Universidade Federal de Pernambuco
Centro de Informática
Graduação em Engenharia da Computação

ESTUDO DE FERRAMENTAS PARA DESENVOLVIMENTO DE SISTEMAS DE REALIDADE AUMENTADA NA WEB

TRABALHO DE GRADUAÇÃO

Aluno: Pablo Carvalho Pinheiro

Orientadora: Veronica Teichrieb

Co-orientador: João Paulo Silva do Monte Lima

Recife, julho de 2010

Universidade Federal de Pernambuco
Centro de Informática
Graduação em Engenharia da Computação

ESTUDO DE FERRAMENTAS PARA DESENVOLVIMENTO DE SISTEMAS DE REALIDADE AUMENTADA NA WEB

TRABALHO DE GRADUAÇÃO

Trabalho apresentado na Universidade Federal de Pernambuco como projeto de conclusão de curso para obtenção do grau de bacharel em Engenharia da Computação

Aluno: Pablo Carvalho Pinheiro

Orientadora: Veronica Teichrieb

Co-orientador: João Paulo Silva do Monte Lima

Recife, julho de 2010

RESUMO

Para que o usuário de uma aplicação de Realidade Aumentada possa interagir naturalmente com a mesma, é importante que o sistema consiga responder rapidamente ao que ocorre no ambiente ao seu redor. Um fator que pode ter forte influência no desempenho da aplicação é a plataforma sobre a qual ela foi desenvolvida. Este trabalho tem por objetivo comparar plataformas de desenvolvimento de aplicações para a *Web* com relação ao desempenho obtido por aplicações de Realidade Aumentada construídas sobre as mesmas. Com isto, visa-se oferecer aos desenvolvedores uma fonte de embasamento ao escolher uma plataforma de desenvolvimento para suas aplicações. Esta comparação baseia-se em resultados obtidos em uma aplicação implementada no escopo deste projeto, assim como em resultados disponíveis na literatura. Resultados mostram que um modelo cliente-servidor para a aplicação pode melhorar o desempenho da mesma, porém a aplicação poderá apresentar baixa escalabilidade. Quando o processamento é feito inteiramente no computador do usuário, pode-se trocar desempenho por portabilidade, dependendo da plataforma escolhida.

ABSTRACT

For the user of an Augmented Reality application to interact with it naturally, it is important that the system can react quickly to what happens on the environment around it. A factor that can influence the performance of the application is the platform on which it is built. This work has as its goal to compare Web development platforms with respect to the performance obtained on Augmented Reality applications built on them. With this, it is expected to offer the Web developers a base when choosing the development platform for his application. This comparison is based on results obtained with an application implemented in the scope of this project, as well as on results available on the literature. The results show that a client-server model for the application can increase its performance, but it can have scalability issues. When the processing is done entirely on the user's computer, performance can be exchanged by portability, depending on the platform chosen.

AGRADECIMENTOS

Esta monografia não é fruto apenas do meu trabalho, mas também do apoio de muitos que me ajudaram a chegar até este momento. Dedico-me a escrever esta seção em agradecimento àqueles que muito me ajudaram. Agradeço, então:

ao meu pai, Guilherme, por ter me apoiado nos meus melhores e piores momentos durante o curso,

à minha mãe, Denise, por, mesmo distante, estar sempre do meu lado, dando bons conselhos e me incentivando a seguir o caminho da computação, do qual tanto gosto e o qual ela me apresentou quando eu era ainda pequeno,

ao restante da minha família: tios, tias, primos e primas, que em muitos momentos também foram como pais e irmãos para mim,

aos meus orientadores, Veronica (vt) e João Paulo (Jonga), que me guiaram desde os meus primeiros passos no mundo da Realidade Aumentada, me sugeriram o tema deste projeto e acreditaram em mim, mesmo quando achei que nada iria dar certo,

à professora Judith, por confiar na minha capacidade de concluir com êxito este projeto e me incentivar a fazê-lo,

aos demais membros do Grupo de Pesquisa em Realidade Virtual e Multimídia, que sempre se mostraram dispostos a ajudar quando eu precisasse e que sempre tinham bons conselhos de como eu poderia melhorar as apresentações que fiz durante este tempo que tenho trabalhado junto a eles (Deus salve as prévias),

ao time, digo, à família UFPE da Maratona de Programação, pelos momentos de estudos, de tensão, de tristezas e de alegrias que temos passado juntos e por terem me ensinado e, ao mesmo tempo, aprendido que estudar, além de ser um dever, pode ser um ótimo lazer quando se gosta do que faz. Em especial, aos meus colegas de time: Eduardo (Curupa), Pedro, Hallan, Filipe e Luiz, pelos conhecimentos trocados, pelo esforço dedicado aos estudos, pela descontração séria e pela seriedade descontraída dos treinos e das competições das quais participei. Também à professora Liliane que, acreditando no potencial dos alunos que se interessavam pela competição, conseguiu transformar a Maratona de Programação em um projeto reconhecido e apoiado no Centro de Informática, e hoje nos incentiva a darmos o melhor para defender o nome da Universidade na competição,

aos colegas da minha turma de Engenharia e de Ciência da Computação, pelas noites viradas, pelos grupos de estudo em véspera de prova, pelos projetos concluídos juntos, enfim, pelo que a gente teve que enfrentar nesses longos anos de curso,

aos amigos do “minha seCÇão de RPG demooora..”, pelos vários e bons momentos de diversão, afinal, nem só de estudos vive um nerd,

a Java e Lucas, que, por se encaixarem em tantas categorias acima, acabaram ganhando mais uma de brinde.

SUMÁRIO

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

1. INTRODUÇÃO

Com o intuito de promover o entendimento do conceito Realidade Aumentada, um dos focos deste trabalho, é interessante expor alguns conceitos relacionados. Primeiramente, um sistema de Realidade Virtual é aquele que visa a imersão do usuário em um mundo virtual, tirando o foco do mesmo da realidade que o circunda e fazendo-o acreditar, da melhor forma possível, que ele está inserido em um mundo criado artificialmente e que com este pode interagir. Em oposição, temos o mundo real, que é formado pelos objetos físicos presentes no ambiente em que o usuário se encontra, ou seja, formado de objetos que não são virtuais. Entre estes extremos, existe o que pode ser chamado de Realidade Misturada, em que objetos virtuais e objetos reais se misturam e, possivelmente, interagem entre si, dando ao usuário a impressão de que todos os objetos presentes na cena realmente estão ao seu redor. Esta realidade pode ser ainda classificada como Virtualidade Aumentada, em que é predominante a existência de objetos virtuais, ou Realidade Aumentada, em que objetos virtuais são sobrepostos a uma cena real, ou seja, em que o mundo real é aumentado com informações geradas computacionalmente [1].

