Arquitetura para o desenvolvimento de um Ambiente de Simulação de Tiro

Luiz José Schirmer Silva, Tiago Boelter Mizdal,
Luiz Felipe Netto, Alex T. Almeida Frazzon,
Guilherme G. Schardong, Cesar Tadeu Pozzer
Laboratório de Computação Aplicada – LaCA
Universidade Federal de Santa Maria
Dept. de Computação Aplicada
{luiz,tiagob,netto,afrasson,schardon,pozzer}@inf.ufsm.br

Leonardo Quatrin Campagnolo[1],Erick Baptista Passos[2]
Pontifícia Universidade Católica do Rio de Janeiro[1]
Instituto Federal do Piauí[2]

lcampagnolo@inf.puc-rio.br
erickpassos@gmail.com

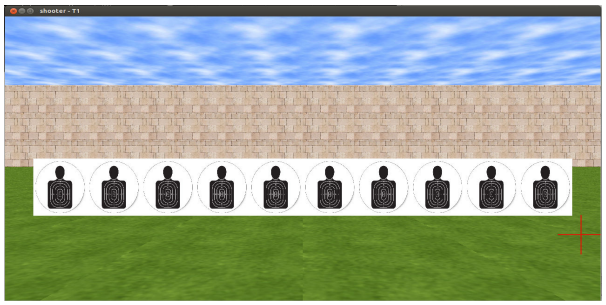

Figura 1. Protótipo de ambiente de tiro desenvolvido.

***Resumo* — Neste artigo apresenta-se uma arquitetura para o desenvolvimento de um simulador de tiro utilizando técnicas de visão computacional e sensores de captura de movimento. O objetivo deste simulador é proporcionar um ambiente virtual para prática de disparo de armas de fogo. Para a implementação do sistema, foram analisados dados gerados a partir da utilização de sensores que fazem uso de acelerômetros no mecanismo de disparo. Além disso, foram avaliadas técnicas de visão computacional que utilizam câmeras para a calibração da mira do usuário e demarcação da área de projeções dos alvos. Dentre estas a que se mostrou mais promissora foi o uso de *fiducials*.**

Palavras-chave—fiducials; simulador de tiro; captura de movimento; ambiente virtual; visão computacional.

I. INTRODUÇÃO

Recentemente, várias tecnologias de captura de movimento vêm sendo desenvolvidas e incorporadas na criação de jogos com o objetivo de oferecer uma maior interatividade e imersão aos jogadores. A maioria dos jogos criados com o auxílio destes dispositivos tem como objetivo de analisar a postura e a movimentação do jogador perante a ação desempenhada de forma similar a muitos programas de simulação.

Um dos principais campos de desenvolvimento de games na atualidade é a criação de jogos de tiro em primeira pessoa (FPS), porém são raros os títulos no mercado que fazem o uso de algum sensor de captura de movimento, a grande maioria faz uso de técnicas de controle tradicionais tornando a ação mecânica e pouco natural. Para o desenvolvimento de simuladores utilizando sensores é necessário, primeiramente, executar o processo de detecção de movimento de modo confiável, diferentemente de um *game* comum. Com relação ao posicionamento do jogador perante o sistema, algumas abordagens já utilizadas por pesquisadores em outros tipos de aplicação, como por exemplo, jogos para reabilitação motora, incluem a utilização de sensores ligados à roupa ou ao corpo do usuário e técnicas de visão computacional, como detectar uma luva colorida na mão do usuário para o processo de calibração e utilizar seu posicionamento como interface para aplicativos [5]. Embora essas técnicas solucionem o problema de detecção do movimento, geram empecilhos para a interação com simuladores de tiro, pois o foco deste tipo de game não é apenas a obtenção de dados de movimentação, mas principalmente dados da mira do jogador.

Os simuladores de tiro podem ser caracterizados como sendo jogos sérios, já que o objetivo principal do sistema não é o entretenimento do usuário, mas sim seu treinamento e análise de sua postura. Sistemas deste tipo são usados para auxiliar no treinamento de agentes de segurança pública, como, por exemplo, policiais militares. O uso de simuladores neste contexto implica em uma diminuição de custos e redução de riscos de acidentes de tiro a "zero" [6].

