrix_upd.html> Acesso em: 21 de novembro de 2010.

STAUFFER, C.; GRIMSON, W. Learning patterns of activity using real time tracking. 2000. IEEE Transaction on Pattern Analysis and Machine Intelligence. v. 22, nº 8, p. 747–767.

SUTHERLAND, Ivan E. SketchPad: A Man-Machine Graphical Communication System. In: AFIPS SPRING JOINT COMPUTER CONFERENCE. 1963. v. 23, p. 329-346.

WINDLE, Justin Clarke. *3D Ribbons in Papervision*. Disponível em <http://blog.soulwire.co.uk/code/actionscript-3/papervision3d-ribbons#more-76>. Acessado em: 23 de novembro de 2010.

VLIET, Lucas J. van; YOUNG, Ian T.; VERBEEK, Piet W. Recursive Gaussian Derivative Filters. 1998. IEEE Computer Society Press, v. 1, p. 509-514.

ZIGELBAUM, Jamie et al. g-stalt: a Chirocentric, Spatiotemporal, and Telekinetic Gestural Interface. 2010. Tangible Media Group, MIT Media Lab. 2010.

9. APÊNDICE

9.1. EXEMPLO DE USO DA BIBLIOTECA

```
private function Main():void
{
	viewport.interactive = true;

	objTracking = new ObjectTracking(stage.stageWidth,
stage.stageHeight);

	trackingDisplay = objTracking.getDisplays();
	trackingDisplay.scaleX = trackingDisplay.scaleY = .3;
	addChild(trackingDisplay);
	trackingDisplay.addEventListener(MouseEvent.CLICK, startTracking);

}

private function startTracking(e:MouseEvent):void
{
	objTracking.addEventListener(ObjectTrackingEvent.INIT, init);
	trackingDisplay.removeEventListener(MouseEvent.CLICK,
startTracking);

	objTracking.initialize(ObjectTrackingType.SINGLE_SEGMENTATION, new
Point(trackingDisplay.mouseX, trackingDisplay.mouseY));
}

private function init(e:ImageProcessEvent):void
{
	cursor = objTracking.getCursorAt(0);
	startRendering();
}

private function updateMouseWorld():void
{
	//mouseXWorldPhys = mouseX / WORLD_SCALE;
	//mouseYWorldPhys = mouseY / WORLD_SCALE;
	mouseXWorldPhys = cursor.centroid.x / WORLD_SCALE;
	mouseYWorldPhys = (cursor.centroid.y + 20) / WORLD_SCALE;
}
```

9.2. EXEMPLO DA ESTRUTURA DO ARQUIVO XML DO APLICATIVO 1

```
<?xml version="1.0" encoding="utf-8"?>
<VGRApplications>
    <app id="0" name="Application1" src=" Application1.swf " icon="Application1.png "></app>
    <app id="1" name=" Application2" src=" Application2.swf " icon=" Application2.png "></app>
```

```
</VGRApplications>
```