Para que seja possível sobrepor objetos virtuais ao mundo real de forma natural, é necessário que o sistema de Realidade Aumentada consiga posicionar os mesmos na cena coerentemente com os objetos reais que nela se encontram. Isto pode ser feito utilizando-se de pontos de referência no mundo real. O problema de localizar os pontos de referência em uma imagem do ambiente real onde devem ser inseridos os objetos virtuais é conhecido como rastreamento. Estes pontos de referência podem ser marcadores artificiais (chamados de marcadores fiduciais) inseridos no ambiente apenas para o correto posicionamento dos objetos virtuais, ou podem ser utilizadas características naturalmente presentes no mundo real como referenciais. A Figura 1 mostra um exemplo de um marcador comumente utilizado em sistemas de Realidade Aumentada baseados em marcadores [2].

FIGURA 1. EXEMPLO DE MARCADOR FIDUCIAL

A Figura 2 mostra um cenário de uma aplicação de Realidade Aumentada sem marcadores, em que um objeto virtual é posicionado sobre uma caixa de brinquedo [3]. Este tipo de apli-

cação baseia-se no uso de características naturais da cena como referência para o posicionamento dos objetos virtuais, como, por exemplo, a textura de um objeto presente na cena ou mesmo a própria geometria do ambiente, que pode ser inferida durante a execução da aplicação.

FIGURA 2. REALIDADE AUMENTADA SEM MARCADORES

Sistemas de Realidade Aumentada sem marcadores têm recebido bastante atenção ultimamente, pois, por não precisarem de marcadores artificiais na cena a ser aumentada, permitem o desenvolvimento de aplicações que podem ser utilizadas pelo usuário sem a necessidade de preparar previamente o ambiente. Um exemplo são *softwares* de publicidade [4, 5], que precisam ser atraentes para o consumidor e, portanto, se beneficiam com a possibilidade de não utilizar marcadores intrusivos na cena.

Outro foco deste trabalho é o desenvolvimento de aplicações voltadas para a *Internet.* A *Web* é uma plataforma bastante atrativa para aplicações que necessitam de visibilidade, como as de publicidade, ou de portabilidade (isto é, capacidade de executar sobre diferentes sistemas operacionais e até mesmo diferentes dispositivos). A *Internet* também tem como vantagem a facilidade da distribuição do produto, visto que normalmente não é necessário que o usuário instale o programa no seu computador, dispensando, assim, o uso de mídias externas (como CD ou DVD) para instalação da aplicação.

Para que seja possível aproveitar as vantagens da plataforma *Web* em sistemas de Realidade Aumentada, é importante que o desenvolvedor conheça as vantagens e desvantagens de diversas plataformas de desenvolvimento disponíveis, de forma que possa escolher qual melhor se adequa à aplicação a ser implementada.

1.1 OBJETIVO

Neste contexto, o objetivo deste trabalho de graduação é realizar um estudo comparativo entre plataformas de desenvolvimento *Web* que podem ser utilizadas para criação de sistemas de Realidade Aumentada sem marcadores. Neste estudo serão realizados testes utilizando os *frameworks* Mammoth [6], OSAKit [7] e Silverlight [8], contemplando três técnicas de rastreamento sem marcadores: *Scale Invariant Feature Transform* (SIFT) [9], otimiza-

da para ser executada utilizando a placa gráfica [10], *Speeded Up Robust Features* (SURF) [11] e Ferns [12].

1.2 ESTRUTURA DO DOCUMENTO

Esta monografia está organizada da seguinte forma. O próximo capítulo visa introduzir o contexto em que este trabalho de graduação se insere. O capítulo 3 trata da implementação de uma aplicação feita durante o período deste trabalho. O capítulo seguinte apresenta uma comparação dos resultados obtidos nos testes das aplicações sobre diferentes plataformas e, por fim, o capítulo 5 traz as conclusões que puderam ser obtidas a partir dos resultados, assim como alguns trabalhos a serem realizados futuramente.

2. CONTEXTUALIZAÇÃO

Uma etapa importante do planejamento de uma aplicação de Realidade Aumentada é a escolha da forma de rastreamento que será utilizada para a correta sobreposição dos objetos virtuais na cena real. Se a adição de marcadores na cena for possível e não trouxer prejuízos à aplicação, o rastreamento baseado em marcadores é uma solução simples e eficiente. Por outro lado, o rastreamento de características naturais da cena pode ser necessário quando, por exemplo, não é possível alterar o ambiente sobre o qual será disposta a informação virtual (ambientes de difícil acesso) ou quando a adição de marcadores não é desejada, como no caso de diversas aplicações de publicidade, nas quais a adição de um marcador fiducial provocaria a perda da naturalidade da interação do usuário com a aplicação.

Como dito anteriormente, este trabalho de graduação foca em aplicações de Realidade Aumentada sem marcadores desenvolvidas para a *Web*. A seção a seguir apresenta uma visão geral sobre técnicas de rastreamento sem marcadores. A seção 2.2 traz alguns exemplos de trabalhos relacionados, mostrando algumas soluções que foram desenvolvidas nos últimos anos trazendo a Realidade Aumentada para a *Internet*.

2.1. TÉCNICAS DE RASTREAMENTO SEM MARCADORES

Aplicações de Realidade Aumentada sem marcadores são, normalmente, mais atrativas para o usuário do que as que utilizam rastreamento baseado em marcadores, pois possibilitam a utilização de objetos mais naturais ao cotidiano do usuário para o cálculo do posicionamento dos objetos virtuais, de forma que o usuário não precise utilizar um marcador fiducial, que costuma ser estranho para usuários leigos, para que possa interagir com a aplicação.

