UNIVERSIDADE FEDERAL DE PERNAMBUCO
GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
CENTRO DE INFORMÁTICA

ESTUDO SOBRE RASTREAMENTO BASEADO EM CARACTERÍSTICAS NATURAIS PARA APLICAÇÕES DE REALIDADE AUMENTADA EM DISPOSITIVOS MÓVEIS

Trabalho de Graduação

Aluno: Guilherme Reis da Rocha Borba Marques (grrbm@cin.ufpe.br)
Orientadora: Veronica Teichrieb (vt@cin.ufpe.br)
Co-orientadores: João Paulo Silva do Monte Lima (jpsml@deinfo.ufrpe.br)
Rafael Alves Roberto (rar3@cin.ufpe.br)

Fevereiro de 2015

Resumo

Este trabalho tem como objetivo avaliar diferentes maneiras de realizar rastreamento baseado em características naturais para dispositivos móveis. Uma solução é autônoma, ou seja, utiliza apenas o dispositivo móvel, e a outra solução transfere parte do processamento para um servidor. As duas soluções foram implementadas para a plataforma Android e utilizam o OpenCV, que é uma biblioteca de visão computacional. Além disso, a solução autônoma utiliza a biblioteca Intel TBB, que permite a paralelização do processamento. A descrição das técnicas utilizadas, bem como os detalhes de como foram implementadas são apresentados ao longo deste trabalho. Finalmente, um estudo comparativo das técnicas é apresentado, tendo como base testes realizados num dispositivo Android recente.

Agradecimentos

Gostaria de agradecer a todo mundo que me ajudou a vencer essa guerra. Primeiramente, agradeço a minha família pelo imenso apoio que dão aos meus estudos, tentando sempre dar todas as condições para que eu atinja um bom desempenho.

Agradeço muito a minha turma de Ciência da Computação 2010.2, cada um ficou marcado na minha memória. Cada ajuda, cada voto de confiança, cada momento de descontração.

Agradeço em especial a duas pessoas: Sabóia e Roberta.

Sabóia, meu irmãozinho que ganhei na graduação. A fé desse cara em si próprio e nos amigos é indescritível. Sempre que me deparo com algum desafio intelectual eu lembro do que você me ensinou. Sempre me pergunto: "O que sabóia diria ?" E acho que vou levar esse dom comigo pro resto da vida. Presente seu. Muito obrigado.

Roberta, que se tornou uma pessoa muito especial para mim. Sempre que precisei de força pude contar com você, e você sempre acreditou em mim. Foi tão consistente com isso que eu mesmo passei a acreditar. Aprendo muito sobre tudo convivendo com você. Com seu apoio, muitas coisas se tornaram possíveis. Obrigado mesmo, de coração.

Agradeço a Ermano, Alberto, Vitinho, Antonio, Sotero, Ligeiro... e todo mundo! Eu sei muito bem a diferença que ter amigos como vocês fez na minha vida !

Agradeço ao Centro de Informática por essa oportunidade que para mim foi única. O desafio e o aprendizado foram imensos. Saio dessa experiência completamente transformado. Queria deixar meus agradecimentos aos professores que mais me motivaram e conseguiram cultivar em mim o interesse pela área: Silvio Melo, Fernando Castor e Paulo Gonçalves.

Por fim, agradeço a minha orientadora Veronica Teichrieb pela oportunidade de trabalhar nesse grupo fantástico que é o Voxar Labs, a meu co-orientador João Paulo (Jonga), pela paciência que teve comigo e também pelo aprendizado constante que me proporcionou desde o início, e a meu (também) co-orientador Rafael Alves por oferecer suporte sempre que precisei, e tirar minhas dúvidas com toda calma e atenção.

Sumário

1. INTRODUÇÃO 7
1.1. OBJETIVOS 7
1.2. ESTRUTURA DO DOCUMENTO 8
2. RASTREAMENTO BASEADO EM CARACTERÍSTICAS NATURAIS PARA DISPOSITIVOS MÓVEIS 9
2.1. ABORDAGENS AUTÔNOMAS 10
2.2. ABORDAGENS DISTRIBUÍDAS 15
3. PROTÓTIPO PARA DISPOSITIVOS MÓVEIS 18
3.1. DESCRIÇÃO DA TÉCNICA 18
3.2. ABORDAGEM AUTÔNOMA 21
3.3. ABORDAGEM DISTRIBUÍDA 22
4. ESTUDO DE CASO E RESULTADOS 24
4.1. AVALIAÇÃO DA ABORDAGEM AUTÔNOMA 24
4.2. AVALIAÇÃO DA VERSÃO DISTRIBUÍDA 26
5. CONCLUSÃO E TRABALHOS FUTUROS 28
REFERÊNCIAS BIBLIOGRÁFICAS 29

Índice de Figuras

Figura 1. Esquemas de diferentes formas de distribuição de carga de processamento em aplicações de RA. 10

Figura 2. Uma forma (contorno) esboçada à mão e uma forma impressa reconhecida, rastreada, e aumentada com o Nestor [22]. 13

Figura 3. Rastreamento ocorrendo com auxílio do PatchTracker, como em [11]. Apesar de grande oclusão, a imagem é rastreada. 14

Figura 4. PTAM rodando em um iPhone 3G. Um mapa é criado sobre o livro (a), expandido para aproximadamente 220 pontos (b), e então utilizado para inserir flores virtuais na cena (c). 14

Figura 5. Dispositivo iPAQ com um modelo de teste, em uma demonstração em ambiente aberto 15

Figura 6. Bordas de um livro juntamente com sua identificação mostradas na tela do *smartphone*, como mostrado em [Streaming Mobile Augmented Reality on Mobile Phones]. 16

Figura 7. Visão geral do arcabouço de rastreamento distribuído proposto em [27]. 17

Figura 8. Resultado do casamento com ORB em tempo real. Linhas verdes representam casamentos válidos e círculos vermelhos indicam pontos que não foram casados. 19

Figura 9. Paralelização das etapas do *pipeline* de rastreamento na versão autônoma implementada. 21

Figura 10. Fluxo dos dados na versão cliente-servidor implementada. .. 23

Figura 11. Captura de tela do rastreamento paralelizado por 4 *threads*. As bordas azuis são o conteúdo virtual inserido na cena. 24

Figura 12. Resultado comparativo das medições de tempo (em ms) de cada uma das etapas do rastreamento entre a versão sem otimização e as versões desenvolvidas. 25

Índice de Tabelas

Tabela 1 Resultado comparativo das medições de memória e armazenamento, ambas em MB. 26

Tabela 2. Resultado das medições de tempo (em ms) da extração de características e de um quadro completo na versão cliente/servidor. 26

Tabela 3. Resultados da versão cliente/servidor quanto ao consumo de memória durante a execução e armazenamento, ambos em MB. 27

1. INTRODUÇÃO

Realidade Aumentada (RA) descreve aqueles sistemas que integram elementos virtuais a visualizações do mundo real (como por exemplo, utilizando a imagem capturada pela câmera) em tempo real. Objetos 3D são sobrepostos a imagens do ambiente real de forma a dar a sensação de fazerem parte da cena, adaptando-se em tempo real a mudanças no ambiente. O maior foco dos pesquisadores da área de RA, atualmente, está no processamento digital de vídeos transmitidos ao vivo. Esses vídeos são "aumentados" com a inserção de objetos 3D gerados por computação gráfica. Algumas aplicações de RA apresentam restrições, como por exemplo liberdade de movimentos do usuário ou o uso de dispositivos leves e compactos. Essas restrições são satisfeitas por dispositivos móveis modernos, tais como *smartphones* e *tablets*. Por essa razão, projetos de RA com foco direcionado para esses dispositivos vêm se tornando mais populares [1].

Desenvolver aplicações de RA para dispositivos móveis possui a vantagem da mobilidade, mas também existem entraves causados pelas restrições de memória, processamento e armazenamento desses dispositivos. A causa desses problemas é o fato de que vários dos algoritmos de rastreamento [2], que são uma parte essencial para o bom funcionamento das aplicações de RA, possuem alto custo computacional.

No contexto do desenvolvimento de aplicações em RA, o presente trabalho pretende estudar o rastreamento baseado em características naturais em dispositivos móveis. Para poder fazer o registro, isto é, o perfeito casamento dos objetos virtuais com a cena real, é preciso realizar um bom rastreamento dos objetos do mundo real. Uma forma bastante comum de se alcançar esse objetivo é utilizando marcadores fiduciais planares, que são elementos adicionados ao ambiente de forma intrusiva que contêm um padrão gráfico fácil de ser reconhecido. Entretanto, em muitas situações é mais vantajoso utilizar as próprias características naturais do mundo real, sem adicionar novos elementos a ele [3]. Realizar rastreamento baseado em características naturais em um dispositivo móvel representa um desafio, pois a detecção e rastreamento dessas características no ambiente é uma tarefa complexa.

