

ATENÇÃO: aqui constam somente as páginas que tinham alguma anotação na revisão.

USO DA REALIDADE AUMENTADA COM MARCADORES DINÂMICOS

Everton da Silva, Dalton Solano dos Reis – Orientador

Curso de Bacharel em Ciência da Computação

Departamento de Sistemas e Computação

Universidade Regional de Blumenau (FURB) – Blumenau, SC – Brasil

evertonslv.silva@gmail.com, dalton@furb.br

Resumo: Esse artigo descreve o processo de desenvolvimento de um aplicativo que utiliza realidade aumentada e tem como objetivo criar, detectar e apresentar objetos 3D utilizando marcadores pré-definidos ou dinâmicos, permitindo que qualquer pessoa possa criar uma cena utilizando realidade aumentada. O aplicativo foi desenvolvido através da plataforma Unity utilizando a linguagem de programação C# para escrever os scripts através da ferramenta Visual Studio Community, em conjunto com as bibliotecas OpenCV para Unity. O aplicativo permite que o usuário importe qualquer arquivo 3D (.fbx, .blender, .prefab, .obj, etc.) e em seguida crie um marcador para o mesmo, seja com ArUco ou utilizando a própria imagem do objeto importado.

Palavras-chave: Marcadores Dinâmicos. OpenCV For Unity. Realidade Aumentada

1 INTRODUÇÃO

Desde os primórdios da informática o homem busca diferentes maneiras de inovar a Interação Homem-Máquina. No início os computadores eram operados manualmente sem a utilização de painéis ou display. Com o passar dos anos foram sendo desenvolvidos monitores com uma interface mais amigável, sistemas operacionais baseados em interface gráfica, mouses e teclados. Essa evolução segue até os dias de hoje e um simples computador pode ser carregado na palma da mão (COSTA et al., 2011, p.11).

Com esses avanços tecnológicos, surgiu a Realidade Aumentada (RA), permitindo a sobreposição de objetos e ambientes virtuais com o ambiente físico através de algum dispositivo tecnológico. A RA teve sua primeira aparição na década de 90, porém, ficou mais acessível no início dos anos 2000 com a convergência de técnicas de visão computacional, software e dispositivos com um melhor custo-benefício (KIRNER; SISCOUTTO, 2007, p.5).

Em geral, a realidade aumentada funciona através de marcadores, eles são configurados e impressos com intuito da aplicação reconhecê-los e detectá-los, obtendo informações necessárias para identificar o que e onde gerar. Esta técnica é conhecida como Marker Based Tracking (rastreamento baseado em marcações) (HESS, 2011, p.19).

Outra técnica adotada para os recursos da RA é o Markerless Tracking (rastreamento sem marcações), considerada a mais ideal, já que não necessita de marcadores impressos (BIMBER; RASKAR, 2005 apud HESS, 2011, p.19). Realidade Aumentada sem marcadores do inglês Markerless Augmented Reality (MAR) é uma técnica baseada no reconhecimento de objetos do mundo real. Os sistemas baseados em MAR podem usar imagens e objetos reais para apresentar os efeitos da realidade aumentada. MAR utiliza algoritmos de reconhecimento de imagens para detecção de objetos, ao contrário dos marcadores que tem sua estrutura fixa e conhecida (OZLU, 2018).

Segundo Teichrieb (2007 apud KUMAGAI, 2011, pg. 17) Qualquer parte da cena real pode ser utilizado como marcador, podendo ser rastreada a posição do objeto, Teichrieb ainda afirma que os sistemas MAR possuem algumas vantagens por conter rastreadores especializados e robustos, além de possibilitar extrair características da cena real, porém para realizar tais ações pode ser complexos e apresentar restrições.

Diante deste cenário, este trabalho propõe o desenvolvimento de uma aplicação utilizando os recursos da realidade aumentada baseado em marcadores dinâmicos, que não necessita de marcadores pré-definidos na cena, com intuito de melhorar a interação com o usuário, permitindo que qualquer pessoa, mesmo sem muito conhecimento tecnológico consiga criar uma cena e utilizar a realidade aumentada.

2 FUNDAMENTAÇÃO TEÓRICA

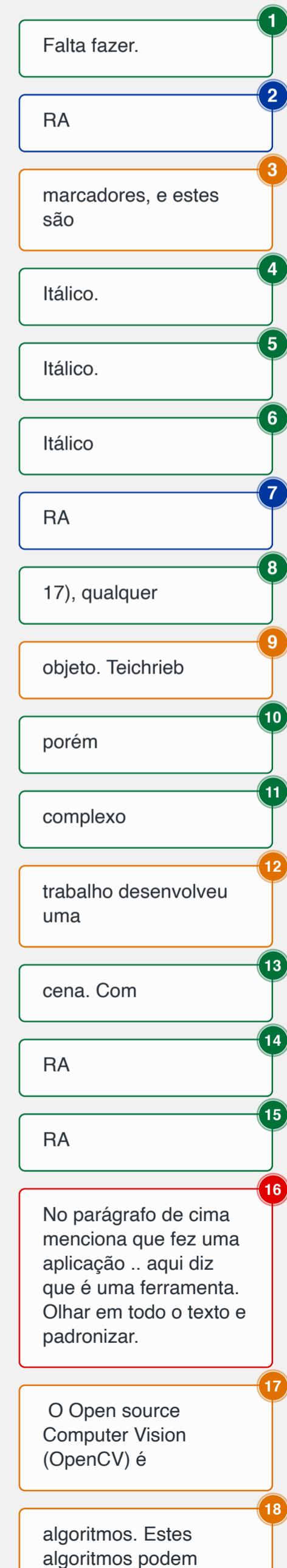
Esta seção apresenta os aspectos da fundamentação teórica utilizados na construção deste trabalho. Na primeira subseção desta seção são apresentados os conceitos utilizados como base para o desenvolvimento da ferramenta. Na segunda subseção são apresentados os trabalhos relacionados à ferramenta desenvolvida.

2.1 OPENCV

OpenCV (Open Source Computer Vision) é uma biblioteca de software de visão computacional e aprendizado de máquina de código aberto, que possui mais de 2.500 algoritmos que podem ser utilizados para detectar e reconhecer

Trabalho de Conclusão de Curso - Ano/Semestre: 2020/2

1



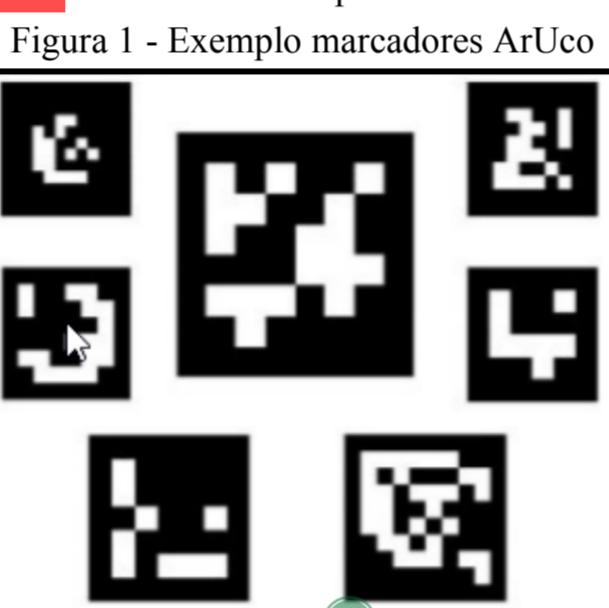
rostos, identificar objetos, classificar ações humanas em vídeos, rastrear movimentos de câmeras, rastrear objetos em movimento, extrair modelos 3D de objetos, produzir nuvens de pontos 3D de câmeras estéreo, juntar imagens para produzir uma alta resolução imagem de uma cena inteira, encontrar imagens semelhantes em um banco de dados, remover olhos vermelhos de imagens, seguir os movimentos dos olhos, reconhecer cenários e estabelecer marcadores para sobrepor com realidade aumentada, etc. (OpenCV, 2019).

A biblioteca OpenCV é escrita nativamente em C++ e possui interfaces para C++, Python, Java e MATLAB, suporta as plataformas, Windows, Linux, Android e MacOS (OpenCV, 2019). OpenCV é compatível também com Unity, porém, para conseguir tal compatibilidade é necessário declarar todas as funções C/C++ para exportação e em seguida compilar todo o código do OpenCV como um pacote de biblioteca e importá-lo no projeto como um plug-in (ENOX SOFTWARE, 2020).