Esta vantagem, porém, costuma vir associada a um maior custo computacional para a identificação do objeto rastreado na cena capturada pela câmera. Diversas técnicas de Realidade Aumentada sem marcadores têm surgido nos últimos anos visando melhorar o desempenho do rastreamento, assim como remover algumas restrições impostas por técnicas anteriormente divulgadas (como a limitação de rastreamento apenas de objetos planares ou não reflexivos, comum a algumas técnicas). Por este motivo, o estudo de diversas técnicas é importante para o desenvolvimento de aplicações de Realidade Aumentada sem marcadores, visando, com isso, a escolha da técnica que melhor se adéqua ao cenário de utilização da mesma.

Esta seção visa apresentar uma visão superficial sobre as principais técnicas de rastreamento sem marcadores utilizadas atualmente, focando, posteriormente, nas técnicas que foram escolhidas para o desenvolvimento das aplicações de Realidade Aumentada para *Web* que foram utilizadas neste trabalho para fins de comparação entre as plataformas testadas.

2.1.1 CLASSIFICAÇÃO

As técnicas de Realidade Aumentada sem marcadores podem ser classificadas quanto a diversos aspectos, como por exemplo:

- Quanto ao número de câmeras necessárias para o rastreamento:
  - Monocular: apenas uma câmera é necessária;
  - Estéreo: necessita de mais de uma câmera registrando a cena simultaneamente.
- Quanto à forma como se dá a inicialização do sistema:
  - Inicialização manual: necessita que o usuário informe alguns dados sobre a condição inicial da cena para que a aplicação inicie (como, por exemplo, a posição inicial do objeto a ser rastreado);
  - Inicialização automática: as informações necessárias para a execução do rastreamento são obtidas automaticamente pela aplicação.
- Quanto ao nível de conhecimento prévio sobre a cena:
  - Baseada em modelos: necessita que um modelo do objeto a ser rastreado seja previamente conhecido. Este modelo pode ser, por exemplo, um modelo 3D gerado por uma ferramenta de CAD ou uma imagem da textura do objeto;
  - *Structure from Motion*: técnicas de *Structure from Motion* são capazes de reconhecer a topologia do ambiente à medida que a aplicação é executada. Com isto, o posicionamento dos objetos pode ser feito mesmo sem que haja um modelo conhecido de algum objeto presente na cena.

Para o desenvolvimento das aplicações utilizadas como referência para este trabalho, foram escolhidas técnicas de rastreamento monocular, pois o custo associado à aplicação é menor, visto que é necessária a utilização de apenas uma câmera. Cabe ressaltar que a precisão do rastreamento monocular é adequada para aplicações de Realidade Aumentada. Foram também priorizadas as técnicas que permitem inicialização automática, pois isto facilita a utilização da aplicação por usuários leigos, público-alvo comum para aplicações *Web*. Por fim, decidiu-se pelo uso de técnicas baseadas em modelos. Esta última decisão se deve ao alto custo computacional envolvido na descoberta da topologia do ambiente através de imagens de apenas uma câmera. Este custo faz com que aplicações que utilizam *Structure from Motion* tenham, normalmente, um desempenho inferior às que utilizam rastreamento baseado em modelos.

As técnicas que se baseiam em modelos do objeto a ser rastreado podem ainda ser classificadas em técnicas baseadas em arestas, que utilizam a informação da geometria do objeto para localizá-lo na cena capturada pela câmera, ou baseadas em texturas, que utilizam informações da textura do objeto a ser rastreado com o mesmo fim [13]. As Figuras 3 e 4 ilustram exemplos de rastreamento baseado em arestas e texturas, respectivamente.

FIGURA 3. RASTREAMENTO BASEADO EM ARESTAS EM TEMPO REAL DE ESTRUTURAS COMPLEXAS: [14] NA ESQUERDA E [15] NA DIREITA

FIGURA 4. RASTREAMENTO BASEADO EM TEXTURA [16]

As técnicas escolhidas para este trabalho são baseadas em textura. Esta escolha se baseia em uma comparação de desempenho e robustez feita entre algumas técnicas baseadas em texturas e baseadas em arestas, tendo as primeiras se destacado nestes aspectos [13].

É uma característica comum às técnicas utilizadas neste trabalho a escolha de alguns pontos da textura do objeto a ser rastreado. Isto se deve ao fato de que comparar a textura completa do objeto para cada possível posição do mesmo no ambiente seria extremamente ineficiente, além de haver a possibilidade de falhar dependendo da distorção sofrida pelo objeto devido à visão em perspectiva. A escolha dos pontos a serem considerados deve ser feita de forma que estes pontos possam ser identificados mesmo quando observados sob diferentes pontos de vista. Estes pontos escolhidos são chamados de *keypoints* e, através do casamento de *keypoints* entre a textura conhecida do objeto a ser rastreado e os *keypoints* detectados na imagem capturada pela câmera, é possível identificar a posição do objeto no ambiente. A Figura 5 ilustra este processo. Este casamento pode ser feito de duas formas, explicadas nas subseções a seguir.

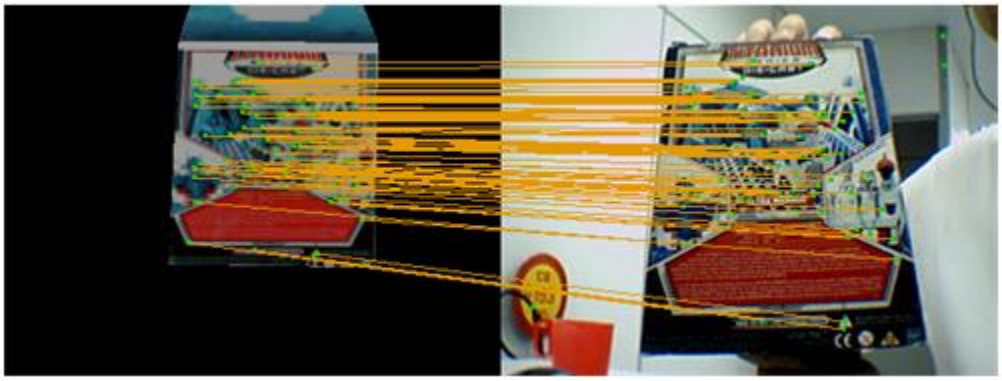

FIGURA 5. EXEMPLO DE CASAMENTO DE *KEYPOINTS*

Técnicas baseadas em descritores

Este tipo de técnica armazena, para cada *keypoint*, um conjunto de características extraídas da imagem ao redor do mesmo. Estas características devem ser invariantes a transformações de perspectiva para que o objeto possa ser reconhecido sob diferentes pontos de vista. Estas características são armazenadas em vetores de alta dimensionalidade, de forma que o casamento de *keypoints* possa ser feito utilizando como referência a distância euclidiana entre os vetores correspondentes: vetores mais próximos indicam *keypoints* mais parecidos, ou seja, com maior probabilidade de corresponderem ao mesmo ponto do objeto rastreado.