1.1. OBJETIVOS

Este trabalho de graduação apresenta duas soluções, ambas para a plataforma Android. As duas utilizaram OpenCV [4], uma biblioteca de visão computacional que possui versões tanto para PC quando para dispositivos móveis.

A primeira solução foi implementada paralelizando-se o pipeline de rastreamento, composto por quatro fases: extração de características, casamento, filtragem e cálculo de pose. Isso foi feito para dispositivos móveis com dois e quatro núcleos, dividindo-se a imagem de entrada da câmera em duas e quatro partes, respectivamente. As etapas do pipeline de rastreamento

foram, então, executadas em paralelo, e obteve-se como resultado uma aceleração no rastreamento.

Foi utilizado para paralelizar as etapas do rastreamento a biblioteca Intel TBB [5], que é uma biblioteca baseada em templates na linguagem C++ que possui versões para Windows, Mac e Linux. A versão para Linux é compatível com Android (pois o sistema Android é baseado em Linux), e foi a utilizada na implementação.

Na segunda solução, a complexidade do rastreamento foi transferida para um servidor. Apenas a extração de características e computação dos descritores permaneceu no lado cliente. Esses dados são enviados para o servidor, que realiza o casamento, filtragem dos outliers e cálculo da pose, devolvendo para o cliente os vetores que compõem a mesma. A transmissão dos dados foi implementada utilizando sockets em Java.

As soluções foram avaliadas quanto ao desempenho em cada etapa do rastreamento e quanto ao uso de memória e armazenamento pela aplicação. Comparamos os resultados da versão autônoma sem otimizações, isto é, utilizando um único núcleo do dispositivo móvel, com os da versão paralelizada e os da versão distribuída.

1.2. Estrutura do Documento

O capítulo 2 descreve os conceitos básicos relacionados ao desenvolvimento de aplicações em RA que fazem uso do rastreamento baseado em características naturais para dispositivos móveis. Nesse contexto, esse capítulo apresenta diversos exemplos de sistemas que foram desenvolvidos e foram significativos para a área, tanto utilizando abordagens autônomas como distribuídas. O capítulo 3 explica a técnica de rastreamento utilizada no presente trabalho e entra em detalhes de como essas técnicas foram exploradas nas soluções implementadas. O capítulo 4 mostra os resultados da avaliação das duas aplicações e o comparativo com a versão sem otimizações. As conclusões e trabalhos futuros são apresentados no capítulo 5.

2. RASTREAMENTO BASEADO EM CARACTERÍSTICAS NATURAIS PARA DISPOSITIVOS MÓVEIS

Técnicas de detecção e rastreamento de objetos a partir de características naturais podem ser utilizadas em sistemas de RA [3]. Esses métodos usualmente dependem de dois tipos de pistas visuais: contornos e textura. Os *contornos* de um objeto consistem de sua silhueta e suas arestas internas [6]. A *textura* de um objeto leva em consideração propriedades tais como suavidade, rugosidade e regularidade de sua superfície [7]. Objetos formados principalmente por texturas suaves com brilho constante são conhecidos como sem textura. Caso a superfície de um objeto seja composta, em sua maior parte, por uma textura rugosa, então é comumente chamado de *texturizado*.

A detecção e rastreamento de características naturais precisam de conhecimento prévio sobre o objeto, ou seja, de um *modelo* [8]. Esse modelo pode ser codificado em diferentes formatos, sendo desenho assistido por computador (*computer-aided design* – CAD), nuvem de pontos 3D e segmentos de plano alguns formatos possíveis. As técnicas de rastreamento têm duas classificações: baseado em modelo e sem modelo. O rastreamento baseado em modelo é realizado utilizando um modelo do objeto alvo computado antecipadamente. Ele é capaz de lidar com situações em que o objeto ou a câmera se movem em relação um ao outro. Técnicas sem modelo também são conhecidas por resolverem o problema de Localização e Mapeamento Simultâneos *(Simultaneous Localization and Mapping* – SLAM). SLAM é o problema computacional que consiste de construir ou atualizar um mapa de um ambiente desconhecido e simultaneamente guardar a posição de um agente dentro do mapa. As técnicas que resolvem esse problema estimam tanto a pose da câmera quanto a geometria 3D da cena em tempo real. Nos métodos sem modelo, a câmera pode mover-se com relação à cena, porém é frequentemente assumido que a cena é rígida [9] [10]. Este trabalho utiliza uma técnica de rastreamento baseada em modelo.

Diferentes tarefas podem ser terceirizadas para um servidor [11]: Na Figura 1a tem-se o caso ideal, em que todo o trabalho é executado nativamente pelo cliente, que fica independente da infraestrutura do servidor. No outro extremo, tem-se aplicações de RA que são baseadas em uma abordagem de cliente fino com um mecanismo de comunicação "video-in/video out" para receber assistência de um servidor computacional (Figura 1d). Essa configuração não apenas necessita de uma comunicação quadro-a-quadro, como também requer o envio de imagens em ambas as direções, exigindo máximo desempenho da conexão de rede.

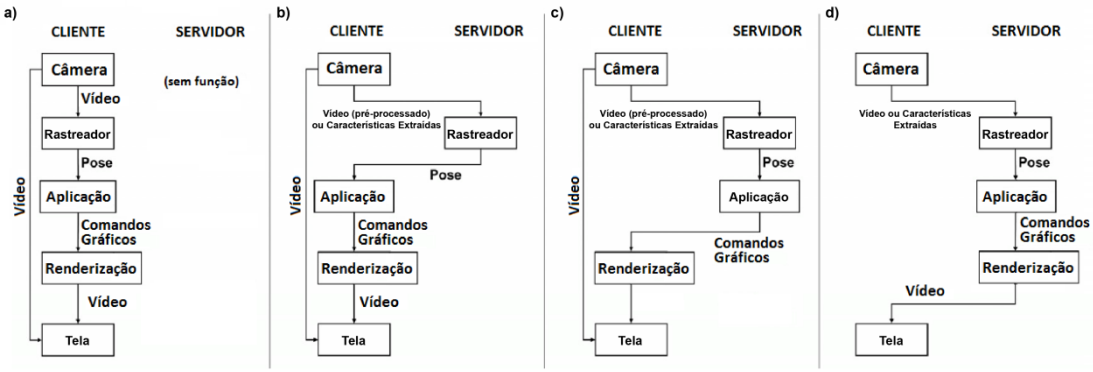

Figura 1. Esquemas de diferentes formas de distribuição de carga de processamento em aplicações de RA.

Considerando apenas sistemas de RA convencionais que utilizam uma única fonte de vídeo para rastreamento e display, o pipeline de processamento é composto das seguintes tarefas: aquisição de vídeo, rastreamento, computação da aplicação, renderização e display. Soluções escaláveis (ou seja, que possuem muitos clientes, muitos servidores etc.) requerem o uso cauteloso da banda disponível [12]. O constante envio e recebimento de vídeo não satisfaz tais restrições de banda. Uma melhor alternativa é o *streaming* de comandos gráficos de volta para o cliente (Figura 1c).

A abordagem ilustrada na Figura 1b transfere a tarefa de rastreamento para um servidor. Esse servidor deve receber vídeo comprimido e pré-processado ou características extraídas do vídeo a fim de realizar o rastreamento, e enviar de volta para o cliente a informação da pose. A vantagem deste método é que uma tarefa computacionalmente cara é transferida para o servidor, enquanto todos os detalhes da aplicação são processados exclusivamente pelo cliente. Desse modo, as dependências entre cliente e servidor são mínimas.

2.1. ABORDAGENS AUTÔNOMAS

As abordagens autônomas são aquelas que rodam exclusivamente no dispositivo móvel. Todo o processamento é realizado localmente no dispositivo, ou seja, nenhuma parte do *pipeline* de rastreamento ou renderização é transferida para um servidor externo.

A Metaio SDK [13] é uma plataforma para desenvolvimento de aplicações de RA disponível no mercado. Ela é bastante completa, oferecendo capacidade de reconhecimento e rastreamento de vários tipos de imagem, objeto ou ambiente do mundo real. Ela possui rastreamento baseado em características naturais de imagens planas, bem como rastreamento 3D do cenário, do tipo SLAM. Também possibilita rastreamento de face. Outra opção de mercado é o Qualcomm Vuforia, que também possui rastreamento de características naturais planas e tridimensionais, porém, no segundo caso, apenas detecta objetos em forma de cilindro ou cubo [14]. Esse SDK não possui rastreamento de face.

Outras opções comerciais bem conhecidas são o Layar SDK [15] e o D'Fusion Mobile, da Total Immersion [16]. Ambas possuem rastreamento baseado em características naturais, mas não dão suporte ao rastreamento 3D. O D'Fusion Mobile permite rastreamento de face baseado em características naturais.