A empresa Enox Software desenvolveu o plug-in OpenCV for Unity para prover eficiência computacional, sobretudo em aplicações em tempo real. O plug-in disponibiliza uma API para C# baseada na API para Java disponibilizada pelo próprio OpenCV. Para adquirir a API OpenCV for Unity é necessário fazer o download na loja Assets do Unity e comprar a licença da mesma (ENOX SOFTWARE, 2020).

2.2 ARUCO (REALIDADE AUMENTADA COM MARCADORES)

ArUco (Augmented Reality University of Cordoba) é uma biblioteca Open Source escrita em C++, responsável pela execução de rotinas de detecção de marcadores. ArUco gera os marcadores quadrados sintéticos compostos por uma borda preta larga e com uma matriz binária que determina um identificador único dentro deles. A borda preta facilita sua rápida detecção na imagem e a codificação binária permite sua identificação. O tamanho do marcador determina o tamanho da matriz interna, por exemplo, um marcador 4x4 é composto por 16 bits (ARUCO, 2020). A **Erro! Fónt de referênciá náo encontrada.** mostra um exemplo de marcadores ArUco.



Fonte: ARUCO (2020)

Suas principais características são: detectar marcadores com uma única linha de código, detectar bibliotecas como: AprilTag, ArToolKit, ArToolKit+, ARTAG, CHILITAGS, mais rápido que qualquer outra biblioteca para detecção de marcadores, multiplataforma (Windows, Linux, Mac OS, Android), poucas dependências e integração com OpenCV, OpenGL e OGRE (AVA, 2018).

Tem de ter um gancho com a seção anterior ... OpenCV.

Acho que ficaria melhor se juntasse num seção as seções 2.3 e 2.4.

Tem de ter um gancho com a seção anterior ... ArUco.

2.3 HOMOGRAFIA

Homografia é uma relação plana que transforma pontos de um plano para outro. É uma matriz 3 por 3 que transforma vetores tridimensionais que representam os pontos 2D no plano. Esses vetores são chamados de coordenadas homogêneas (PERI, 2017).

Coordenadas homogêneas são coordenadas projetivas que representam pontos dentro da visão computacional. Como existe uma conversão 3D para 2D em uma foto, a escala de profundidade é perdida. Com isso, uma quantidade infinita de pontos 3D é projetada para o mesmo ponto 2D, tornando as coordenadas homogêneas muito versátil na descrição do raio de possibilidade, uma que são semelhantes em escala (PERI, 2017).

O algoritmo da homografia é muito simples em relação a outros algoritmos por ser direto e intuitivo, porém, só é possível utilizar homografia quando a coordenada Z for igual a 0, ela só funciona em cenários planos, caso contrário, outras abordagens são necessárias. O algoritmo se torna inútil caso o alvo desejado saia de vista da câmera, sendo necessário movimentar a câmera, até que ela possa olhar o tempo todo para o alvo (PERI, 2017).

A Figura 2 mostra um exemplo de como a homografia relaciona a transformação entre dois planos.

1 objetos e classificar.

2 vídeos. Também permite rastrear

3 inteira. Bem como encontrar

4 aumentada, entre outros (

5 MATLAB. O OpenCV suporta

6 Itálico.

7 Itálico.

8 A Augmented reality University of cordoba (ArUco) é

9 A ArUco

10 Ponto final.

11 As principais características do ArUco são

12 código; detectar

13 CHILITAGS; ser mais

14 Android); e ter poucas

15 marcadores; ser multiplataforma

16 HOMOGRAFIA E TRANSFORMAÇÕES DE PERSPECTIVA

17 Já coordenadas

18 igual a 0. Pás a homografia só

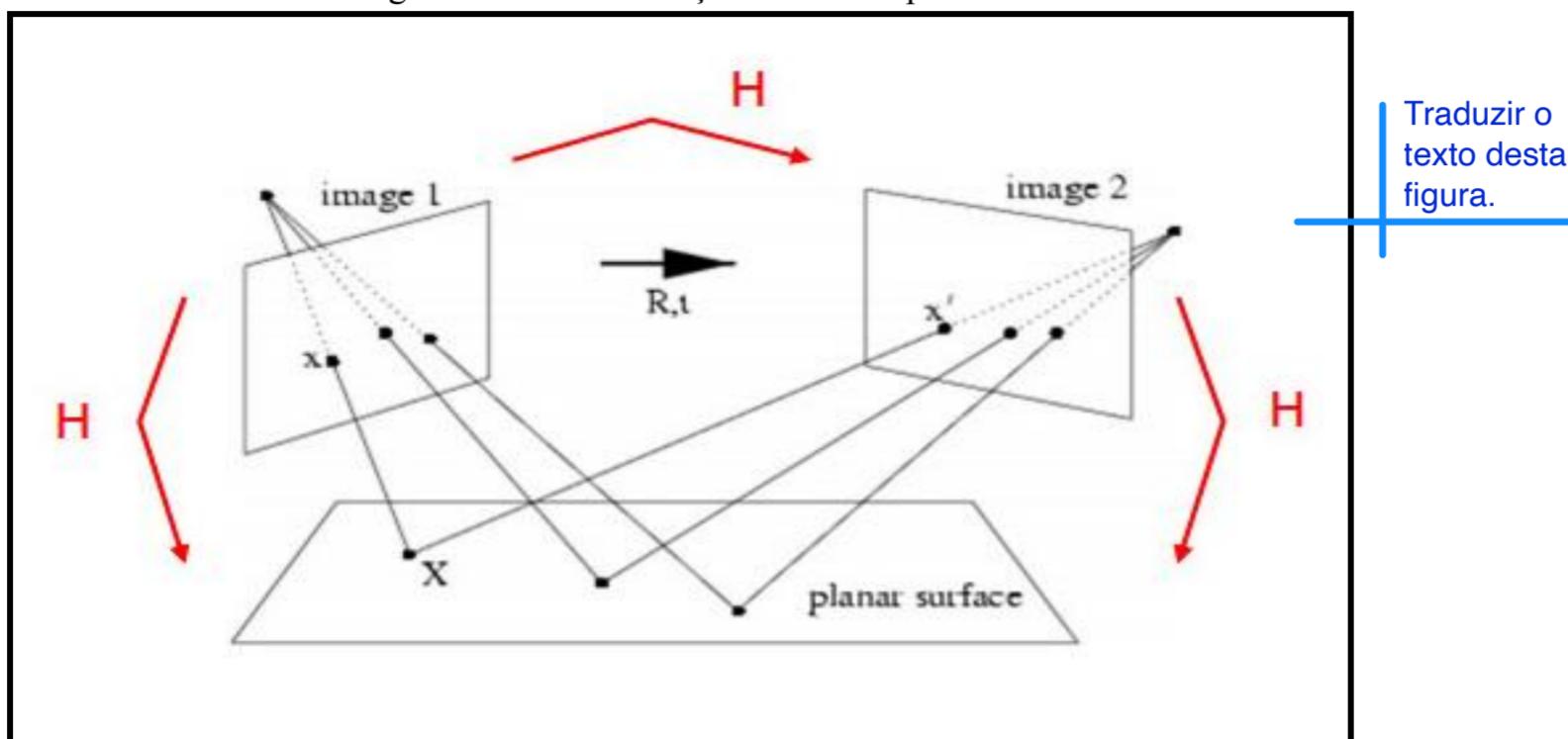
19 caso

20 Desta forma o algoritmo se

Figura 2 – Transformação entre dois planos

Melhorar a explicação desta figura.

Traduzir o texto desta figura.



Fonte: Opencv (2020)

2.4 TRANSFORMAÇÃO DE PERSPECTIVA

A transformação de perspectiva é usada para mudar a perspectiva de um objeto, por exemplo, quando os olhos humanos veem um objeto de perto, eles parecem serem maiores em relação aos objetos mais longes, isso é chamado de perspectiva. No geral, a transformação da perspectiva trata da conversão do mundo 3D em imagem 2D. O mesmo princípio sobre o qual a visão humana e câmera funcionam (TUTORIALSPPOINT, 2013).