Em busca de uma solução mais prática e simplificada este trabalho propõe uma nova arquitetura para um ambiente de simulação de tiro (Figura 1) onde são abordadas estratégias para análise da mira do usuário utilizando técnicas de visão computacional. A área de tiro é demarcada através de *fiducials* [3] e os disparos e sua a acurácia são avaliados de

acordo com algoritmos que utilizam dados obtidos através de uma câmera acoplada ao mecanismo de disparo. Além disso, também são processadas informações relacionadas à movimentação do atirador através de acelerômetros e giroscópios também acoplados à arma.

Este trabalho está organizado da seguinte maneira: na seção 2 serão discutidos trabalhos relacionados e suas diferenças com relação a esta proposta. A seção 3 realiza uma descrição do ambiente de tiro, bem como suas possíveis limitações, e ainda é apresentado o processo de desenvolvimento, as técnicas utilizadas e a integração dos algoritmos de visão com o ambiente virtual. Os resultados são apresentados na seção 4. Na seção 5 é apresentada a conclusão e os trabalhos futuros.

II. FUNDAMENTAÇÃO TEÓRICA

Para justificar a escolha do uso de *fiducials* na demarcação da área de tiro e ainda da escolha do dispositivo para captura de movimento, foi realizado um estudo comparativo das soluções existentes para a construção de simuladores de tiro. A maioria dos sistemas disponíveis no mercado faz uso de *lasers* infravermelhos para demarcar o alvo definido pelo atirador. O *laser* é acoplado a um dispositivo, neste caso o mecanismo de disparo usado para simular uma arma de fogo, e é disparado contra uma superfície onde está localizada a área de tiro contendo os respectivos alvos. Para determinar de forma precisa a localização do laser nesta superfície, um sensor óptico é usado para registrar a sua posição [7]. Considerando ainda que os alvos são formandos por um ambiente virtual 3D projetado nesta mesma área, combinando o uso de lasers com técnicas de visão computacional, esta implementação mostra-se eficiente. Segundo Huang, a utilização de *lasers* é uma prática que possui uma grande precisão para o cálculo de posicionamento, inclusive técnicas semelhantes, como *Laser speckle,* são capazes de produzir um grau de acurácia elevadíssimo, considerando uma margem de erro de 0,3%. Porém na busca por uma abordagem simplificada, reduzindo o número de dispositivos envolvidos e diminuindo o custo de implementação, nesta presente pesquisa foram analisados modelos implementados em sistemas de realidade aumentada (RA). Esta solução permite eliminar a utilização de *lasers*, definindo a área de tiro através de marcadores em suas bordas e utilizando uma API para RA, como ARToolkit, combinada à uma câmera comum, define-se a mira do atirador pelo centro de imagem capturada e a posição dos alvos em relação a esta câmera. Os dados obtidos a partir da imagem capturada são posteriormente analisados de acordo com a área definida por múltiplos *fiducials*, a fim de identificar os alvos dispostos na tela, e calcular o número de disparos que atingiram esses alvos. Esta técnica possui como característica principal, ser uma abordagem alternativa aos métodos tradicionais e ainda possuir um grau de precisão aceitável. De forma adicional ainda foram analisados dispositivos que utilizam sensores como acelerômetros e giroscópios. Estes dispositivos são usados a fim de analisar a movimentação do braço do atirador no momento de cada disparo para identificar possíveis erros e "vícios", como por exemplo, o fato de o usuário movimentar levemente a mão antes de cada disparo, o que pode interferir no resultado final. Foram estudados sensores utilizados em *videogames* comuns, como o Playstation Move e o Nintendo Wii Remote, e dentre eles o que apresentou melhores resultados foi o PS Move [8]. O PS Move possui uma resposta mais rápida que o Wii Remote além de ser possível fazer uma análise mais detalhada da movimentação nos eixos x, y e z do que o dispositivo da Nintendo.