Em [11] é descrito um método de mapeamento e rastreamento baseados em características naturais que é eficiente, robusto e permite rastreamento com três graus de liberdade em cenários *outdoor* para dispositivos móveis. O método é testado em um *smartphone* Asus P565 com uma CPU Arm Xscale que roda a 800MHz. Graus de liberdade de um sistema são o número de parâmetros dele que podem variar independentemente. Aqui isso representa a possibilidade de se rastrear objetos que podem ser rotacionados em torno dos três eixos do seu sistema de coordenadas. Considerando-se movimentos puramente rotacionais, o método cria um mapa panorâmico a partir das imagens da câmera. A abordagem é parecida com o SLAM. Para cada frame, a câmera é registrada baseando-se nas características do mapa. Em seguida, o mapa é extendido com novas características extraídas de diferentes direções que não haviam sido vistas ainda. De forma diferente do SLAM, esse método cria um mapa denso de características, que são mapeadas durante sua primeira observação e não são posteriormente refinadas. O método considera que os movimentos da câmera são puramente rotacionais, o que nem sempre é o caso. Entretanto, como mostrado em [17], a maior parte dos cenários *outdoor* possui a distância entre a câmera e os objetos muito maior que o movimento de translação involuntário que ocorre quando se rotaciona o dispositivo móvel, e erros que venham a ser causados devido a esse movimento de translação são, portanto, desprezíveis.

Em [18] são apresentadas peças fundamentais para a construção de um sistema de RA baseado em visão computacional. Técnicas são propostas para construir e manter um banco de dados de características naturais baseado em imagens de referência, bem como para fazer consultas nesse banco utilizando como entrada as imagens capturadas pela câmera. Os autores conseguiram diminuir bastante a quantidade de características naturais que precisam ser armazenadas no banco de dados do dispositivo para realizar um rastreamento bem sucedido.

Já em [19], é investigado como a direção da gravidade estimada a partir dos sensores inerciais do dispositivo móvel (um iPod touch 4G) pode beneficiar os estágios de uma aplicação de RA. A detecção, o cálculo de descritores, o casamento e o rastreamento de objetos planares, bem como a qualidade de renderização de objetos 3D, são melhorados. Quando o rastreamento é realizado na presença de superfícies aproximadamente verticais, são utilizados descritores alinhados com a gravidade (*Gravity-Aligned Feature Descriptors*, ou GAFDs) para melhorar a velocidade dos algoritmos de rastreamento. Quando as superfícies observadas são aproximadamente horizontais, a proposta é utilizar o vetor de gravidade para retificar a cena e então extrair suas características, resultando em descritores retificados em relação à gravidade (*Gravity-Rectified Feature Descriptors*, ou GREFDs). Utilizar esses descritores modificados possibilitam um rastreamento mais preciso e rápido.

Um novo método é apresentado em [20], que permite determinar um conjunto de características de um objeto especificamente para que este se torne detectável a partir de vários ângulos de visão. Várias visões distintas do objeto são geradas sinteticamente, e suas características são extraídas e reunidas em um banco de dados. Então essas características do banco de dados são casadas entre si, a fim de encontra-se o conjunto de descritores mais representativo daquele banco de dados. Esse processo ocorre durante a inicialização, e aumenta bastante a porcentagem de sucesso do rastreamento sem prejudicar a aplicação em tempo de execução.

O Nestor [21] é uma aplicação que roda localmente em um dispositivo móvel em tempo real. Ela foi testada no dispositivo Nokia N95, que possui sistema operacional Symbian. O sistema lê arquivos de formas ou utiliza um passo de aprendizado no qual o usuário mostra a forma nova para a câmera. A forma é analisada e inserida em uma biblioteca, que é usada para manter o conjunto de formas a serem rastreadas e suas propriedades, tais como os modelos associados a elas. Quando uma forma aprendida é reconhecida em tempo de execução, sua pose é estimada em cada quadro e podem ser inseridos os conteúdos aumentados, como mostrado na Figura 2. O sistema realiza reconhecimento e registro robustos, mantém rastreamento preciso e opera a taxas interativas em um dispositivo móvel. O reconhecimento de formas é feito analisando-se estruturas de contornos e gerando-se assinaturas a partir de suas concavidades. Essas assinaturas são invariantes a projeção. As concavidades são então utilizadas para extrair características para o cálculo da pose e rastreamento. A pose é então refinada minimizando-se o erro de reprojeção entre pontos amostrados em cada contorno da imagem e sua parte equivalente na biblioteca. Como o método de extração de características do *Nestor* é baseado em concavidades, o sistema é limitado a reconhecer formas não-convexas. Para alcançar esses resultados robustos, pelo menos duas concavidades são requeridas. Além disso, o passo de extração de forma é baseado em limiarização, o que implica que as formas utilizadas devem ter alto contraste em relação ao plano de fundo.

Figura 2. Uma forma (contorno) esboçada à mão e uma forma impressa reconhecida, rastreada, e aumentada com o Nestor [21].

Em [22], é apresentado um método de rastreamento visual 3D em um dispositivo móvel. Propõe-se uma abordagem fundamentada em rastreamento baseado em modelo utilizando um modelo 3D de um objeto na cena alvo. As comparações são feitas utilizando-se as arestas do modelo e dos objetos na cena. Uma pose de câmera é estimada atualizando-se iterativamente movimentos infinitesimais da câmera, explicitamente representados por transformações de corpo rígido 3D. O rastreamento visual 3D é implementado em um *smartphones* Android recente, o Samsung Galaxy S.

Em [11], foi criado o primeiro sistema de rastreamento autocontido capaz de calcular a pose perfeitamente a taxas de tempo real (30 Hz) a partir de características naturais usando somente a câmera do celular. O dispositivo escolhido para testes foi o Asus P552W. O único dado conhecido previamente nessa aplicação é o modelo texturizado. O restante do sistema é completamente geral e pode realizar tanto a inicialização quanto o rastreamento incremental de forma totalmente automática. Para realizar tal feito, os autores modificaram amplamente dois descritores de características de estado-da-arte, o SIFT e o Ferns, além de um rastreador baseado em modelo chamado de PatchTracker. O SIFT é conhecido por ser um excelente descritor de características, porém computacionalmente caro, enquanto que o Ferns é um classificador rápido, mas requer grandes quantidades de memória. O SIFT e o Ferns modificados e usados em conjunto possibilitaram um rastreamento satisfatório, mas com uma grande fraqueza que foi permitir apenas leves inclinações (em torno de 40 graus). Uma inclinação um pouco maior e o rastreamento era perdido. Esse problema foi resolvido integrando-se as duas modificações com o PatchTracker, que pode rastrear com grandes inclinações, próximas dos 90 graus (como ilustrado na Figura 3).

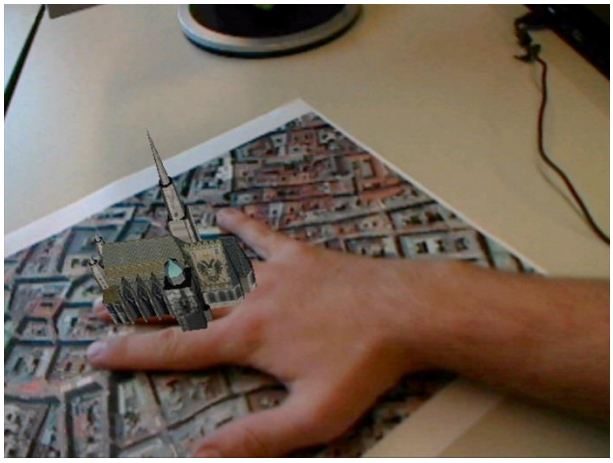

Figura 3. Rastreamento ocorrendo com auxílio do PatchTracker, como em [11]. Apesar de grande oclusão, a imagem é rastreada.

O trabalho descrito em [23] realiza a tarefa de detectar a posição de um *smartphone* em um ambiente previamente desconhecido. Mais especificamente, ele implementa um sistema de SLAM no iPhone 3G da Apple, que pode ser visto em funcionamento na Figura 4. Isso é obtido através de diversas modificações feitas em um sistema PTAM pré-existente, a fim de mitigar o impacto das deficiências do aparelho com relação a imagem da câmera. PTAM (Parallel Tracking and Mapping) é um sistema de rastreamento com câmera que resolve o problema do SLAM. Ele não requer marcadores, mapas pré-construídos, modelos conhecidos ou sensores inerciais. Esse dispositivo apresenta uma fraca capacidade de processamento, e uma câmera que opera em uma taxa baixa, tem um estreito campo de visão e possui um obturador rolante, isto é, as imagens são capturadas escaneando a cena rapidamente (em outras palavras, nem todas as partes da imagem são capturadas exatamente no mesmo instante). A presença de um obturador rolante acaba causando distorções nas imagens capturadas. Entretanto, uma série de mudanças no algoritmo utilizado mitigaram o impacto dessas limitações de imagem e processamento. Entre as otimizações feitas, destacam-se novas estratégias para a inicialização estéreo, detecção de características, compensação do obturador rolante, e poda do mapa.