Em transformação de perspectiva, precisamos fornecer os pontos da imagem a partir dos quais queremos coletar informações, alterando a perspectiva. Também precisamos fornecer os pontos dentro dos quais queremos exibir nossa imagem. Em seguida, obtemos a transformação de perspectiva dos dois conjuntos de pontos fornecidos e a envolvemos com a imagem original (GEEKSFORGEEKS, 2020).

A Figura 3 apresenta um exemplo utilizando a transformação de perspectiva, que tem os pontos iniciais [[50,0], [150,0], [200,200], [200,200]] e deve ser transformada para os novos pontos [[0,0], [200,0], [0,200], [200,200]].

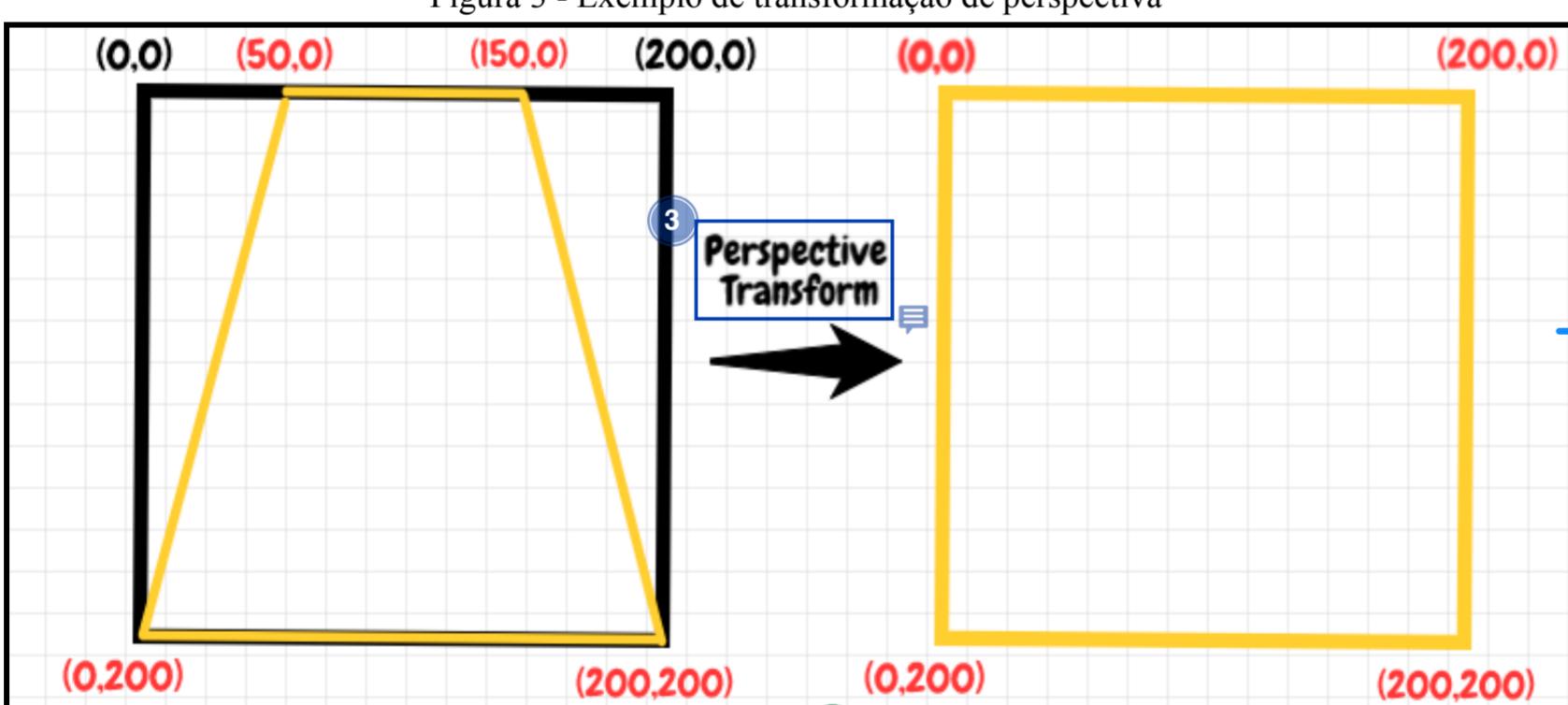
Figura 3 - Exemplo de transformação de perspectiva

Continuar a explicação... parece estar truncado.

Podes diminuir um pouco a fonte do texto desta imagem para ficar no mesmo tamanho da figura anterior.

Ponto final.

perspectiva precisamos



Fonte: Shaikh (2020)

2.5 FILTRO DE CANNY

Algoritmo de Canny foi desenvolvido em 1986 por John F. Canny. O detector de bordas de Canny é reconhecido como um dos mais rápidos e eficientes algoritmos para encontrar descontinuidade em uma imagem. A partir de um conjunto de parâmetros, ele produz como resultado uma imagem binária contendo as bordas obtidas a partir de uma imagem (BRITO, 2015).

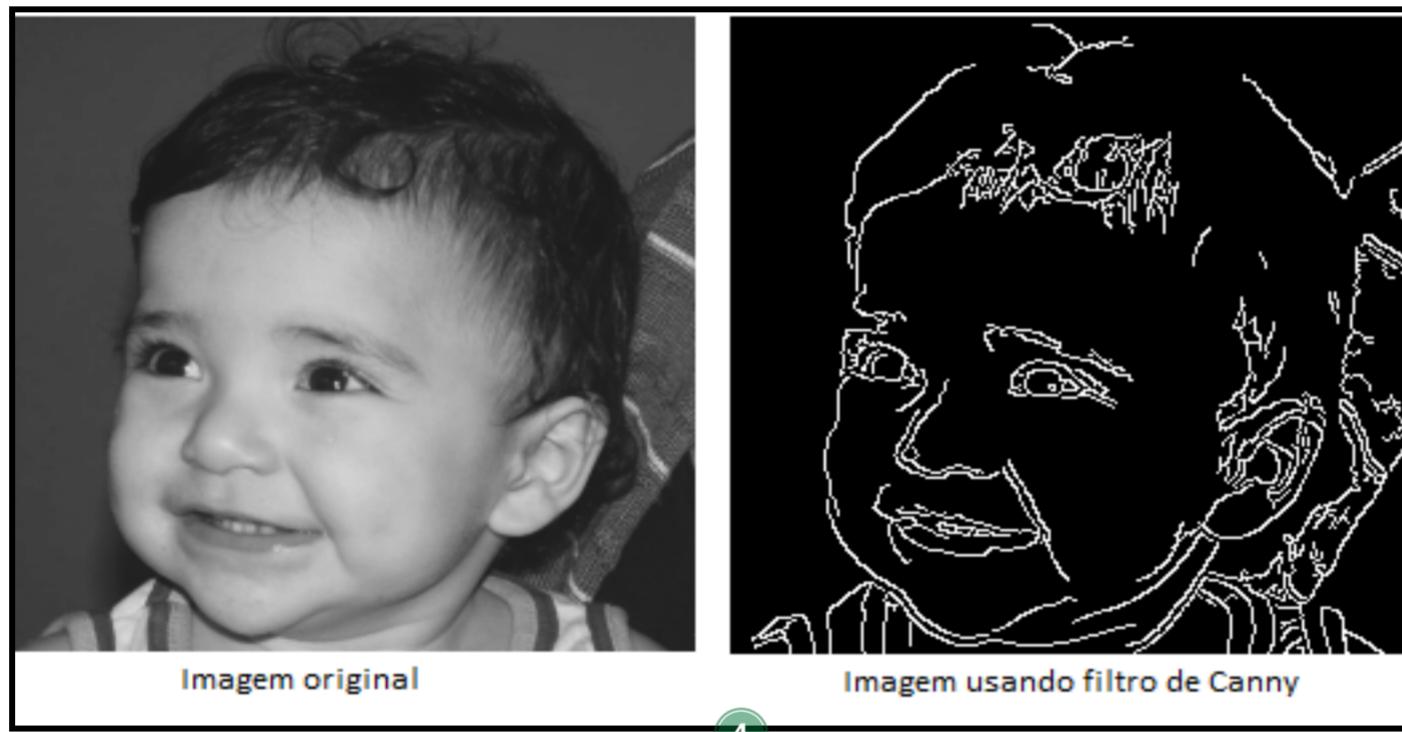
A explicação ficaria mais fácil de entender se tivesse uma figura para ilustrar esta explicação.