III. ANÁLISE DE TECNOLOGIAS

No processo desenvolvido para a demarcação da área de tiro foram avaliadas diferentes técnicas de visão computacional. A área de tiro, a qual contém os alvos a serem considerados, é exibida em um monitor ou através da projeção sobre uma superfície. Por sua vez, a mira do atirador é detectada através da área central da imagem captada pela câmera acoplada ao dispositivo de disparo. Caso a imagem central da câmera coincida com parte da área de tiro projetada é possível verificar se algum dos alvos foi atingido. Uma comparação entre as APIs OpenCV e ARToolkit foi realizada a fim de determinar a técnica para demarcação da área de tiro com maior precisão. Com relação ao OpenCV, foram testados dois algoritmos de detecção de bordas para identificar a tela ou área onde serão projetados os alvos, sendo eles o algoritmo de Harris [2] e o algoritmo de Shi-Tomasi [4]. A abordagem de Harris para a detecção de bordas faz uso de segmentação de imagens, criando uma nova janela com apenas uma parte da imagem original e deslocando esta imagem em todas as direções sobre a imagem original. Posteriormente, dados dois autovalores para a função de Harris, e calculando a diferença produzida entre a janela e a região que está sendo sobreposta na imagem original, é possível conseguir uma pontuação, onde se define através dela uma constante utilizada para a demarcação de um dos cantos da área definida. Se a pontuação for superior à constante, aquela janela pode ser considerada um canto da área total. Esta abordagem se mostrou eficiente para o reconhecimento dos cantos da área de projeção, porém, ao utilizar esta técnica em um simulador de tiro, o algoritmo não foi eficaz, devido a sua incapacidade de realizar a detecção dos cantos a cada *frame* captado pela câmera, o que é necessário, pois é preciso que o usuário possa se movimentar para mirar nos alvos. Ao utilizar-se este algoritmo observou-se uma grande flutuação nos pontos detectados ao movimentar a câmera, onde se acaba perdendo a referência de cada canto da imagem reconhecida durante a execução do programa.

O algoritmo de Shi-Tomasi realiza uma abordagem similar a de Harris para a detecção de cantos, porém o cálculo da pontuação é simplificado, pois leva em consideração apenas o valor mínimo entre os dois autovalores. Este algoritmo foi

mais eficiente durante a realização de testes, sendo eficaz tanto na detecção dos cantos quanto na definição da área de tiro, realizando a detecção mais rapidamente que a abordagem de Harris. Porém, para que seja possível processar a imagem capturada, é necessário limitar a quantidade de cantos que o algoritmo reconhece por imagem. Contudo, ao limitar esta quantidade, não se obtém o reconhecimento dos mesmos cantos da imagem anterior, e não é possível detectar os cantos da tela ao mover a câmera rapidamente.

Além dos métodos de detecção de bordas anteriormente apresentados, também foi testado o algoritmo de Suzuki [11] em conjunto com o algoritmo de Canny [12], ambos já implementados na API OpenCV, para detectar a área de projeção. O algoritmo de Canny localiza as bordas presentes na imagem. A imagem gerada pelo detector de bordas é então submetida ao algoritmo de Suzuki que localiza todas as bordas fechadas. Após esses passos, as áreas dos objetos encontrados são calculadas. Para fins de teste, foi assumido que o maior valor de área pertence à tela. Porém, como o ambiente de tiro pode conter inúmeros objetos, o cálculo das áreas resultou em um maior tempo de resposta, tornando essa abordagem inviável. A utilização de um algoritmo de *blur* para tentar eliminar o ruído da imagem tornou mais difícil a detecção da tela. A figura 2 ilustra o processo descrito acima.

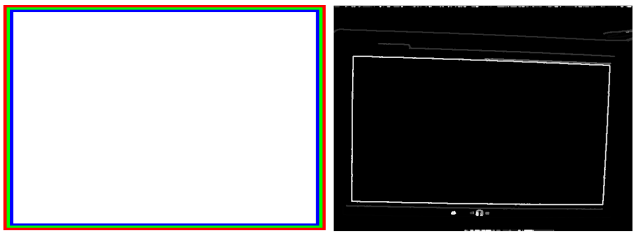

Figura 2. A marcação da área de tiro pode ser vista na imagem da esquerda. A imagem da direita apresenta o resultado dos algoritmos de Canny e Suzuki para a detecção da área de tiro

Com base nos problemas encontrados até então, foi utilizada uma estratégia alternativa, fazendo o uso de marcadores para identificar os cantos da tela. Para isso foi testada a API ARToolkit a fim de utilizar dois *fiducials* dispostos nos cantos da diagonal da tela (figura 3) para demarcar o ambiente onde estão localizados os alvos. A área de projeção é então definida pelos dois fiducials utilizados, um na posição considerada de origem da tela (0,0) e outro na posição final (larguraMax, alturaMax). Esta API recebe informações da câmera e busca identificar marcadores que possuem a forma quadrada, e que normalmente são construídos utilizando imagens monocromáticas ou em escala de cinza [1]. Com base na identificação destes *fiducials* é possível determinar a posição da câmera, acoplada ao dispositivo de disparo, em relação à área de tiro.