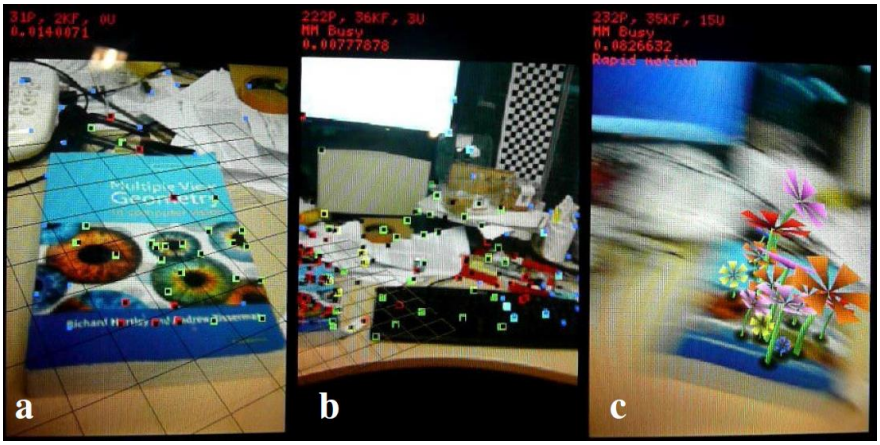

Figura 4. PTAM rodando em um iPhone 3G. Um mapa é criado sobre o livro (a), expandido para aproximadamente 220 pontos (b), e então utilizado para inserir flores virtuais na cena (c).

2.2. ABORDAGENS DISTRIBUÍDAS

As abordagens distribuídas são aquelas em que parte do processamento é transferida para um servidor. O dispositivo móvel pode simplesmente enviar as imagens capturadas para o servidor, ou então alguma etapa do *pipeline* de rastreamento (por exemplo, extração de características) pode ser realizada no próprio dispositivo, e as demais etapas são transferidas. O cliente então fica aguardando a resposta do servidor para dar prosseguimento a aplicação.

Uma implementação cliente-servidor para executar aplicações de RA exigentes em um dispositivo *Personal Digital Assistant* (PDA) foi criada em [24]. O PDA utilizado (um Compaq iPAQ H3800, iliustrado na Figura 5) não possui capacidade de processamento suficiente para rodar uma aplicação de RA no próprio dispositivo, e também não é equipado com suporte a hardware 3D. Esses problemas são resolvidos com a abordagem cliente-servidor, onde o rastreamento e a renderização ficam por total responsabilidade do servidor, e o PDA serve apenas como display. Os dados são transmitidos via WiFi.

Figura 5. Dispositivo iPAQ com um modelo de teste, em uma demonstração em ambiente aberto

Em [25] um sistema é apresentado que realiza reconhecimento contínuo de livros e capas de CDs em vídeo capturado em tempo real por um *smartphone*. O modelo utilizado foi um Nokia N95. O usuário pode apontar a câmera para um livro ou CD e ver sua identidade na tela do aparelho em menos de 1 segundo (Figura 6). Os limites do objeto são mostrados e corretamente rastreados em tempo real. Tanto a identidade do objeto como sua geometria são rapidamente recuperados via WiFi de um servidor que hospeda um banco de dados de 20.000 capas de livros e CDs. Enquanto o usuário navega pela cena, o sistema automaticamente reconhece novos objetos que aparecem na cena, sem que o usuário tenha que pressionar nenhuma tecla. O sistema realiza análise de movimento e seletivamente decide quando enviar a nova informação de consulta, ao invés de continuamente transmitir o vídeo pela rede sem fio.

Figura 6. Bordas de um livro juntamente com sua identificação mostradas na tela do *smartphone*, como mostrado em [Streaming Mobile Augmented Reality on Mobile Phones].

Em [26] é proposto um arcabouço de RA que é escalável em relação ao número de objetos sendo aumentados e fornece um nível elaborado de rastreamento. A escalabilidade é conseguida com um módulo de reconhecimento do lado do servidor. O nível elaborado de rastreamento é conseguido utilizando-se um algoritmo aperfeiçoado do lado do dispositivo móvel (um Android Nexus One). Esses dois lados se comunicam através de Wi-Fi convencional.

O problema de auto-localização é muito importante quando se quer realizar aplicações de RA em campo aberto. Em [27] é mostrado um sistema que utiliza informações do GPS embutido em um dispositivo móvel (Nokia N900), e também uma visão panorâmica online do ambiente, para auxiliar na orientação do usuário. O panorama é atualizado constantemente, sendo incrementado a medida que o usuário explora o ambiente. A pose é calculada em paralelo utilizando-se reconstrução 3D, possibilitando auto-localização em tempo real e registro em ambientes de larga escala. O arcabouço é configurado em duas partes: dispositivo móvel e computador do servidor remoto, conectados através de Wi-Fi. A informação visual é capturada com a câmera do dispositivo móvel e é requisitado ao usuário que selecione uma região de interesse na tela do dispositivo. Após esse passo, a informação contida na região de interesse é enviada ao servidor, no qual uma árvore de vocabulário [28] é adotada para o reconhecimento escalável. Alguns candidatos são recuperados dessa árvore e o melhor casamento é encontrado utilizando um método de casamento geométrico baseado no algoritmo *Progressive Sample Consensus* (PROSAC) [29]. Em seguida, o servidor envia a informação de rastreamento do objeto casado e os conteúdos de RA a serem aumentados no lado do dispositivo móvel. Com a informação de rastreamento recebida, o dispositivo móvel realiza o processo de detecção até encontrar o objeto alvo. Após a detecção, o rastreamento continua. Utilizando a pose estimada do objeto alvo a partir da detecção e do rastreamento, os conteúdos de RA são desenhados (aumentados). O arcabouço está ilustrado na Figura 7.

Figura 7. Visão geral do arcabouço de rastreamento distribuído proposto em [27].

Em [30] é realizado um esforço similar ao deste trabalho. Duas soluções para criar aplicações de RA em dispositivos móveis com baixa capacidade de processamento são propostas: uma abordagem local mais rápida, porém com menor precisão e robustez; e uma abordagem distribuída, que sofre com o atraso de comunicação e é difícil de implementar em aplicações que precisam rodar em tempo-real. Os autores comparam as duas abordagens em termos de desempenho em um Nexus One (NX1).

Um serviço interessante é o Metaio CVS [31]. Ele fornece um banco de dados na nuvem onde você pode guardar até 10.000 imagens que podem ser reconhecidas pela aplicação no dispositivo móvel através de conexão com a internet. O reconhecimento de objetos é feito integralmente no servidor. Dessa forma, o cliente não precisa fazer download de uma pesada configuração de rastreamento antes de iniciar a aplicação. Também é possível atualizar as imagens no banco de dados do servidor de forma trivial, utilizando um Web Service fornecido pela companhia Metaio.

3. PROTÓTIPO PARA DISPOSITIVOS MÓVEIS

Aqui é realizada uma descrição em alto nível da técnica de rastreamento utilizada nas duas soluções, explicando alguns conceitos do rastreamento em RA e, em seguida, falamos sobre os detalhes de como as soluções propostas foram implementadas, descrevendo as tecnologias utilizadas e explicando com maior profundidade as nuances de como a técnica explanada foi empregada nas soluções.

3.1. DESCRIÇÃO DA TÉCNICA

O rastreamento realizado no presente trabalho é baseado em características invariantes locais e é composto de algumas etapas. Primeiramente, são extraídas da imagem da cena características discriminativas e repetíveis . O tipo de característica utilizada neste trabalho é o ponto-chave, que pode ser entendido como um ponto de interesse na imagem. A detecção dos objetos é então realizada através do casamento das características extraídas da imagem de consulta com características que foram extraídas antecipadamente de imagens do objeto onde sua pose é conhecida. Uma alternativa para realizar esse casamento é utilizando-se descritores locais, que são vetores de alta dimensão que descrevem a vizinhança em torno da característica. O casamento é feito através da busca pelo vizinho mais próximo baseando-se na distância entre os vetores de alta dimensão. Após o casamento, uma lista de pares de características com propriedades semelhantes será obtida. Porém, muitas dessas características não serão casadas corretamente por vários motivos. Por exemplo, pode acontecer de a característica casada ter sido extraída de uma região do fundo da imagem. Então, é interessante descartar os casamentos que provavelmente são *outliers*, verificando se a razão entre as distâncias do primeiro e segundo vizinhos mais próximos é menor que um limiar. Essa etapa pode ser chamada de filtragem. Por último, é calculada a pose do objeto. A pose pode ser entendida como as informações de posição e orientação de um objeto em relação ao sistema de coordenadas da câmera. Para realizar seu cálculo, utilizamos o algoritmo *Efficient Perspective-n-Pose* (EPnP) [32] juntamente com o algoritmo *Random Sample Consensus* (RANSAC) [33] para remoção de *outliers*.