O primeiro passo do algoritmo de filtro de Canny é remover o ruído da imagem com um filtro gaussiano, isso se faz necessário devido a detecção de bordas ser suscetível a ruídos na imagem. Após isso a imagem é filtrada com um kernel Sobel na direção horizontal e vertical para obter a primeira derivada na direção horizontal e na direção vertical. A partir dessas duas imagens, pode-se encontrar o gradiente de borda e a direção de cada pixel (BRITO, 2015).

Em seguida uma varredura completa da imagem é feita para remover quaisquer pixels indesejados que possam não constituir a borda. Para isso, a cada pixel é verificado se ele é um máximo local em sua vizinhança na direção do gradiente, se o valor da magnitude do gradiente for inferior a pelo menos um de seus vizinhos, é colocado 0, caso contrário é considerado para próxima etapa (BRITO, 2015).

O último passo, a Limiarização com histerese é usada para a quebra do contorno, ou seja, decidir quais arestas são realmente arestas e quais não são. Para isso, precisa-se de dois valores de limite, `minVal` e `maxVal`. Quaisquer arestas com gradiente de intensidade maior que `maxVal` são consideradas arestas e aquelas abaixo de `minVal` são consideradas não arestas, portanto, são descartadas. Aqueles que estão entre esses dois limites são classificados como bordas ou não com base em sua conectividade. Se eles estiverem conectados a pixels de "aresta segura", eles serão considerados parte das arestas. Caso contrário, eles também são descartados (BRITO, 2015). A Figura 4 mostra um exemplo da utilização do filtro de Canny.

Figura 4 - Exemplo utilização filtro de Canny



Fonte: Brito (2015)

2.6 TRABALHOS CORRELATOS

Foram selecionados quatro trabalhos correlatos que apresentam semelhanças com o proposto neste trabalho. Na seção 2.1 é apresentado o iAR ferramenta desenvolvida por Hess (2011), capaz executar os recursos da realidade aumentada na plataforma iOS. Na seção 2.2 é apresentada a ferramenta animar, desenvolvida por Reiter (2018) que tem como objetivo criar cenas animadas utilizando os recursos da realidade aumentada. Por fim, na seção 2.3 é apresentado um protótipo desenvolvido por Kumagai (2011), utilizando técnicas de reconhecimento de objetos sem marcadores.

Quadro 1 - iAR

Referência	Hess (2011)
Objetivos	Executar os recursos da realidade aumentada na plataforma iOS, que disponibilize opções para a interação do usuário com os objetos virtuais na tela.
Principais funcionalidades	iAR foi desenvolvida em formato Application Programming Interface (API), com funções para facilitar a reutilização da ferramenta para qualquer objetivo.
Ferramentas de desenvolvimento	A ferramenta foi desenvolvida em C++ e explora o uso de algumas ferramentas como OpenGL ES versão 2.0 para apresentação de objetos 3D, bibliotecas como ArUco para identificar a posição e a orientação dos objetos 3D a partir de marcadores, OpenCV biblioteca para tratamentos de imagens digitais e Libobj um analisador de arquivos com extensão .obj, que auxilia na criação de objetos 3D a partir de definições geométricas contidas no arquivo.
Resultados e conclusões	Segundo Hess (2011, p.58), foi identificado problemas na geração de matrizes de visualização e de projeção inconsistente devido ao tamanho e orientação das imagens capturadas, sendo necessário forçar o usuário a utilizar a aplicação sempre na vertical, além de redimensionar a imagem para o tamanho necessário. Apesar de alguns aspectos apresentarem bons desempenhos, Hess (2011, p.59) afirma serem necessárias alterações para tornar a ferramenta mais produtiva.

Fonte: elaborado pelo autor.

1 imagens se pode encontrar

2 Itálico.

3 Itálico.

4 Ponto final.

5 três

A Figura 5 ilustra a aplicação sendo executada, é possível observar dois botões na parte inferior da tela. O botão do lado esquerdo, permite ao usuário selecionar entre os modos de visualização 3D. O segundo botão mostra as informações de desempenho dos recursos da realidade aumentada, como tempo de execução de detecção de marcadores e o número de quadros por segundos atingidos (HESS, 2011, p.57).

Figura 5 - Quadro da aplicação em execução



Fonte: Hess (2011, p.57) 2 2

Quadro 2 – ANIMAR

1 modos de visualização arramado e 3D.

2 Ponto final.

Referência	Reiter (2018)
Objetivos	A ferramenta tem como objetivo criar cenas, adicionar objetos e gravar animações, utilizando os conceitos da realidade aumentada e Interfaces Tangíveis.
Principais funcionalidades	Permite a criação e manipulação de cenários e objetos tridimensionais virtuais, sendo possível dar “vida” à cena ao utilizar Interfaces Tangíveis para a criação de animações dos objetos virtuais.
Ferramentas de desenvolvimento	Utilizando engine gráfica Unity em conjunto com Microsoft Visual Studio Ultimate 2013 como editor de código, Vuforia foi utilizado para a realidade aumentada, Marker Generator by Brosvision para criação dos marcadores e Adobe Photoshop para edição de imagens.
Resultados e conclusões	Reiter realizou testes diretamente com alunos do curso de Pedagogia. Todos conseguiram realizar a atividade proposta, porém o teste do seletor de cenas foi o mais difícil, 28,7% dos alunos tiveram dificuldades em realizar o teste (REITER, 2018, p.69). Segundo Reiter (2018, p.69), os resultados foram satisfatórios, mesmo os alunos apresentarem uma certa dificuldade na utilização da aplicação, isso devido a maioria dos alunos não utilizarem recursos de realidade aumentada e Interfaces Tangíveis. Segundo Reiter (2018, p.71) este é o passo mais difícil de ser executado, pois, há uma maior interação com a Interface Tangível 3 4 5

Fonte: elaborado pelo autor.

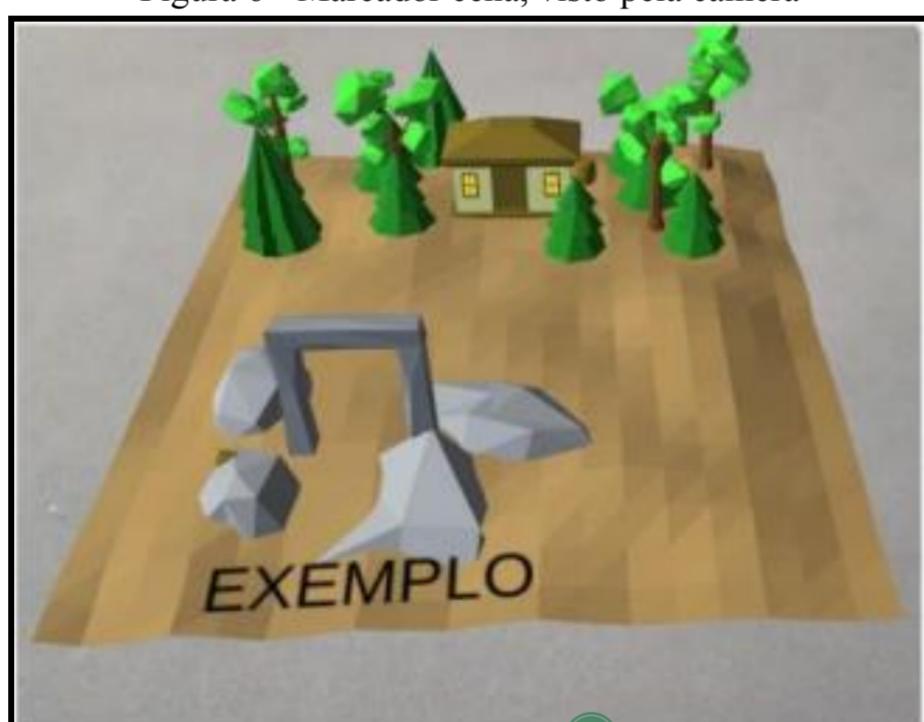
3 Reiter (2018) realizou

4 difícil, pois 28,7%

5 Ponto final.

A Figura 6 mostra um exemplo da aplicação em execução, neste cenário a câmera está sendo apontada para um marcador e aplicação está sobrepondo uma representação gráfica (Reiter 2019, p.63).