Para fazer a detecção do objeto na cena, portanto, é necessário identificar os *keypoints* na imagem obtida pela câmera e compará-los a uma base de dados que contém os descritores dos *keypoints* da textura do objeto, conhecida previamente. Para tornar a busca pelo *keypoint* mais próximo mais eficiente, esta base de dados pode ser organizada na forma de uma kd-tree e, para cada *keypoint*, procurar os dois pontos mais próximos na árvore. O potencial correspondente então é escolhido como sendo o ponto mais próximo, enquanto o segundo ponto mais próximo é utilizado para tentar minimizar o número de correspondências espúrias. Para isto, as distâncias entre estes pontos e o *keypoint* obtido da imagem atual são comparadas. Se estas distâncias forem muitos próximas (isto é, a razão entre elas for pequena), isto indica um possível casamento espúrio e não deve ser considerado para o cálculo da pose da câmera. Isto se deve ao fato de que, como os vetores possuem muitas dimensões, é bastante improvável que haja dois pontos similares na base de dados.

Duas técnicas utilizadas nas aplicações que foram referência para este projeto se encaixam nesta categoria: a SIFT [9] e a SURF [11].

Técnicas baseadas em classificadores

Outra forma de resolver o problema de casamento de *keypoints* é modelando-o como um problema de classificação. Nesta abordagem, cada *keypoint* da textura previamente conhecida do objeto a ser rastreado representa uma classe, e encontra-se o casamento entre *keypoints* classificando os *keypoints* da imagem obtida pela câmera em uma das classes conhecidas.

A técnica utilizada neste trabalho que se enquadra nesta categoria é a Ferns [12]. Esta técnica utiliza um classificador bayesiano ingênuo para encontrar as correspondências de *keypoints*. O treinamento do classificador é feito da seguinte forma: para cada *keypoint* da textura do objeto, são geradas várias imagens contendo um retalho da textura ao redor do *keypoint* após a aplicação de várias transformações de perspectiva. Estas imagens servirão como instâncias da classe que representa aquele *keypoint*. Para cada imagem destas, então, são aplicados testes binários e os resultados são armazenados em um vetor binário que será utilizado como critério de classificação.

2.2. REALIDADE AUMENTADA NA WEB

Como foi dito anteriormente, a escolha da técnica de rastreamento a ser utilizada em um sistema de Realidade Aumentada é uma etapa importante do planejamento da aplicação. Quando a aplicação a ser desenvolvida visa divulgação via *Web*, outra etapa importante do planejamento é a escolha da plataforma de desenvolvimento que será utilizada. Alguns aspectos que podem ser considerados nesta escolha são:

- Portabilidade: algumas plataformas de desenvolvimento restringem o conjunto de sistemas operacionais sobre os quais a aplicação pode ser executada. Isto se deve, normalmente, a não existência de um *plug-in* para navegadores da *Internet* em alguns sistemas operacionais. Por causa disto, a escolha de uma plataforma inadequada pode fazer com que a aplicação perca portabilidade, uma das vantagens normalmente desejadas ao planejar sistemas de Realidade Aumentada para *Web*;
- Desempenho: é comum que aplicações que são executadas através da *Web* tenham um desempenho inferior, mesmo que levemente, ao que teriam se fossem executadas como uma aplicação *offline*;
- Popularidade: para o usuário, uma das vantagens de executar uma aplicação através do navegador é não precisar instalar a mesma em sua máquina, bastando acessar o *site* que a hospeda para utilizá-la. Isto, porém, só é possível caso o usuário tenha instalado o *plug-in* necessário para executar a aplicação desejada. Quanto mais popular a plataforma utilizada, isto é, quanto mais aplicações estiverem disponíveis na plataforma escolhida, mais provavelmente o usuário já terá o *plug-in* necessário instalado e, portanto, será mais cômoda a utilização da aplicação para ele.

Outros fatores que podem influenciar na escolha da plataforma são:

- Suporte ao acesso à câmera do computador, sem a qual não é possível capturar imagens da cena sobre a qual serão dispostos os objetos virtuais;
- Suporte à renderização 3D, facilitando a sobreposição dos objetos na cena na posição correta;
- Familiaridade com a linguagem de programação suportada pela plataforma, o que pode reduzir significativamente o tempo necessário para a conclusão do projeto.

Neste trabalho, foi escolhida a plataforma Silverlight como base para o desenvolvimento de uma aplicação de Realidade Aumentada sem Marcadores para comparação com os resul-

tados obtidos em um trabalho prévio, que abordou as plataformas Mammoth e OSAKit, conforme descrito em [19]. Esta escolha se baseou na portabilidade oferecida pela plataforma, que permite que uma mesma aplicação possa ser executada tanto via *Web* quanto em *desktop* ou mesmo em dispositivos móveis. Apesar de não ser ainda uma plataforma muito popular, as ferramentas oferecidas ao desenvolvedor, tal como a possibilidade de escolher uma linguagem de programação dentre as suportadas pelo *framework* .NET para o desenvolvimento da aplicação, podem atrair a atenção de mais desenvolvedores para a plataforma, aumentando a popularidade da mesma.