Existem alguns algoritmos para realizar a extração de características e compôr os descritores. O detector e descritor de pontos-chave SIFT [34] é um dos mais conhecidos, sendo utilizado com sucesso em diversas aplicações nos últimos anos. O SIFT pode ser utilizado para realizar tarefas tais como reconhecimento de objetos [34], costura de imagens [35] e mapeamento visual [11]. Entretanto, muitas vezes a carga computacional ao se utilizar o SIFT é muito alta, não sendo adequada para sistemas de tempo real ou para uso em dispositivos com pouco poder computacional, como celulares.

Outras técnicas surgiram, em uma tentativa de superar as limitações do SIFT. Uma das mais bem-sucedidas é o ORB [36], que é utilizada neste trabalho (Figura 8). Em comparação com o SIFT, o ORB possui desempenho

similar para realizar o casamento de características, é menos afetado por ruído de imagem, e é capaz de prover desempenho de tempo real.

O ORB foi construído em cima do detector de pontos-chave FAST [37] e do descritor BRIEF [38] (por isso o nome ORB – *Oriented FAST and Rotated BRIEF*), ambas técnicas que possuem boa performance e baixo custo.

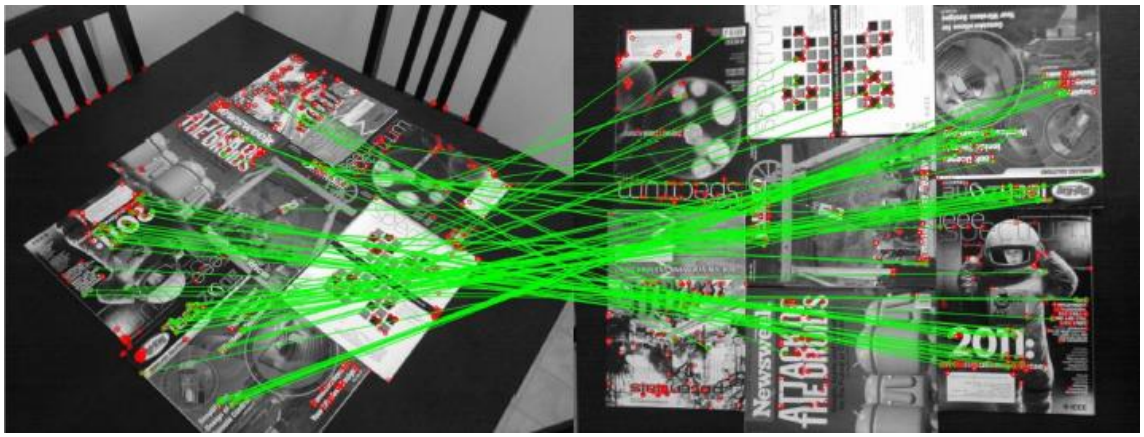

Figura 8. Resultado do casamento com ORB em tempo real. Linhas verdes representam casamentos válidos e círculos vermelhos indicam pontos que não foram casados.

Após a extração dos descritores de características com o ORB, é feito o casamento dos descritores do modelo com os descritores da imagem da câmera. O melhor candidato para cada ponto-chave é encontrado identificando-se o vizinho mais próximo no banco de dados de pontos-chave da imagem do modelo. O vizinho mais próximo de um ponto chave é definido como sendo o ponto-chave cujo descritor possui a menor distância Euclidiana para o descritor do ponto-chave em questão.

Entretanto, muitas características de uma imagem não vão ter nenhum casamento correto com as características do banco de dados, por não terem sido detectados na imagem do modelo ou por pertencerem ao plano de fundo da cena. Portanto, seria útil que houvesse uma forma de descartar características que não possuem nenhum casamento com o banco de dados. Uma alternativa seria usar um limiar global na distância para a característica mais próxima (apenas seriam considerados os casamentos entre características próximas o suficiente), mas essa abordagem não apresenta bom desempenho, pois descritores são muitas vezes bem discrepantes entre si, com alguns sendo muito mais discriminativos do que outros.

Uma medida mais efetiva, que é usada neste trabalho, é comparar a distância ao vizinho mais próximo com a distância ao segundo vizinho mais próximo [34]. Essa métrica funciona bem porque casamentos corretos precisam ter o vizinho mais próximo significativamente mais próximo que o casamento incorreto mais próximo para ser confiável. Caso o casamento fosse incorreto, provavelmente haveria uma grande quantidade de outros casamentos incorretos nas proximidades devido a alta dimensionalidade do espaço de características. Pode-se pensar no segundo casamento mais próximo como um indicativo da densidade de casamentos incorretos dentro dessa porção do espaço de

características, e ao mesmo tempo serve para identificar casos específicos de características ambíguas.

Para realizar o cálculo de pose, é utilizada uma função que resolve o problema PnP, isto é, o problema de estimar a pose da câmera dadas n correspondências 2D-3D. O problema PnP recebe explicitamente os parâmetros intrínsecos da câmera como uma de suas entradas, e estima os parâmetros extrínsecos sem precisar de qualquer estimativa prévia da pose. Parâmetros intrínsecos são aqueles que não dependem da posição da câmera, como distância focal e distorção da lente, e parâmetros extrínsecos são a rotação e a translação da câmera, que juntos compõem a matriz de pose.

Várias soluções existem para resolver o problema PnP. No presente trabalho é utilizada uma das soluções mais eficientes delas, chamada EPnP. Essa solução não-iterativa foi mais precisa e muito mais barata computacionalmente que todos os métodos não-iterativos anteriores [32], com pequena perda de precisão. Ela consegue calcular o resultado em $O(n)$ enquanto que todos os outros métodos ou são especializados para valores pequenos de n, ou são muito sensíveis a ruído, ou muito mais lentos. A solução EPnP representa todos os pontos como uma soma ponderada de quatro pontos de controle virtuais. Então o problema fica reduzido a estimar esses pontos de controle no sistema de coordenadas de câmera [3].

É necessário também tornar o cálculo de pose robusto. Normalmente, quando se vai estimar a pose, ocorrem correspondências 2D-3D espúrias (chamadas de *outliers*). Essas correspondências podem destruir a estimativa de pose mesmo no caso de haverem muitas correspondências corretas (*inliers*). Para diminuir a influência dos *outliers*, é utilizado um método conhecido como RANSAC.

O RANSAC é um algoritmo iterativo que utiliza uma sequência de pequenas amostras aleatórias das correspondências 2D-3D. A probabilidade de ter um *outlier* em uma amostra pequena é muito menor do que quando leva-se em consideração o conjunto de correspondências inteiro.

O método RANSAC funciona primeiro determinando o número de iterações a ser determinado pelo algoritmo. Esse número pode ser reduzido durante sua execução, caso a pose já esteja boa o suficiente. Após isso, um número fixo de correspondências é escolhido e a pose é calculada baseada nessas correspondências. Em seguida, todas as outras correspondências que não fazem parte da amostra são utilizadas para checar a qualidade da pose encontrada. Se o erro de reprojeção da correspondência for menor que um limiar, então a correspondência é um *inlier*. Caso contrário, é um *outlier*. Erro de reprojeção é a distância ao quadrado entre a projeção dos pontos 3D e suas coordenadas 2D medidas

3.2. ABORDAGEM AUTÔNOMA

Foi implementada uma abordagem que funciona exclusivamente no dispositivo móvel. Essa abordagem utilizou uma versão da biblioteca OpenCV de visão computacional para a plataforma Android. Essa versão é conhecida como OpenCV4Android [39]. Essa biblioteca permite o uso de várias funções de visão computacional na linguagem Java.

A fim de paralelizar o processamento do *pipeline* de rastreamento, foi necessário utilizar uma biblioteca de paralelização chamada de Intel Threading Building Blocks, ou TBB [5]. Essa biblioteca gerencia o paralelismo, dando suporte a programação multinúcleo. Ela tem como base a linguagem C++ e, portanto, para usá-la com o Android, é necessário utilizar o conjunto de ferramentas Android NDK [40], que permite que partes do código em Android seja implementado utilizando código nativo, em C ou C++.

Inicialmente, um modelo a ser rastreado é carregado. Logo após, inicia-se a execução da aplicação. A imagem da câmera é capturada e enviada para o módulo da aplicação que funciona com código nativo. Lá, a fim de otimizar o desempenho, as etapas do *pipeline* de rastreamento foram paralelizadas. Para tal, decidiu-se pelo procedimento de dividir, inicialmente, a imagem de entrada da câmera em duas ou quatro partes. Esse procedimento é exemplificado na Figura 9.