Figura 6 - Marcador cena, visto pela câmera



Fonte: Reiter (2019, p.63) 6

6 Ponto final.

Quadro 3 - Estudo e implementação de técnicas de realidade aumentada

Referência	Kumagai (2011).
Objetivos	Possibilitar a utilização dos recursos da realidade aumentada sem marcadores.
Principais funcionalidades	Permite a inserção dos elementos virtuais no ambiente real, utilizando informações naturalmente presentes como arestas, texturas ou a própria estrutura da cena sem a necessidade de marcadores.
Ferramentas de desenvolvimento	Para implementação do projeto Kumagai (2011, p.33) utilizou a API OpenCV com a ferramenta Microsoft Visual Studio 2010 com a linguagem C++. Já a biblioteca OpenGL foi utilizada para desenhar os objetos 3D na tela.
Resultados e conclusões	A técnica de reconhecimento de objetos foi implementada com sucesso e mostrou-se robusta a movimentos suaves e variação de luminosidade mesmo com classificador de qualidade baixa. Porém, em algumas cenas o classificador identifica áreas de interesse mesmo quando a mesma não é de interesse. O protótipo implementado possui algumas limitações, como, de não projetar mais de um objeto virtual na mesma cena, e pela questão de variação de luminosidade e movimentos bruscos, sendo necessário equipamentos de alta qualidade. Kumagai (2011, p.42).

Fonte: elaborado pelo autor.

1 Remover ponto final.

2 qualidade (KUMAGAI, 2011, p.42).

3 das

4 mesmo. A partir

5 Sobrepõe Iniciar com maiúsculo e tem acento.

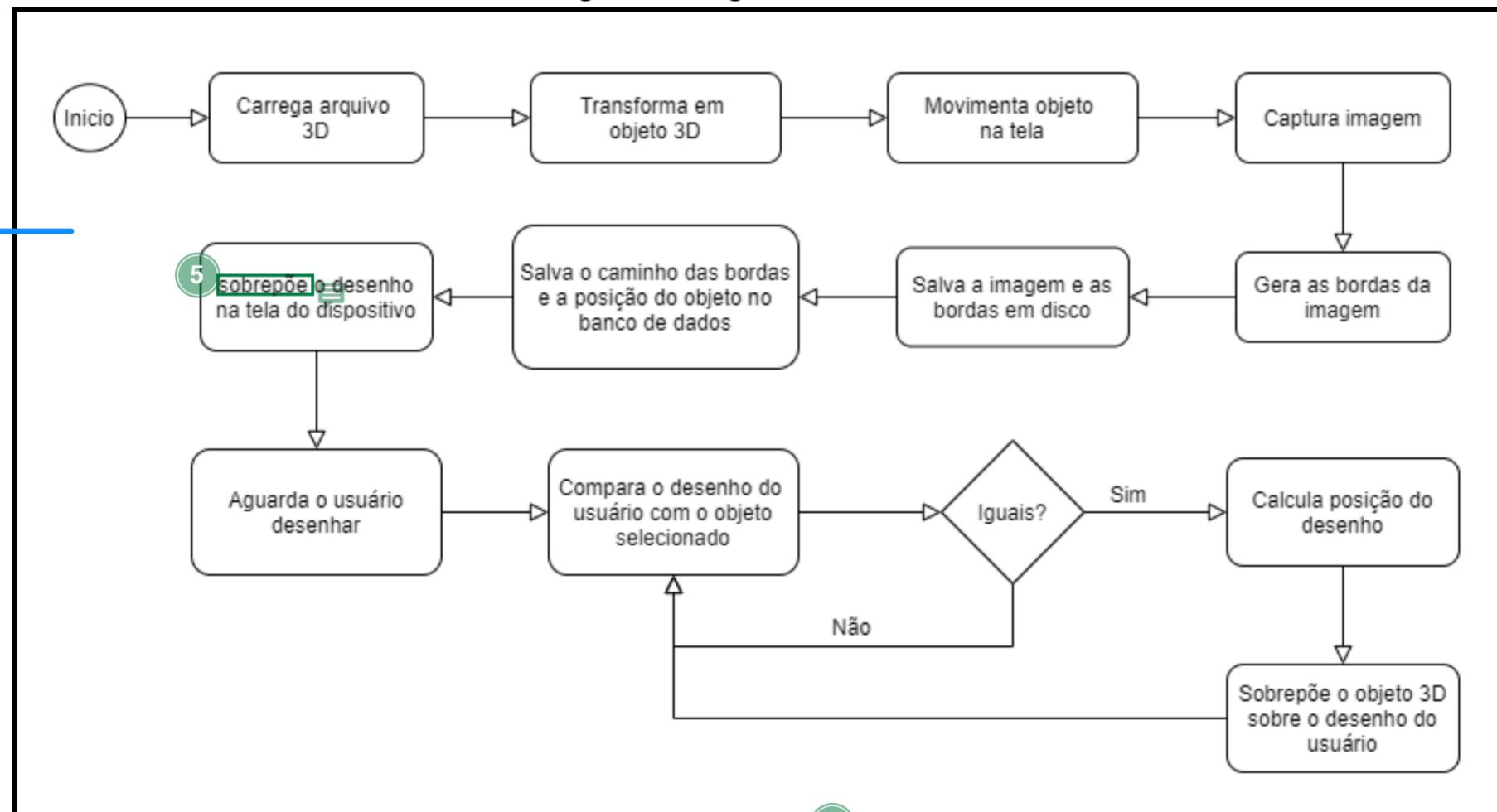
3 DESCRIÇÃO DO APLICATIVO

Neste capítulo é descrito o que foi desenvolvido e uma visão geral do aplicativo. A primeira seção apresenta uma visão geral do aplicativo, bem como seu objetivo, funcionamento e forma de utilização do aplicativo por parte do usuário. Na segunda seção é destacado as principais técnicas implementadas para a construção de funcionalidades.

3.1 VISÃO GERAL DO APLICATIVO

O aplicativo tem como principal objetivo permitir que qualquer usuário possa criar sua própria cena 3D com realidade aumentada. Ele permite ao usuário importar para o aplicativo um arquivo 3D baixado da internet ou criado por ele mesmo, a partir disso o aplicativo converte esse arquivo para um objeto 3D do Unity sendo possível selecionar, definir sua posição e criar uma cena 3D com ele para posteriormente ser desenhado pelo usuário e criado uma sobreposição no dispositivo com realidade aumentada. A Figura 7 detalha a sequência das atividades do aplicativo.

Figura 7 - Diagrama de atividade



Fonte: elaborado pelo autor.

6 Ponto final.

7 Existem outras extensões mesmo?

detecta marcadores ArUco e cenas do mundo real e compara com objetos criados no aplicativo, caso a detecção retorne com sucesso, um objeto 3D vinculado ao marcador é sobreposto na tela do dispositivo. Por fim o menu Desenhar novamente, que permite o usuário desenhar uma cena já criada nos menus anteriores. A Figura 8 detalha a tela do menu principal.

1 aplicativo. E caso

Figura 8 – Menu do aplicativo



Fonte: elaborado pelo autor.

As opções Criar cena sem marcador e Criar cena com marcador, tem certa semelhança, nelas são apresentadas uma lista de objetos 3D adicionados no aplicativo sendo possível selecionar qual objeto será adicionado na cena para posteriormente ser desenhado pelo usuário, existe também um botão de Voltar que volta para o menu principal do aplicativo. A Figura 9 mostra a tela que lista todos os objetos 3D no aplicativo e o botão Voltar.