2.3. TRABALHOS RELACIONADOS

O desenvolvimento de aplicações de Realidade Aumentada para *Web* foi fortemente impulsionado pela criação da biblioteca FLARToolKit, uma das primeiras grandes iniciativas em direção à divulgação das aplicações de Realidade Aumentada nesta plataforma [20]. Esta biblioteca é uma adaptação da biblioteca de rastreamento baseado em marcadores ARToolKit [21] para ActionScript [22], a linguagem de *scripting* utilizada para o desenvolvimento de aplicações em Flash [23]. A biblioteca FLARToolKit, porém, é responsável apenas pela etapa de rastreamento de marcadores na cena, sendo necessária a utilização de outra biblioteca para fazer a renderização dos objetos virtuais, como, por exemplo, a Papervision3D [24], bastante utilizada para este fim. Aplicações desenvolvidas utilizando estas bibliotecas podem ser executadas diretamente no navegador *Web* do usuário, sendo necessária apenas a instalação do plug-in adequado (Flash Player).

Um exemplo de aplicação utilizando as bibliotecas citadas anteriormente é a ação publicitária criada pela General Eletric, em que o usuário posiciona um marcador em frente à câmera e, sobre ele são sobrepostas turbinas eólicas, com as quais o usuário pode interagir soprando no microfone, como ilustrado na Figura 6 [25].

FIGURA 6. REALIDADE AUMENTADA PARA INTERNET USANDO FLARTOOLKIT

Outra solução desenvolvida para a criação de aplicações de Realidade Aumentada na *Web* foi o *plug-in* Unifeye Viewer, uma iniciativa da empresa Metaio [26]. Este *plug-in* permite o desenvolvimento de aplicações de Realidade Aumentada sem marcadores, pois é capaz de rastrear a textura de objetos planares para o cálculo do posicionamento dos objetos virtuais.

Uma aplicação desenvolvida sobre esta plataforma é mostrada na Figura 7 [27]. Esta aplicação é similar à primeira, porém os objetos virtuais são renderizados sobre a capa de uma revista, em contraste ao marcador fiducial utilizado anteriormente.

FIGURA 7. APLICAÇÃO UTILIZANDO O PLUG-IN UNIFEYE VIEWER

Podem ser citadas ainda outras plataformas para desenvolvimento de sistemas de Realidade Aumentada na *Web*, como o plug-in D'Fusion@Home [28], criado pela companhia Total Immersion e a FLARE (Flash Augmented Reality Engine), solução baseada em Flash criada pela Imagination [29]. Estas plataformas também utilizam rastreamento sem marcadores para o posicionamento dos objetos virtuais.

3. IMPLEMENTAÇÃO

Neste trabalho foi escolhida a plataforma Silverlight [8] para o desenvolvimento de uma aplicação de Realidade Aumentada sem marcadores para que os resultados obtidos fossem comparados com os disponíveis em [14].

Silverlight é uma plataforma de desenvolvimento para a *Web* disponibilizada pela Microsoft. Ela utiliza como base a plataforma .NET, da mesma empresa, para o desenvolvimento de aplicações. Isto proporciona ao desenvolvedor uma série de vantagens, dais quais se pode destacar:

- Possibilidade de escolha da linguagem de programação: é possível escrever programas para Silverlight em qualquer linguagem suportada pela plataforma .NET;
- Gerenciamento de memória e coletor de lixo (*Garbage Collector*);
- Compatibilidade com os navegadores e sistemas operacionais mais usados atualmente;
- Possibilidade de executar a aplicação criada como aplicação *Web*, *desktop* ou até mesmo em dispositivos móveis.

O desenvolvimento de sistemas de Realidade Aumentada ainda é uma novidade para a plataforma, visto que apenas a partir do lançamento da versão Silverlight4 beta, em novembro de 2009, são oferecidos meios de acesso à câmera de vídeo do usuário.

O processo de criação de uma aplicação em Silverlight, ilustrado na Figura 8, pode ser resumido da seguinte forma: primeiramente o núcleo da aplicação é codificado em uma das linguagens aceitas pela plataforma. Em seguida este código é compilado utilizando o *Silverlight Software Development Kit* (Silverlight SDK), disponível na página oficial da plataforma [8]. O Silverlight SDK também é responsável por juntar o código-fonte da aplicação compilado às bibliotecas de ligação dinâmica (dlls) das quais a aplicação depende e aos demais recursos necessários para a execução da mesma em um único pacote de extensão .xap. Este pacote deve ser embutido em uma página HTML para que, então, a aplicação possa ser executada em qualquer navegador que possua o *plug-in* adequado.

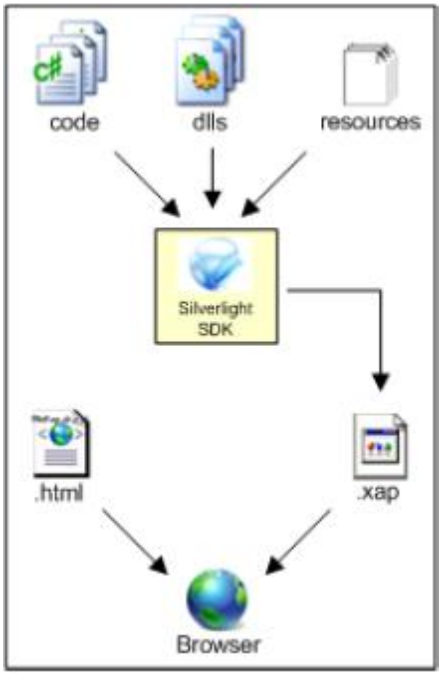

FIGURA 8. PROCESSO DE CRIAÇÃO DE APLICAÇÃO EM SILVERLIGHT

A aplicação criada teve por finalidade rastrear a capa de um livro e, a ele, sobrepor um objeto virtual. Com isto, visou-se medir a taxa de atualização média alcançada pela aplicação (em *frames* por segundo) e compará-la aos resultados obtidos utilizando as plataformas Mammoth e OSAKit, publicados em [14].

Para que a comparação entre as plataformas de desenvolvimento pudesse ser feita de forma justa, foi adotada a técnica de rastreamento sem marcadores baseada em classificadores Ferns, também adotada em [14].