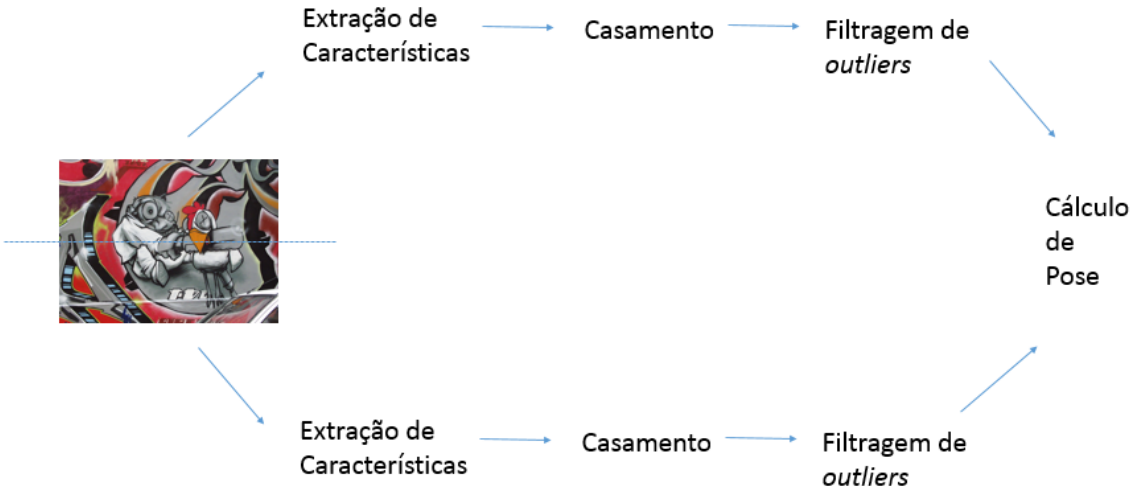

Figura 9. Paralelização das etapas do *pipeline* de rastreamento na versão autônoma implementada.

Antes de iniciar a aplicação, há uma etapa de inicialização em que a imagem do modelo tem suas características extraídas e seus descritores computados. Esses elementos são guardados. Então, a aplicação se inicia e começam as etapas do *pipeline* de rastreamento.

A extração de características é realizada para cada parte da imagem de entrada, e os descritores de cada parte são computados separadamente, em paralelo. Cada conjunto de características e descritores é guardado em sua própria estrutura separada.

Cada uma das estruturas contendo as características extraídas são então passadas para a próxima fase, o casamento com a base de dados, que ocorre em paralelo. Todos os casamentos válidos obtidos são colocados em uma estrutura única. Essa estrutura é utilizada como entrada para realizar o cálculo de pose.

O OpenCV4Android já garante internamente que o cálculo de pose será feito de forma paralela, não sendo necessário alterar esse processo. Uma vez calculada, a pose é enviada de volta para a parte em Java da aplicação, onde pode ser utilizada para desenhar sobre a imagem da câmera o conteúdo virtual desejado.

Existem duas formas de construir a aplicação descrita. A primeira forma é o modo desacoplado, no qual o fluxo da câmera é deixado livre, sem esperar. Uma vez capturada, a imagem da câmera é passada para uma função não bloqueante, e então a aplicação passa para o próximo quadro. O cálculo da pose é realizado paralelamente ao fluxo da câmera, e as poses vão sendo passadas para a *thread* principal a medida que forem calculadas. Isso possibilita uma taxa de quadros mais alta, mas como existe um atraso para as poses calculadas serem retornadas, a renderização do conteúdo virtual também sofre um atraso, não ficando perfeitamente casada com a imagem da câmera quando há movimento.

A outra forma é o modo acoplado, que consiste em passar a imagem capturada pela câmera para uma função que realiza o rastreamento logo em sequência. O programa então fica aguardando o retorno da pose calculada, e, uma vez recebida, a utiliza para a renderização. A vantagem dessa opção é que a exibição do conteúdo virtual fica casada perfeitamente com a imagem do objeto real capturada pela câmera. A desvantagem é que, como o fluxo de câmera tem que esperar, ocorre uma diminuição na taxa de quadros exibidos. Neste trabalho, optou-se pela segunda opção.

3.3. ABORDAGEM DISTRIBUÍDA

Foi implementada também uma abordagem distribuída. Essa abordagem é cliente-servidor. Para desenvolvê-la foi utilizada a biblioteca OpenCV4Android para a parte que roda no dispositivo móvel e uma versão da biblioteca OpenCV para Desktop, chamada de OpenCV Java [41], para a parte que roda no PC. As duas versões são compatíveis entre si, isto é, possuem as mesmas classes.

A ideia aqui foi transferir o processamento para o servidor. A extração de características é realizada no dispositivo móvel. Os pontos-chave e os descritores obtidos nesta etapa são enviados para o servidor. Para otimizar o desempenho, são enviados apenas dois atributos de cada ponto-chave para o servidor: suas coordenadas x e y. Isso é suficiente para construir os pontos-chave do lado do servidor. Lá são executadas as demais etapas do rastreamento, e os seis `doubles` que compõem a matriz de pose são enviados de volta para o cliente, um por um, por questão de desempenho. Sempre é

mais rápido enviar valores primitivos que enviar uma estrutura inteira do OpenCV, porque a quantidade de bytes a ser enviada é menor. Uma visão geral desse processo está ilustrada na Figura 10.

Figura 10. Fluxo dos dados na versão cliente-servidor implementada.

No servidor, os pontos-chave são recebidos um a um e colocados em uma estrutura única. Também são recebidos os descritores. Logo em seguida, os pontos-chave e os descritores são passados para a próxima etapa, o casamento de características. O servidor tem uma cópia do modelo a ser rastreado, e os pontos-chave e descritores do modelo já foram extraídos previamente. Após obtidos os casamentos, é realizada a filtragem dos *outliers* e o cálculo da pose, da mesma forma que na abordagem autônoma.

Uma vez que a pose foi calculada, obtem-se duas matrizes. Uma representa a parte de rotação da matriz de pose, e a outra representa sua translação. Cada uma desses é uma matriz com uma única coluna e três linhas, composta por valores do tipo `double`.

Os seis valores das matrizes chegam no lado cliente, e a matriz de pose é composta novamente, sendo então utilizada para desenhar algum conteúdo virtual sobre o objeto real rastreado.

4. ESTUDO DE CASO E RESULTADOS

Para realizar o estudo de caso, foi utilizado um tablet Samsung Galaxy Note 10.1, com 3GB de memória RAM, 32GB de memória ROM e um processador Quad-core 2.3 GHz Krait 400, modelo LTE.

A aplicação utilizada em todos os testes consistiu de carregar um modelo previamente ao início da execução, e rastrear esse modelo no mundo real, projetando linhas virtuais sobre suas bordas, como ilustrado na Figura 11. As aplicações foram avaliadas quanto ao desempenho de cada etapa do rastreamento, bem como em relação ao consumo de memória e armazenamento. Ao final do desenvolvimento, foi possível deixar a precisão do registro de todas as versões igual a da versão sem otimizações. Ou seja, foi possível otimizar as etapas do rastreamento mantendo a precisão.

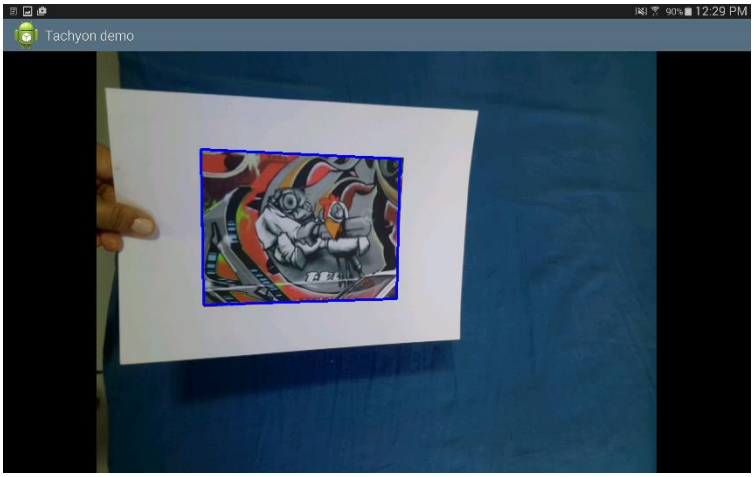

Figura 11. Captura de tela do rastreamento paralelizado por 4 *threads*. As bordas azuis são o conteúdo virtual inserido na cena.

4.1. AVALIAÇÃO DA ABORDAGEM AUTÔNOMA

A versão autônoma foi avaliada quando às quatro etapas do rastreamento: extração de características, casamento, filtragem dos *outliers* e cálculo de pose. Foi avaliado também o tempo necessário para processar um quadro por completo. As medidas foram feitas capturando-se os dados de aproximadamente uma centena de quadros ao longo de alguns minutos de execução, e tirando-se a média. Todos os dados foram medidos em milissegundos (Figura 1).