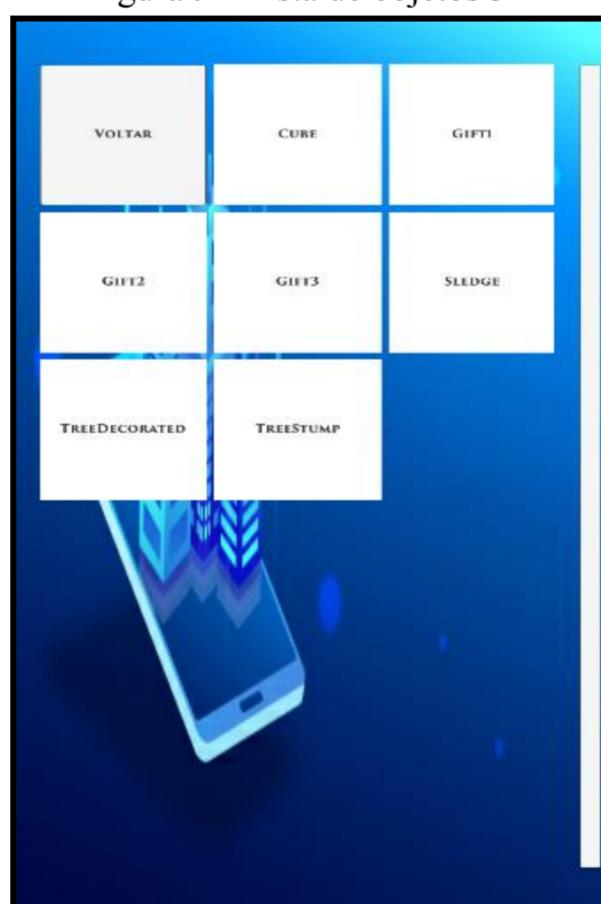
2 marcador tem

3 semelhanças. Nas

4 aplicativo, sendo

5 usuário. Existe

Figura 9 - Lista de objetos 3D

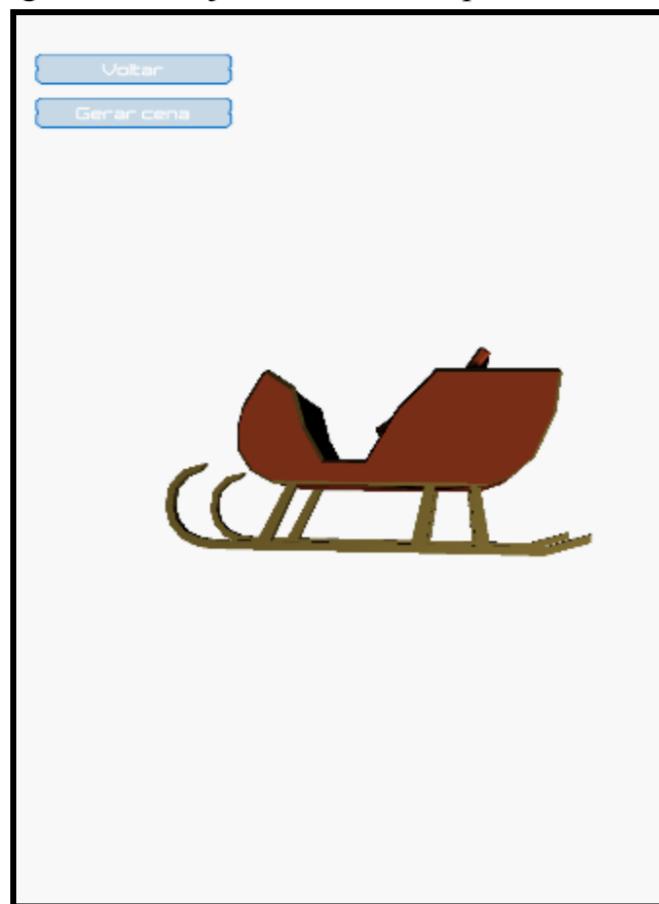


Fonte: elaborado pelo autor.

Após selecionar um objeto, o usuário será redirecionado para a próxima tela, que permite o usuário rotacionar o objeto 3D na posição desejável. Nesta tela existe o botão de Voltar que volta para a lista de objetos 3D e o botão de

Gerar cena, que gera uma cena para que o usuário possa desenhar o objeto selecionado. A Figura 10 mostra a tela de rotacionar o objeto 3D.

Figura 10 - Objeto selecionado para rotacionar



Fonte: elaborado pelo autor

Ao clicar em Gerar cena o aplicativo irá gerar as bordas do objeto selecionado na posição selecionada e sobrepor na câmera do dispositivo permitindo que o usuário desenhe em um papel tendo como base o desenho na tela. A Figura 11 mostra um exemplo do usuário desenhando após selecionar e rotacionar o objeto 3D.

Figura 11 - Exemplo do usuário desenhando pela aplicação

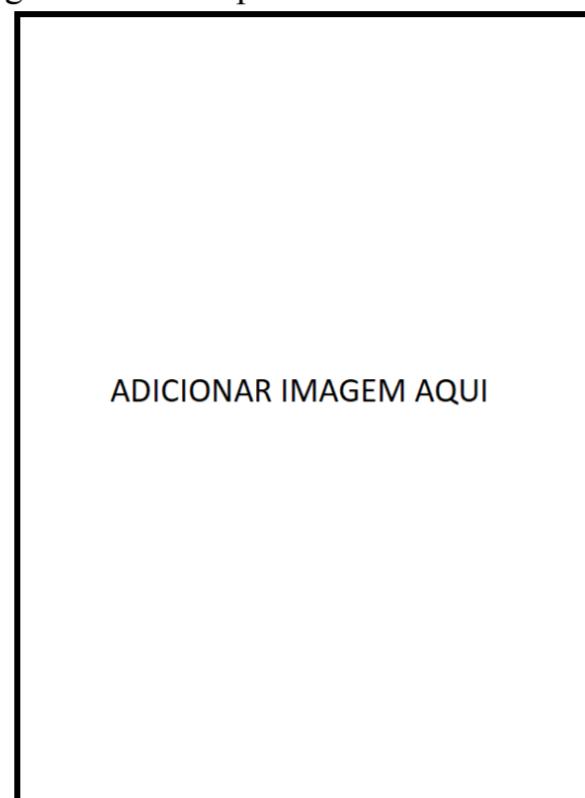


Fonte: elaborado pelo autor

A diferença entre as opções de Gerar cena sem marcador e Gerar cena com marcador está na tela da Figura 11, quando selecionado a opção Gerar cena com marcador é criado um marcador através da biblioteca ArUco e fixado na parte superior direito da tela, como mostra o exemplo na Figura 12.

Ponto final.