A implementação da aplicação iniciou-se com a adaptação do código do rastreador Ferns disponibilizado em [30] para a linguagem de programação C#, uma das linguagens aceitas por Silverlight. Para que a aplicação possa ser executada via *Web*, porém, o código, assim como suas dependências, deve ser compilado para a plataforma Silverlight. Esta compilação impõe algumas restrições sobre o código, como, por exemplo, não permitir a utilização de código *unsafe* (isto é, códigos não gerenciados). Estas restrições têm por objetivo prevenir que o desenvolvedor crie aplicações que possam provocar danos ao computador do usuário através, por exemplo, de acesso indevido à memória. Outra restrição imposta pela plataforma impede que a aplicação tenha acesso direto aos arquivos armazenados no computador do usuário sem autorização explícita do mesmo. Para obedecer a estas restrições, foi necessário alterar os códigos-fonte das bibliotecas utilizadas pela aplicação.

A captura de imagens do ambiente através da câmera do usuário é feita de forma bastante simples através de um conjunto de funções disponibilizadas pela plataforma para este fim. Os passos que foram seguidos na aplicação implementada são descritos a seguir.

Primeiramente foi selecionado o dispositivo de entrada a ser utilizado. Isto foi feito utilizando uma instância da classe `CaptureSource`. O dispositivo de captura de vídeo padrão foi obtido através da função `CaptureDeviceConfiguration.GetDefaultVideoCaptureDevice()`. Este dispositivo foi utilizado para configurar a fonte de captura através da propriedade `VideoCaptureDevice`. O trecho de código da Tabela 1 mostra como isto pode ser feito em poucas linhas de código.

TABELA 1. CONFIGURAÇÃO DO DISPOSITIVO DE CAPTURA DE VÍDEO

```
// inicializa uma instância da classe CaptureSource
CaptureSource captureSource = new CaptureSource();

// configure o dispositivo de captura de vídeo com a opção padrão
captureSource.VideoCaptureDevice = CaptureDeviceConfiguration.GetDefaultVideoCaptureDevice();
```

Em seguida foi iniciada a captura da câmera, usando a função `Start` da fonte de captura. Esta função, porém, só pode ser executada caso o usuário tenha dado permissão à aplicação de utilizar a câmera. Para checar se o programa tem permissão para acessar a câmera, foi utilizada a propriedade `AllowedDeviceAccess` da classe `CaptureDeviceConfiguration`. Se esta permissão não foi dada ainda, pode ser requisitada ao usuário através da função `RequestDeviceAccess`, da mesma classe. A Tabela 2 mostra como isto foi feito na implementação deste projeto.

TABELA 2. INICIALIZAÇÃO DA CAPTURA DE IMAGENS PELA CÂMERA

```
// checa se a aplicação possui permissão de acesso à câmera e, em caso negativo, solicita-a
if (CaptureDeviceConfiguration.AllowedDeviceAccess ||
CaptureDeviceConfiguration.RequestDeviceAccess()) {

// inicia a captura pela câmera
captureSource.Start();
}
```

Vale ressaltar que a requisição de acesso à câmera só pode ser feita como reação a uma ação voluntária do usuário, como o clique em um botão. Por este motivo, o trecho de código disposto acima teve que ser colocado na função de tratamento de um clique de um botão.

O processamento das imagens capturadas pela câmera foi feito de forma assíncrona. Quando a função da fonte de captura `CaptureImageAsync` é chamada, é iniciada a captura de uma imagem da câmera e, quando esta termina, é gerado um evento `CaptureImageCompleted` pela fonte de captura. Para tratar a imagem que foi capturada, foi associada uma função a este evento. Estes passos são ilustrados no código da Tabela 3.

TABELA 3. CAPTURA ASSÍNCRONA DA IMAGEM DA CÂMERA

```
// associa uma função ao evento CaptureImageCompleted
captureSource.CaptureImageCompleted += ProcessImage;

// captura uma imagem assincronamente
captureSource.CaptureImageAsync();
```

No exemplo acima, a função `ProcessImage` é responsável por realizar a detecção da textura a ser rastreada na imagem e, a partir dos resultados obtidos, calcular o posicionamento dos objetos virtuais. Este cálculo foi feito utilizando o método *Robust Planar Pose* (RPP), proposto em [31].

Para que a captura assíncrona fosse feita a cada quadro, foi associada uma função ao evento `Rendering`, gerado pela classe `CompositionTarget` sempre que é necessário renderizar um novo quadro. O código referente a estes passos é mostrado na Tabela 4.

TABELA 4. CAPTURA ASSÍNCRONA ASSOCIADA À RENDERIZAÇÃO DE QUADROS

```
// associa uma função ao evento Rendering de CompositionTarget
CompositionTarget.Rendering += ProcessFrame;

// função ProcessFrame
private void ProcessFrame(object sender, EventArgs e)
{
// verifica se a captura de imagens foi iniciada
if (captureSource != null && captureSource.State == CaptureState.Started)
{
// captura uma imagem assincronamente
captureSource.CaptureImageAsync();
}
}
```

Silverlight não oferece, por padrão, suporte a renderização de objetos 3D. Para este fim, foi utilizada a biblioteca Balder [32], versão 0.8.8.5 alfa. Esta biblioteca, por estar ainda em estágio alfa de desenvolvimento, está sujeita a grandes modificações em curtos períodos de tempo, o que pode dificultar o trabalho do desenvolvedor que a utilizar.

Como mencionado anteriormente, para fins de comparação foram usados dois estudos de caso adicionais, além do explicado acima. Detalhes sobre a sua implementação podem ser encontrados em [33].

4. RESULTADOS

A primeira aplicação utilizada como referência para comparação foi desenvolvida utilizando a plataforma Mammoth. Esta aplicação utiliza um modelo cliente-servidor para realizar o rastreamento. Duas técnicas de rastreamento foram testadas sobre esta plataforma: SIFT, otimizada para utilizar aceleração na placa de processamento gráfico (*Graphics Processing Unit*, GPU) e Ferns. A Figura 9 ilustra o funcionamento da aplicação. Nela, a capa de um livro está sendo rastreada e, sobre ela, é passado um vídeo musical.