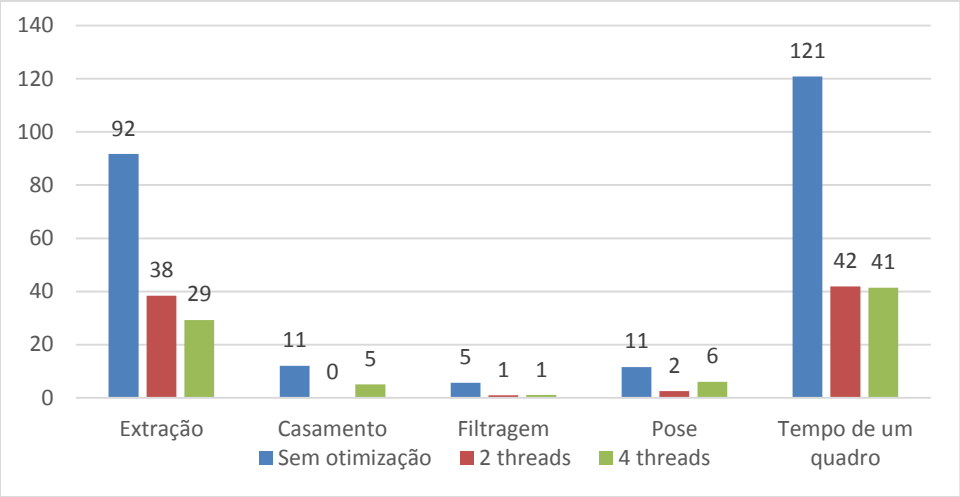

Figura 12. Resultado comparativo das medições de tempo (em ms) de cada uma das etapas do rastreamento entre a versão sem otimização e as versões desenvolvidas.

No que diz respeito a extração de características, houve um bom ganho de desempenho, como esperado. A versão sem otimizações realizou esta etapa em 91,67 ms, contra 38,37 ms da versão com duas *threads* e 29,27 ms para a com quatro.

Nas demais etapas do *pipeline* de rastreamento, também houve um ganho claro de desempenho. Vale notar, no entanto, que o gargalo de desempenho da aplicação estava, de fato, na extração de características, pois as demais etapas do rastreamento são mais rápidas.

O casamento foi realizado em 11,99 ms para a versão sem otimizações, 3,75 ms para a versão com duas *threads* e 5,02 ms para a versão com quatro *threads*. Nesse caso, o custo adicional de se criar mais duas *threads* para realizar o processamento quad-core não foi compensado pela agilidade adicional de processamento obtida na etapa de casamento. O mesmo ocorreu com as etapas de filtragem de *outliers* e cálculo da pose, ficando o desempenho da versão com duas *threads* ligeiramente melhor que o da versão com quatro. Novamente, isso ocorreu por serem etapas rápidas, em que criar *threads* adicionais não compensou. A filtragem da versão sem otimização foi executada no tempo de 5,62 ms, contra 1,00 ms da versão com duas *threads* e 1,03 ms da versão com quatro *threads*, enquanto que o cálculo da pose foi realizado em 11,60 ms pela versão sem otimização contra 2,55 ms da versão com duas *threads* e 6,07 ms da versão com quatro. Como podemos notar, também ocorreram diminuições significativas no tempo de execução dessas etapas.

O resultado final da avaliação foi que houve o consumo de aproximadamente 121ms (taxa de 8,26 quadros por segundo - *frames per second* - fps) para a versão sem otimização processar um quadro, aproximadamente 42ms (23,80 fps) para a versão com duas *threads* e aproximadamente 41ms (24,39 fps) para a versão com quatro *threads*.

Quanto ao consumo de memória e armazenamento, as versões desenvolvidas se tornaram mais custosas que a versão sem otimização. Isso já era esperado, pois existe um custo adicional inerente ao uso do módulo NDK do Android, bem como a inclusão da biblioteca TBB. Além disso, nas versões desenvolvidas, temos duas ou quatro *threads* realizando simultaneamente operações custosas, que consomem memória, como a extração de características. Por isso, houve um maior consumo de memória, mas o desempenho da aplicação não chegou a ser afetado em nada por conta disso. Os resultados podem ser vistos na Tabela 1.

Tabela 1 Resultado comparativo das medições de memória e armazenamento, ambas em MB.

	Sem otimização	2 threads	4 threads
Memória	25	102	114
Armazenamento	2,04	2,65	2,66

4.2. AVALIAÇÃO DA VERSÃO DISTRIBUÍDA

A versão distribuída também foi avaliada quanto ao desempenho, consumo de memória e armazenamento. Os das medições de desempenho podem ser vistos na Tabela 2. Não houve modificações na etapa de extração de características, sendo esta a única etapa do rastreamento que continua sendo executada no dispositivo móvel. Isso foi decidido com base em testes realizados, nos quais foi avaliado a possibilidade de enviar as imagens capturadas pela câmera para o servidor. Nesses testes, percebeu-se que o tempo de transmissão de imagens inteiras via WiFi é extremamente alto, não sendo adequado a uma aplicação em tempo-real como a proposta. Computar os pontos-chave e descritores no próprio aparelho se mostrou uma alternativa muito mais viável. Como consequência disso, a extração de características demorou aproximadamente 92ms, tanto na versão sem otimização quando na versão cliente/servidor.

Tabela 2. Resultado das medições de tempo (em ms) da extração de características e de um quadro completo na versão cliente/servidor.

Extração	91,67613636
Comunicação com servidor	26
Tempo de um quadro	117,2997932

As demais etapas do rastreamento foram processadas em um servidor. No servidor, o tempo total do casamento de características, filtragem de *outliers* e cálculo da pose é irrisório (próximo de 2 ms). Entretanto, como já foi comentado anteriormente, o gargalo de processamento do rastreamento é a extração de características. As demais etapas são relativamente rápidas.

Conseguiu-se nessa aplicação cliente-servidor uma melhora no desempenho das três etapas finais do rastreamento de características naturais, transferindo essas etapas para um servidor, porém, o tempo necessário para enviar os dados via Wi-Fi partindo do dispositivo móvel para o servidor no PC acabou por tolher o desempenho da aplicação, ficando este muito similar a versão sem otimizações. O tempo de um quadro na versão cliente/servidor ficou em 117ms contra 121ms da versão sem otimizações.

Houve alguma melhora em relação ao consumo de memória e armazenamento na versão cliente/servidor. Como a aplicação ficou menor, pois parte do código agora é exclusivo do lado servidor, a parte da aplicação que reside no dispositivo móvel ficou mais leve. Por motivo similar (parte do processamento agora é executado no servidor), o consumo de memória também ficou um pouco menor. As medidas de memória podem ser visualizadas na Tabela 3. A versão cliente/servidor consumiu 18MB de memória RAM, enquanto que a versão sem otimização consumiu 25MB. O gasto com armazenamento da versão cliente/servidor foi de 1,84MB contra 2,04MB da versão sem otimizações.

Tabela 3. Resultados da versão cliente/servidor quanto ao consumo de memória durante a execução e armazenamento, ambos em MB.

	Cliente/Servidor
Memória	18
Armazenamento	1,84

5. CONCLUSÃO E TRABALHOS FUTUROS

Foram implementadas duas soluções que possibilitaram o estudo do rastreamento baseado em características naturais. A versão *multithread* se mostrou muito mais eficiente que a versão sem otimizações, confirmando a viabilidade de se acelerar o processo de rastreamento utilizando paralelismo. Foi possível reduzir consideravelmente o tempo de cada etapa, principalmente a mais custosa, que foi a extração de características. Isso tudo foi conseguido mantendo-se a precisão do rastreamento. A solução distribuída foi capaz de acelerar as etapas de casamento, filtragem e cálculo de pose do rastreamento transferindo essas funções para um servidor. Essa solução também se mostrou precisa. No entanto, percebeu-se que o tempo de transferir os pontos-chave e descritores para o servidor acabou se tornando um gargalo de desempenho.

Em cenários que o desempenho é prioridade, avaliou-se que é mais vantajoso utilizar a versão autônoma. Já no caso de aplicações que rastreiam uma cena mais complexa, necessitando de uma base de dados grande e que utiliza muita memória e armazenamento, a versão distribuída é mais indicada. Dependendo da capacidade de processamento do dispositivo móvel, também pode ser feita uma escolha entre as duas versões. Dispositivos mais modernos, com vários núcleos, poderão tirar vantagens da implementação *multithread*. Caso o processamento do dispositivo seja mais limitado, é possível utilizar a versão cliente-servidor para transferir parte do processamento para um servidor, ganhando desempenho na aplicação.

Como trabalho futuro, a intenção é de refinar o tratamento dos dados para melhorar o desempenho da versão autônoma com quatro *threads*. Pretende-se implementar para a solução distribuída uma forma mais otimizada de transferência dos dados via sockets. Também será feita uma modificação no código para que a extração de características ocorra numa área limitada da imagem da câmera. Considerando que o usuário segurando um dispositivo móvel numa situação de RA se move lentamente, podemos estimar, baseado nos quadros anteriores, onde o objeto detectado deve estar no quadro atual. Assim, limitamos a extração de características a uma região não muito distante de onde estimamos que o objeto estará. Com isso, acredita-se que seja possível conseguir taxas de rastreamento otimizadas para as duas versões. Na versão distribuída, esse procedimento será particularmente vantajoso, por diminuir a quantidade de dados que precisam ser transmitidos via Wi-Fi. Finalmente, pretende-se avaliar o desempenho das duas soluções em cenários mais complexos, como rastreamento para setor automotivo, a fim de comparar melhor as vantagens e desvantagens de se usar cada uma das soluções.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] R. T. Azuma, Y. Baillot, R. Behringer, S. Feiner, S. Julier e B. MacIntyre, "Recent Advances in Augmented Reality," *IEEE Computers & Graphics,* pp. 34-47, Novembro 2001.