Figura 12 - Exemplo da cena com marcador

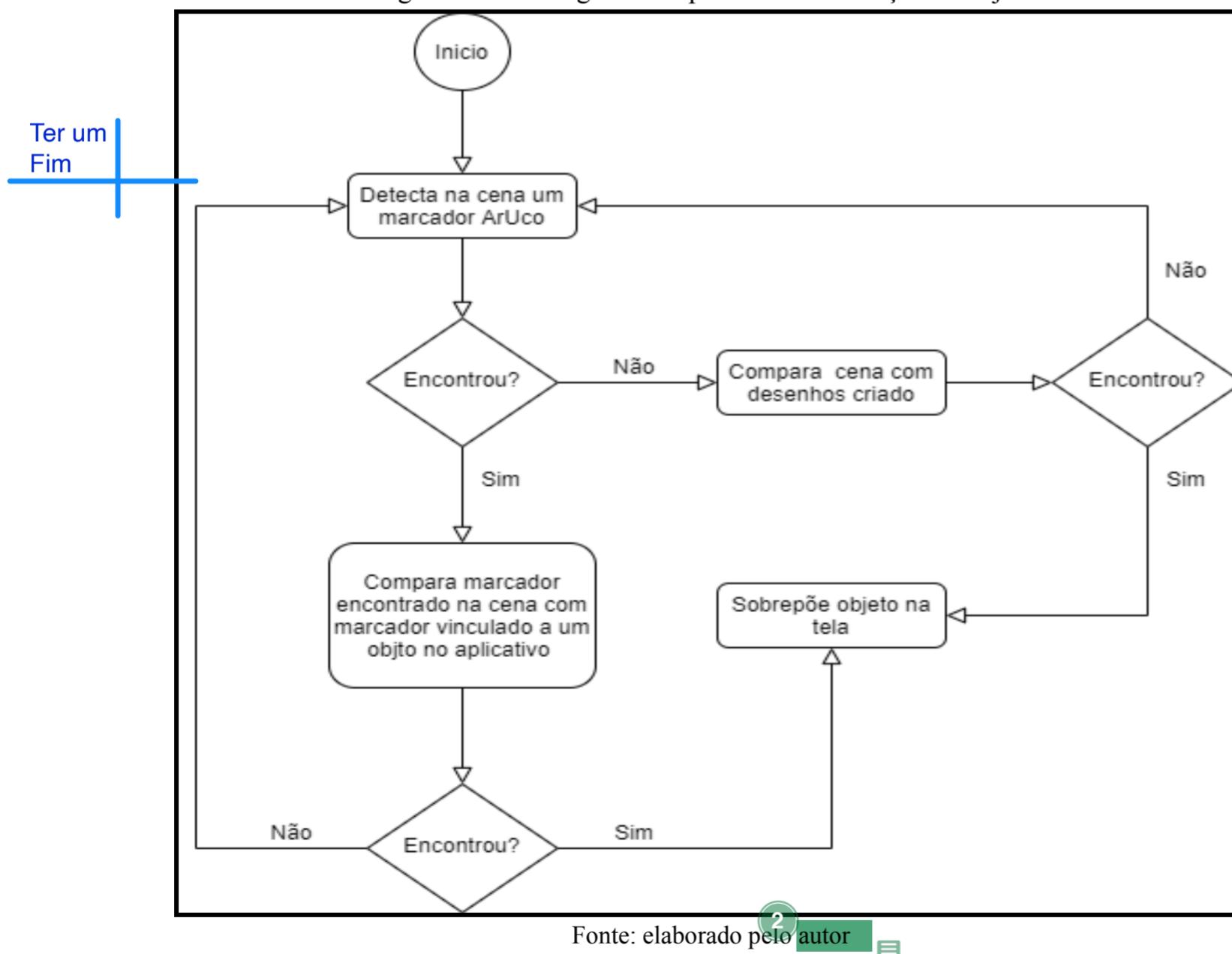


Fonte: elaborado pelo autor 1

Nesta tela existe o botão de **Voltar** que volta para o menu principal do aplicativo e o botão de **Detectar desenho** que redireciona para a tela de realidade aumentada que tem o mesmo efeito do botão **Realidade aumentada** no menu principal. Nesta tela o aplicativo tenta detectar um marcador ArUco, caso não encontre, irá comparar uma cena do mundo real com objetos criados no aplicativo. A Figura 13 detalha a sequência das atividades do aplicativo no processo de detecção de objetos, enquanto a Figura 14 mostra o aplicativo detectando um objeto e sobrepondo-o na tela do dispositivo.

Ponto final. 1

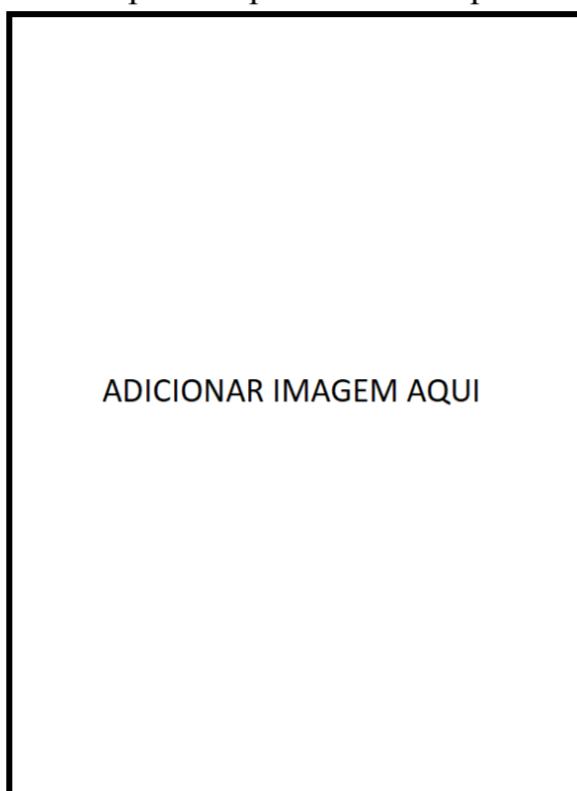
Figura 13 – Fluxograma do processo de detecção de objetos



Fonte: elaborado pelo autor 2

Ponto final. 2

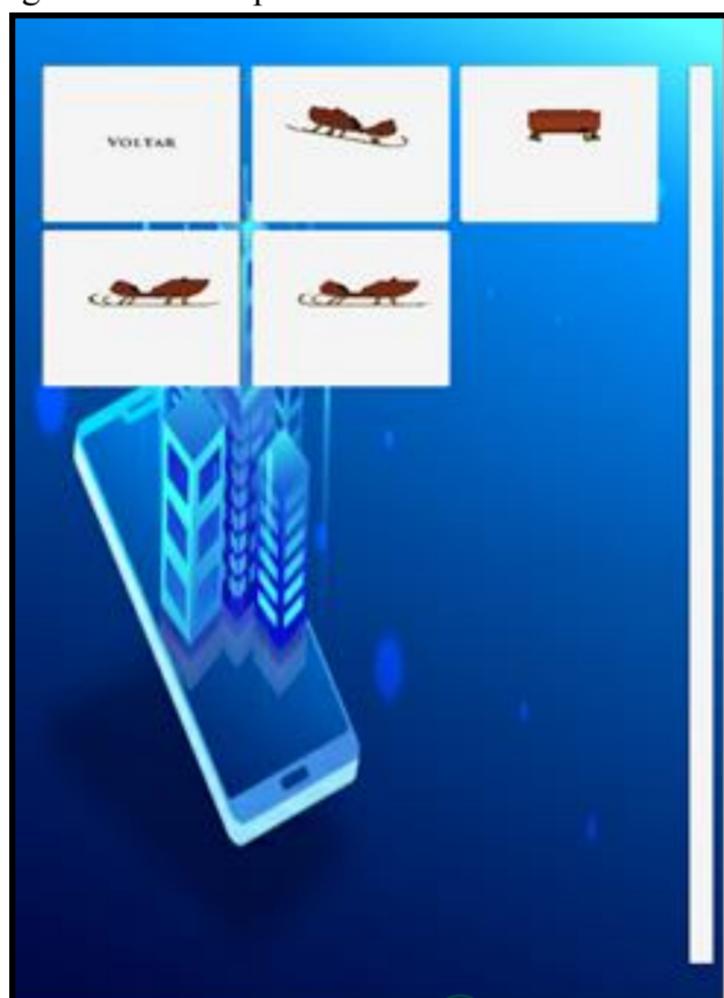
Figura 14 - Exemplo do aplicativo sobrepondo um objeto



Fonte: elaborado pelo autor 1

Na última opção do menu principal Desenhar novamente, traz uma lista de objetos já criados anteriormente através dos menus Gerar cena sem marcador e Gerar cena com marcador. Ao ser clicado em um dos itens da lista, o usuário será redirecionado para tela de desenho como na Figura 11, sendo possível fazer um novo desenho para ser detectado, a Figura 15 mostra a tela de seleção de objetos já criados. 2

Figura 15 - Exemplo de a tela desenhar novamente



Fonte: elaborado pelo autor 4

Ponto final. 1

Dois pontos finais. 2

detectado. A Figura 3

Ponto final. 4

3.2 IMPLEMENTAÇÃO

Para o desenvolvimento da aplicação foi utilizado a plataforma Unity 3D na versão 2019.3.10f1 Personal com scripts na linguagem C# juntamente com a plataforma Visual Studio Community para controle dos componentes das cenas e OpenCV para realização do processamento de imagem.

Foi utilizado o asset MarkerLess AR Example como base para o desenvolvimento da realidade aumentada sem marcadores pré-definidos e o asset MarkerBased AR Example como base para o desenvolvimento da realidade aumentada com marcadores ArUco, ambos os assets estão disponíveis na Unity Asset Store para download gratuito, porém tem como requisito para funcionamento o asset OpenCVForUnity. 5

OpenCVForUnity que é pago (ver seção 2.1).

Para captura das imagens através da câmera foi utilizado o método `GetMat` através do componente `WebCamTextureToMatHelper` que retorna um objeto do tipo `Mat` (específico do OpenCV) contendo a imagem da cena, em seguida é utilizado o método `fastMatToTexture2D` do componente `Utils` que converte um objeto do tipo `Mat` para o tipo `Texture2D` para ser utilizado no Unity.