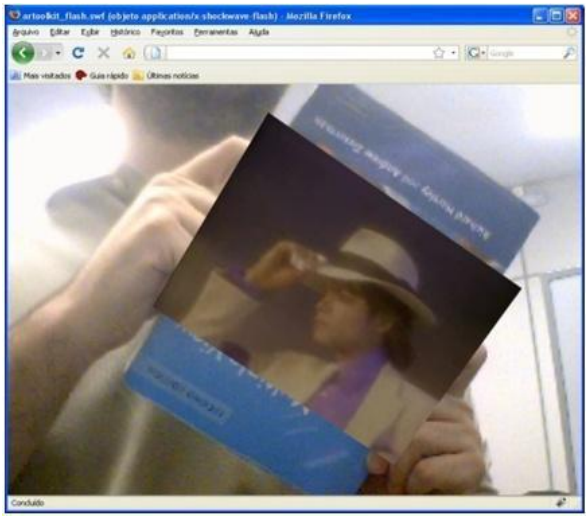

FIGURA 9. APLICAÇÃO EM MAMMOTH

A aplicação mostrou um bom desempenho quando apenas um usuário estava conectado ao servidor. A abordagem cliente-servidor, porém, mostrou-se ineficiente quando vários usuários se conectavam ao mesmo tempo (na aplicação testada, cerca de cinco usuários era o suficiente para que o atraso na resposta do servidor fosse significativo). Isto se deve à grande carga de processamento imposta em uma mesma máquina.

Outra aplicação utilizada como referência foi desenvolvida utilizando a plataforma OSAKit. Nesta aplicação, o rastreamento é feito na própria máquina do usuário, o que evita a sobrecarga de processamento do servidor. Novamente duas técnicas foram utilizadas para fins de comparação: SURF e Ferns. A execução da aplicação é ilustrada na Figura 10, em que a capa de um livro é rastreada e, sobre ela, é renderizado um bule.

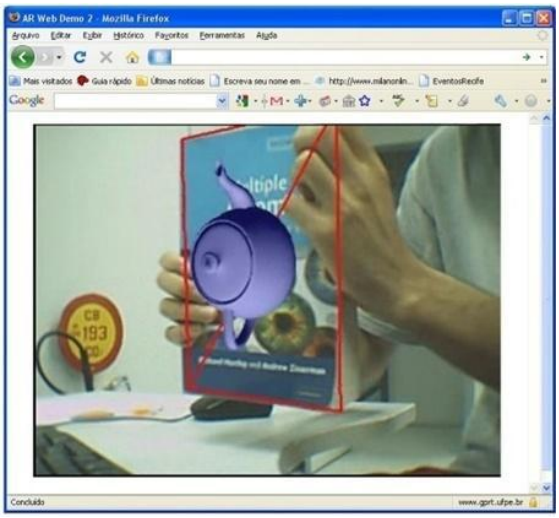

FIGURA 10. APLICAÇÃO EM OSAKIT

Esta aplicação foi testada utilizando um computador *desktop* com processador Intel Core2 Quad 2.66 GHz, 4 GB de memória RAM, placa gráfica NVIDIA GeForce 9800 GX2 com 512 MB de memória e resolução de tela de 1440 x 900 pixels. Sob estas condições, a aplicação atingiu uma taxa de atualização em torno de 20 *frames* por segundo quando foi utilizada a técnica de rastreamento SURF e de cerca de 25 *frames* por segundo ao usar Ferns.

A Figura 11 mostra a aplicação implementada em Silverlight. Nela, o modelo de um carro foi renderizado sobre a capa de um livro para indicar a posição que foi calculada para o mesmo.

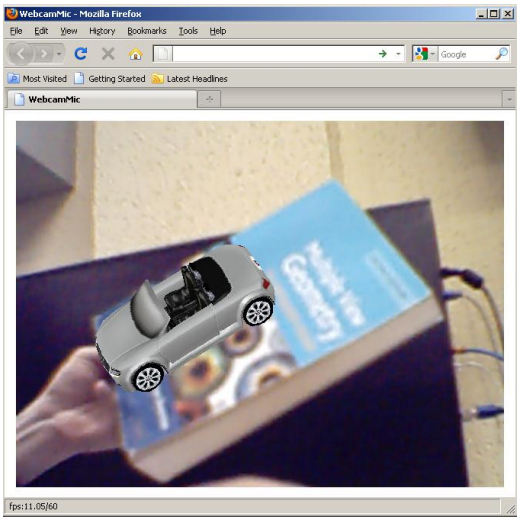

FIGURA 11. APLICAÇÃO EM SILVERLIGHT

Esta aplicação foi testada sob as mesmas condições da aplicação feita com OSAKit, tendo atingido uma taxa de atualização em torno de 10 fps.

A Tabela 5 mostra de forma resumida os resultados obtidos pelas aplicações desenvolvidas com rastreamento do lado do cliente. Nela, podemos ver que a técnica Ferns apresentou uma taxa de atualização melhor do que a SURF, porém, devido à grande quantidade de dados relativos ao treinamento do classificador que devem ser carregados, o tamanho da aplicação fica significativamente maior do que a da aplicação utilizando SURF, o que tem conseqüência direta no tempo necessário para inicializar a aplicação.

TABELA 5. COMPARATIVO ENTRE APLICAÇÕES TESTADAS

	OSAKit	OSAKit	Silverlight
Técnica de rastreamento	SURF	Ferns	Ferns
Taxa de atualização	20 fps	25 fps	10 fps
Tamanho da aplicação	1.8 MB	20 MB	15 MB
Tempo de carregamento	1 min	10 min	8 min

5. CONCLUSÃO E TRABALHOS FUTUROS

Este trabalho apresentou algumas soluções de desenvolvimento de aplicações de Realidade Aumentada voltadas para a *Web.* Foram adotadas duas abordagens diferentes quanto à localização do rastreamento: o rastreamento do lado do servidor, que, apesar de ter como vantagem o baixo tempo de carregamento, mostrou problemas de escalabilidade, o que pode ser um fator limitante para muitas aplicações, e o rastreamento do lado do cliente, que, apesar do alto tempo de carregamento, mostrou-se mais promissor por não ter apresentado o problema de escalabilidade notado com a outra abordagem.

A plataforma OSAKit mostrou, como principal vantagem, a possibilidade de utilizar uma aplicação desenvolvida para *desktop* na *Internet*, sem necessidade de alterar o código-fonte ou recompilar a mesma. Por não apresentar perda de desempenho ao embutir a aplicação em um navegador, esta plataforma mostrou-se interessante para o desenvolvimento de aplicações em que o tempo de resposta é importante, como é o caso de aplicações de Realidade Aumentada. Como desvantagens, podem ser citadas a restrição da aplicação ao sistema operacional Windows, perdendo, com isso, portabilidade, um dos principais benefícios de se desenvolver uma aplicação para a *Web*, e também o fato de que é necessário ter instalado o *plug-in* do OSAKit, pouco difundido atualmente, para executar a aplicação.