Estimando a distância entre os dois marcadores através do ARToolkit, definimos um plano delimitado pelas arestas superior, inferior, a esquerda e a direita da tela. O ponto central da câmera é tratado como um objeto na altura do plano e então é calculada sua posição relativa para determinar o ponto de disparo no cenário apresentado pelo simulador.

Algumas limitações na detecção dos *fiducials* foram observadas durante a utilização desta API, como por exemplo, a sensibilidade em relação à iluminação do ambiente. Há a necessidade de uma boa incidência de luz sobre o *fiducial*, mas deve haver o cuidado para evitar o excesso de luz, pois o brilho e o reflexo dificultam a identificação do marcador. A detecção também é sensível ao tamanho dos *fiducials*, bem como a sua distância e orientação em relação à câmera.

Além da implementação do modelo para identificação da mira do usuário, foi adicionado ao mecanismo de disparo o sensor de captura de movimento PS Move a fim de analisar a movimentação do braço do atirador. Através da PS Move API [10], os dados advindos do sensor foram integrados ao sistema e puderam ser analisados através de um gráfico gerado pela API que apresenta a aceleração do PS Move nas dimensões X, Y e Z conforme a figura 4.

A partir da análise destas informações, obtidas através de acelerômetros contidos no sensor, é possível verificar possível flutuações na mira baseadas na movimentação do atirador. Estas informações podem ser úteis para a correção da postura do usuário durante o treinamento de disparos.

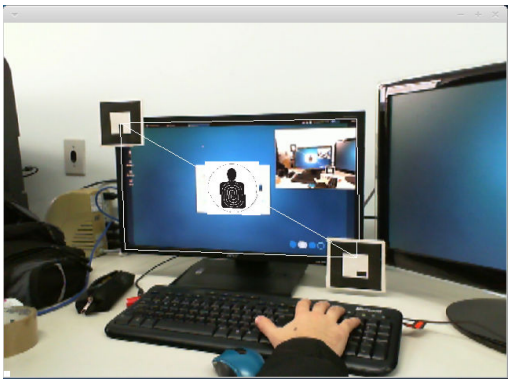

Figura 3. Protótipo para identificação da área de tiro

Por fim, para a realização de testes, foi desenvolvido um ambiente 3D em OpenGL semelhante à um estande de tiro, contendo os alvos a serem considerados.

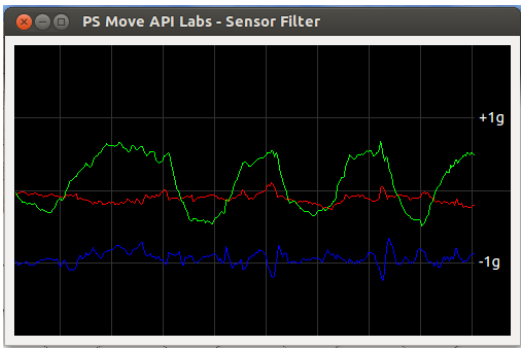

Figura 4. Gráfico contendo informações sobre a aceleração do atirador, sendo o eixo x representado pela cor vermelha, o eixo y pela verde e o z pela azul.

IV. RESULTADOS

Após uma série de testes, foi possível identificar uma projeção aceitável do plano fazendo uso de *fiducials*, os quais devem possuir um tamanho de 8 x 8 centímetros, a uma distância de 1.0 a 2.0 metros da câmera com um ângulo de visão de 90º graus na horizontal e 45º graus na vertical. A escolha de um marcador com complexidade simples também foi um fator determinante para uma melhor identificação. Padrões complexos podem gerar resultados inesperados devido às distorções geradas na obtenção das imagens pela câmera. Foram detectadas ainda pequenas oscilações devido a distância da câmera sobre os marcadores, porém este problema foi minimizado, pois o ARToolkit dispõe de uma função de detecção de marcadores baseado no histórico de detecção, que minimiza o efeito de *jitter* durante a projeção do plano. Esta técnica mostrou-se eficaz e com um grau de precisão aceitável, além de eliminar os problemas de detecção da área de projeção a cada *frame* descritos nas técnicas anteriores utilizando somente o OpenCV.