[2] J. P. Lima, F. Simões, L. Figueiredo e J. Kelner, "Model Based Markerless 3D Tracking applied to Augmented Reality," *Journal on 3D Interactive Systems,* vol. 1, pp. 1-15, 2010.

[3] J. P. S. d. M. Lima, "Object Detection and Pose Estimation from Rectification of Natural Features Using Consumer RGB-D Sensors," Recife, 2014.

[4] "OpenCV," [Online]. Available: http://opencv.org/ . [Acesso em 13 Outubro 2014].

[5] "Threading Building Blocks," [Online]. Available: https://www.threadingbuildingblocks.org/. [Acesso em 13 Outubro 2014].

[6] J. D. J. Shotton, "Contour and Texture for Visual Recognition of Object Categories," Cambridge, 2007.

[7] R. C. Gonzalez e R. E. Woods, Digital Image Processing, New Jersey: Prentice-Hall, 2002.

[8] V. Lepetit e P. Fua, "Monocular Model-Based 3D Tracking of Rigid Objects: A Survey," *Foundations and Trends in Computer Graphics and Vision,* p. 1–89, 2005.

[9] G. Klein e D. Murray, "Parallel tracking and mapping for small AR workspaces," em *International Symposium on Mixed and Augmented Reality*, 2007.

[10] A. J. Davison, I. D. Reid, N. D. Molton e O. Stasse, "MonoSLAM: Real-Time Single Camera SLAM," *Ieee Transactions on Pattern Analysis and Machine Intelligence,* pp. 1-16, Junho 2007.

[11] D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond e D. Schmalstieg, "Real-Time Detection and Tracking for Augmented Reality on Mobile Phones," *IEEE Transactions on Visualization and Computer Graphics,* 2009.

[12] M. Zyda, R. Gossweiler, J. Morrison, S. Singhal e M. P. Macedonia, "Networked Virtual Environments," em *Proceedings of the Virtual Reality Annual International*, 1995.

[13] "metaio," 23 2 2015. [Online]. Available: http://www.metaio.com/products/sdk/. [Acesso em 2015].

[14] "Cylinder Targets," Qualcomm, [Online]. Available: https://developer.vuforia.com/resources/dev-guide/cylinder-targets. [Acesso em 23 2 2015].

[15] "Layar," [Online]. Available: https://www.layar.com/. [Acesso em 23 2 2015].

[16] "D'Fusion Mobile," [Online]. Available: http://www.t-immersion.com/products/dfusion-suite/dfusion-mobile. [Acesso em 23 2 2015].

[17] S. DiVerdi, J. Wither e T. Höllerer, "Envisor: Online Environment Map Construction for Mixed Reality," em *IEEE VR 2008*, 2008.

[18] A. Swaminathan, M. Chari, S. Diaz e S. Grzechnik, "Information-Theoretic Database Building and Querying for Mobile Augmented Reality Applications," em *IEEE International Symposium on Mixed and Augmented Reality 2011*, 2011.

[19] D. Kurz e S. Benhimane, "Gravity-Aware Handheld Augmented Reality," em *IEEE International Symposium on Mixed Augmented Reality*, 2011.

[20] X. Yang e K.-T. Cheng, "LDB: An ultra-fast feature for scalable Augmented Reality on mobile devices," em *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, 2012.

[21] N. Hagbi, O. Bergig, J. El-Sana e M. Billinghurst, "Shape Recognition and Pose Estimation for Mobile Augmented Reality," em *8th IEEE International Symposium on Mixed and Augmented Reality*, 2009.

[22] B.-K. Seo, J. Park e J.-I. Park, "3-D visual tracking for mobile augmented reality applications," em *IEEE International Conference on Multimedia and Expo*, 2011.

[23] G. Klein e D. Murray, "Parallel Tracking and Mapping on a camera phone," *8th IEEE International Symposium on Mixed and Augmented Reality,* 2009.

[24] W. Pasman e C. Woodward, "Implementation of an Augmented Reality System on a PDA," *2nd IEEE/ACM International Symposium on Mixed and Augmented Reality,* 2003.

[25] D. M. Chen, S. S. Tsai, R. Vedantham, R. Grzeszczuk e B. Girod, "Streaming mobile augmented reality on mobile phones," em *8th IEEE International Symposium on Mixed and Augmented Reality*, 2009.

[26] J. Ha, K. Cho, F. A. Rojas e H. S. Yang, "Real-time scalable recognition and tracking based on the server-client Model for mobile Augmented Reality," em *IEEE International Symposium on VR Innovation (ISVRI)*, 2011.

[27] C. Arth, M. Klopschitz, G. Reitmayr e D. Schmalstieg, "Real-Time Self-Localization from Panoramic Images on Mobile Devices," em *10th IEEE*

International Symposium on Mixed and Augmented Reality, 2011.

[28] D. Nister e H. Stewenius, “Scalable Recognition with a Vocabulary Tree,” *IEEE Computer Society Conference on Computer Vision and Pattern Recognition,* 2006.

[29] O. Chum e J. Matas, “Matching with PROSAC - progressive sample consensus,” em *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2005.

[30] P. Hasper, N. Petersen e D. Stricker, “Remote Execution vs. Simplification for Mobile Real-time Computer Vision,” em *9th International Conference on Computer Vision Theory and Applications*, 2014.

[31] “Metaio CVS,” [Online]. Available: http://www.metaio.com/cvs/. [Acesso em 23 2 2015].

[32] F. Moreno-Noguer, V. Lepetit e P. Fua, “Accurate non-iterative O(n) solution to the PnP problem,” em *IEEE International Conference on Computer Vision*, 2007.

[33] M. Fischler e R. Bolles, “Random Sample Consensus: A paradigm for model fitting with applications to image analysis and automated cartography,” *Communications of the ACM 24(6),* p. 381–395, 1981.

[34] D. Lowe, “Distinctive image features from scale-invariant keypoints,” *International Journal of Computer Vision 60(2),* pp. 91-110, 2004.

[35] N. Snavely, S. M. Seitz e R. Szeliski, “Skeletal sets for efficient structure from motion,” *Computer Vision and Pattern Recognition,* 2008.

[36] E. Rublee, V. Rabaud, K. Konolige e G. Bradski, “ORB: An efficient alternative to SIFT or SURF,” em *IEEE International Conference on Computer Vision (ICCV)*, 2011.

[37] E. Rosten e T. Drummond., “Machine learning for highspeed corner detection,” em *European Conference on ComputerVision*, 2006.

[38] M. Calonder, V. Lepetit, C. Strecha e P. Fua, “Brief: Binary robust independent elementary features,” em *11th European conference on Computer vision*, 2010.

[39] “OpenCV4Android SDK,” [Online]. Available: http://docs.opencv.org/doc/tutorials/introduction/android_binary_package/O4A_SDK.html. [Acesso em 23 2 2015].

[40] “Android NDK,” [Online]. Available: https://developer.android.com/tools/sdk/ndk/index.html. [Acesso em 23 2 2015].

[41] "JavaCV," [Online]. Available: https://github.com/bytedeco/javacv. [Acesso em 23 2 2015].

[42] D. M. Chen, S. S. Tsai, R. Vedantham, R. Grzeszczuk e B. Girod, "Streaming Mobile Augmented Reality on Mobile Phones," em *IEEE International Symposium on Mixed and Augmented Reality*, Orlando, 2009.

[43] W. Pasman e C. Woodward, "Implementation of an Augmented Reality System on a PDA," em *Proceedings of the Second IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR 2003)*.

[44] "Camera Calibration and 3D Reconstruction," [Online]. Available: http://docs.opencv.org/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html. [Acesso em 13 Outubro 2014].

[45] M. Ozuysal, P. Fua e V. Lepetit, "Fast keypoint recognition in ten lines of code," em *IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[46] D. Wagner e D. Schmalstieg, "Making Augmented Reality Practical, Part 1," *IEEE Computer Graphics and Applications,* pp. 12-15, 2009.

[47] S. Se, D. Lowe e J. Little, "Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks," *International Journal of Robotic Research,* Agosto 2002.

Datas e Assinaturas

Recife, 23 de fevereiro de 2015.

_____

Veronica Teichrieb (Orientadora)

_____

João Paulo Silva do Monte Lima (Co-orientador)

_____

Rafael Alves Roberto (Co-orientador)

_____

Guilherme Reis da Rocha Borba Marques (Proponente)