A terceira solução testada, Silverlight, mostrou grande potencial, visto que a aplicação roda independente da plataforma e utiliza a linguagem C#, facilitando a utilização da mesma infraestrutura de *software* tanto para *desktop* quanto para a *Web.* Por outro lado, esta solução ainda necessita de melhoras no desempenho para atingir os requisitos temporais de uma aplicação de Realidade Aumentada.

Como trabalhos futuros, pode-se citar:

- Realizar um estudo de como resolver o problema de escalabilidade da abordagem utilizando rastreamento do lado do servidor;
- Desenvolver uma solução com rastreamento baseado em descritores, como SIFT ou SURF, em Silverlight para comparação;
- Desenvolver uma solução com rastreamento do lado do cliente em Flash, que é, atualmente, a plataforma mais popular, para comparação com as demais.

[1] Milgram, P., Takemura, H., Utsumi, A., Kishino, F. **Augmented Reality: A class of displays on the reality-virtuality continuum**. Telemanipulator and Telepresence Technologies, SPIE, v. 2351, p. 282-292.

[2] ARToolKit Home Page. http://www.hitl.washington.edu/artoolkit, acessado em julho de 2010.

[3] Metaio: Media gallery. http://www.metaio.com/media-press, acessado em julho de 2010.

[4] Ray-Ban Official Web Site | Virtual Mirror. http://www.ray-ban.com/usa/science/virtual-mirror, acessado em julho de 2010.

[5] We Are Autobots -Transformers II: Revenge of the Fallen. http://www.weareautobots.com, acessado em julho de 2010.

[6] Mammoth Server. http://mammothserver.org, acessado em julho de 2010.

[7] OSAKitAbout. http://www.osakit.com, acessado em julho de 2010.

[8] The Official Microsoft Silverlight Site. http://www.silverlight.net, acessado em julho de 2010.

[9] Lowe, D. **Distinctive Image Features from Scale-Invariant Keypoints**. IJCV, v. 60 (2), p. 91-110.

[10] Wu, C. **SiftGPU: A GPU Implementation of Scale Invariant Feature Transform (SIFT)**. http://cs.unc.edu/~ccwu/siftgpu, acessado em julho de 2010.

[11] Bay, H., Ess, A., Tuytelaars, T., Van Gool, L. **SURF: Speeded Up Robust Features**. CVIU, v. 110 (3), p. 346-359.

[12] Ozuysal, M., Calonder, M., Lepetit, V., Fua, P. **Fast Keypoint Recognition using Random Ferns**, TPAMI, v. 32(3), p. 448-461

[13] Lima, J., Simões, F., Figueiredo, L., Teichrieb, V., Kelner, J. **Online Monocular Markerless 3D Tracking for Augmented Reality**. Abordagens Práticas de Realidade Virtual e Aumentada, p. 1-30.

[14] Drummond, T., Cipolla, R. **Real-Time Visual Tracking of Complex Structures**, IEEE Transactions on Pattern Analysis and Machine Intelligence, v. 24(7), p. 932-946.

[15] Wuest, H., Vial, F., Stricker, D. **Adaptive Line Tracking with Multiple Hypotheses for Augmented Reality**, Proc. International Symposium on Mixed and Augmented Reality, p. 62-69.

[16] Benhimane, S., Ladikos, A., Lepetit, V., Navab, N. **Linear and Quadratic Subsets for Template-Based Tracking**, Proc. IEEE Conference on Computer Vision and Pattern Recognition, 6 p.

[17] Lima, J., Simões, F., Figueiredo, L., Teichrieb, V., Kelner J., Santos, I. **Model Based 3D Tracking Techniques for Markerless Augmented Reality**, SVR, SBC, Porto Alegre, 2009, p. 37-47.

[18] Lima, J., Simões, F., Figueiredo, L., Teichrieb, V., **Kelner, J. Model based markerless 3D tracking applied to augmented reality**. SBC JIS. (aceito para publicação)

[19] Lima, J., Pinheiro, P., Teichrieb, V., Kelner, J. **Markerless Tracking Solutions for Augmented Reality on the Web**, SVR, SBC, 2010.

[20] FLARToolKit. http://www.libspark.org/wiki/saqoosha/FLARToolKit/en, acessado em julho de 2010.
[21] ARToolKit. http://www.hitl.washington.edu/artoolkit, acessado em julho de 2010.
[22] ActionScript Technology Center. http://www.adobe.com/devnet/actionscript, acessado em julho de 2010
[23] Adobe Flash Professional CS5. http://www.adobe.com/products/flash, acessado em julho de 2010
[24] Papervision3D. http://code.google.com/p/papervision3d, acessado em julho de 2010.
[25] GE | Plug Into the Smart Grid. http://ge.ecomagination.com/smartgrid/#/augmented_reality, acessado em julho de 2010.
[26] Metaio Unifeye Viewer 2.0. http://www.metaio.com/products/viewer, acessado em julho de 2010.
[27] GE and Popular Science. http://ge.ar-live.de, acessado em julho de 2010.
[28] D'Fusion@Home. http://www.t-immersion.com/en,do-it-at-home,34.html, acessado em julho de 2010.
[29] Imagination – Augmented Reality for Flash. http://www.imagination.at/en/?Products:Augmented_Reality_for_Flash, acessado em julho de 2010.
[30] Computer Vision Laboratory Software. http://cvlab.epfl.ch/software, acessado em julho de 2010.
[31] Schweighofer,G.,Pinz, A. **Robust Pose Estimation from a Planar Target**, *TPAMI, v. 28(12)*, p. 2024-2030.
[32] Balder. http://balder.codeplex.com, acessado em julho de 2010.
[33] Lima, J. Realidade Aumentada Sem Marcadores Multiplataforma Utilizando Rastreamento Baseado em Modelo. Dissertação de Mestrado, Universidade Federal de Pernambuco, 2010. 89p.