V. CONCLUSÕES E TRABALHOS FUTUROS

A arquitetura apresentada busca ser uma alternativa aos métodos de desenvolvimento tradicionais para simuladores de tiro. A pesquisa focou-se na identificação dos requisitos necessários ao desenvolvimento de software neste meio e de métodos computacionais para a implementação do sistema de avaliação da precisão dos disparos. A solução encontrada para validar os disparos foi baseada em técnicas de visão computacional e realidade aumentada e tem o intuito de ser genérica, podendo ser aplicada em outras medições semelhantes, sendo elas em simuladores de tiro ou em outras que necessitem de abordagens alternativas ao uso de *lasers*. Acredita-se que essa solução seja eficaz para facilitar a criação de aplicativos neste contexto e que requeiram avaliação da movimentação dos usuários.

Uma das vantagens do sistema com câmeras é que há a necessidade de apenas uma calibração no primeiro uso do sistema. Uma nova calibração será necessária apenas se houver a troca da câmera utilizada. Esse é um ponto onde o sistema a laser é menos robusto, pois ele necessita de calibração a cada movimentação da câmera.

A API ARToolkit mostrou-se útil para a identificação da área de tiro através dos marcadores dispostos nos cantos da área de projeção. Utilizando apenas uma câmera RGB comum foi possível identificar a mira do usuário de acordo com a posição central da imagem captada pela câmera, embora em testes preliminares tenham sido encontradas flutuações maiores quando comparado a um sistema baseado em *lasers*. Porém, a taxa de precisão seja limitada de acordo com o tamanho dos *fiducials*. Isso que pode comprometer algumas aplicações diferentes, mas não chegou a ser um empecilho no sistema.

Futuramente pretende-se realizar mais testes para determinar com precisão a taxa de erro do sistema desenvolvido com *fiducials* em relação à abordagem clássica com *lasers*, bem como melhorar a precisão do reconhecimento dos marcadores. Pretende-se também desenvolver um ambiente contendo alvos dinâmicos e a criação de inimigos providos de inteligência artificial (IA). Além disso, também serão incorporadas características de jogos de realidade aumentada, onde a cena poderá ser observada de diferentes ângulos. Para isso serão usados sensores de profundidade para mapear o ambiente como, por exemplo, o Microsoft Kinect [9].

REFERÊNCIAS

[1] A. Xu and G. Dudek, "Fourier Tag: A Smoothly Degradable Fiducial Marker System with Configurable Payload Capacity". Canadian Conference on Computer and Robot Vision (CRV), p.40-47, 25-27, 2011.

[2] C. Harris and M. Stephens, "A combined corner and edge detector". Alvey vision conference, Vol. 15, 1988.

[3] I. Poupyrev, H. Kato and M. Billinghurst, “ARToolKit User Manual Version 2.33”. Human Interface Technology Lab, University, 2000.

[4] J. Shi and C. Tomasi, "Good features to track". Computer Vision and Pattern Recognition, 1994.

[5] L. Geurts, V. Vanden, J. Husson and F. Wyndei, “Digital Games for Physical Therapy: Fulfilling the Need for Calibration”, p. 117-124, 2010.

[6] L. M. A. Furuie, “Melhoria na Segurança Pública com Treinamento Continuado do Policial Militar em Estande de Tiro Modelo: Elaboração de Modelo de Avaliação e Melhoria da Qualidade”, Curitiba, 2013.

[7] P. S. Huang, L. Chih-Ming and C. Ghung-Cheng, “Real-time Analysis of Laser Speckle Patterns for Precision Positioning”. First International Conference on Pervasive Computing, Signal Processing and Applications, 2010.

[8] R. Miller, “PlayStation Move Review”, December 2010. <http://www.engadget.com/2010/09/01/playstation-move-review>

[9] S. Kean, J. Hall and P. Perry, “Meet the Kinect: An Introduction to Programming Natural User Interfaces”. Apress, 2011.

[10] T. Perl, “Cross-Platform Tracking of a 6DoF Motion Controller”. Using Computer Vision and Sensor Fusion, Austria, 2012.

[11] S. Suzuki, and Abe, K., Topological Structural Analysis of Digitized Binary Images by Border Following. CVGIP 30 1, pp 32-46 (1985)

[12] J. Canny, A Computational Approach To Edge Detection, IEEE Trans. Pattern Analysis and Machine Intelligence, 8(6):679–698, 1986.