Para geração das bordas foi utilizado o algoritmo filtro de Canny. O OpenCV possui o método `Canny` do componente `Imgproc` que aplica o algoritmo filtro de Canny em uma imagem do tipo `Mat` e retorna uma imagem com o fundo preto e as bordas da imagem branca, sendo necessários inverter as cores deixando o fundo branco e as bordas pretas. Para isso foi utilizado o método `bitwise_not` que recebe uma imagem do tipo `Mat` e inverte as cores. A Figura 16 mostra o processo para gerar as bordas, sendo a primeira imagem sendo a original, a segunda após o filtro de Canny e a terceira após utilizar o método `bitwise_not` para inverter as cores.

Figura 16 – Resultado algoritmo Canny



Fonte: elaborado pelo autor

Após esse processo, é colocado o resultado das bordas na tela para o usuário desenhar, para isso foi necessário criar um fundo transparente na imagem. O Quadro 4 apresenta o método `GetTransparentTexture` da classe `WebCamDrawing` que tem como objetivo adicionar um fundo transparente à uma textura. Na linha 152 através da Classe `Color` é definido uma cor da estrutura RGBA.

Cada componente de cor é um valor de ponto flutuante com um intervalo de 0 a 1. Os componentes (r, g, b) definem uma cor no espaço de cores RGB, já o componente alfa (a) define uma transparência alfa, sendo 1 totalmente opaco e 0 totalmente transparente (UNITY, 2014). Na linha 160 é feito uma validação, e adicionando um pixel totalmente transparente caso ele seja totalmente branco.

Quadro 4 – Código fonte utilizado para criar uma textura transparente

```

150     public Texture2D GetTransparentTexture(Texture2D texture)
151     {
152         Color transparentColor = new Color(1.0f, 1.0f, 1.0f, 0f);
153
154         for (int y = 0; y < texture.height; y++)
155         {
156             for (int x = 0; x < texture.width; x++)
157             {
158                 if (!Color.black.Equals(texture.GetPixel(x, y)))
159                 {
160                     texture.SetPixel(x, y, transparentColor);
161                 }
162             }
163         }
164
165         texture.Apply();
166         return texture;
167     }

```

Fonte: elaborado pelo autor

A Figura 17 apresenta um exemplo detalhado do resultado da textura sem o método `GetTransparentTexture`, ficando com o fundo todo branco sobrepondo a câmera e como fica após a utilização do método, ficando com o fundo transparente, sobrepondo apenas as bordas do desenho.

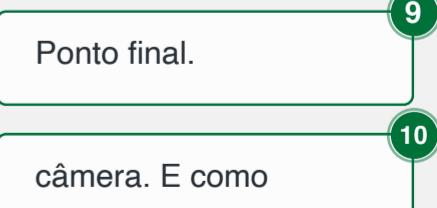
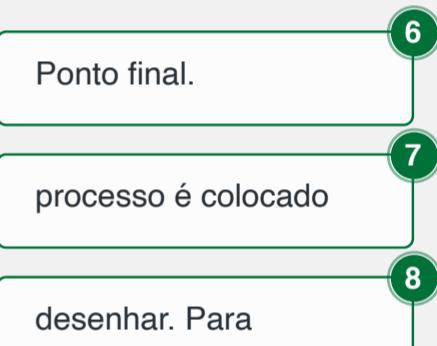
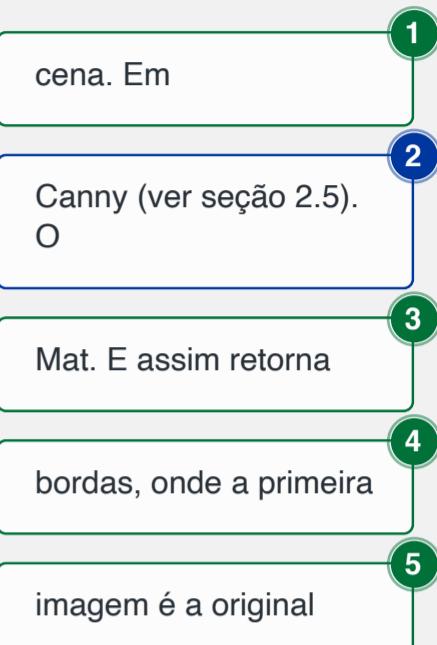
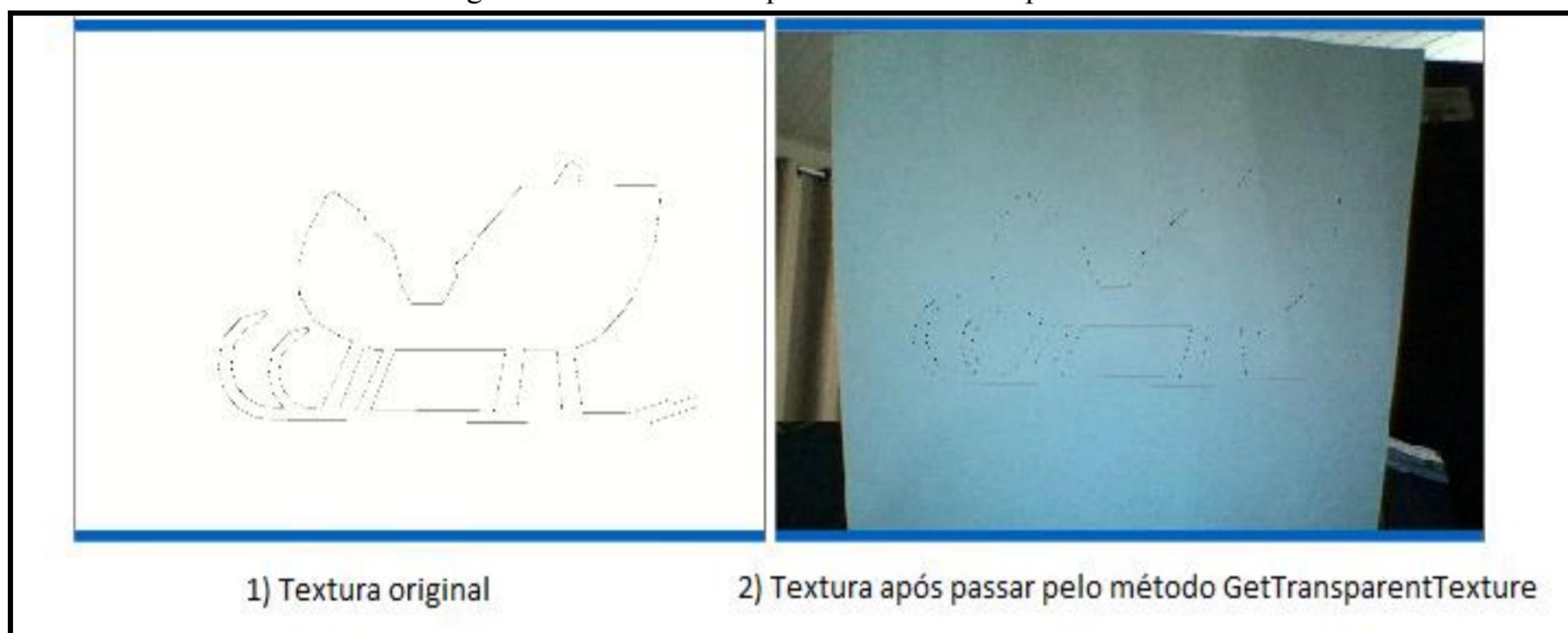


Figura 17 – Bordas sobrepostas na tela do dispositivo



Fonte: elaborado pelo autor

1

Para comparação do desenho feito pelo usuário com o objeto selecionado, foi utilizado o método `findHomography` do componente `Calib3d` do OpenCV, que recebe como primeiro e segundo parâmetro um conjunto de pontos da imagem original e da imagem que está sendo detectada. O terceiro parâmetro é o método usado para calcular a homografia, nesse aplicativo foi utilizado o método `RANSAC`. O quarto parâmetro define a quantidade de erro permitido para tratar os pontos, utilizamos quantidade máxima de 3. O sexto parâmetro define o número de iterações, nesse aplicativo foi utilizado 2000 iterações. E por último é definido o nível de confiança entre 0 e 1, nesse aplicativo foi utilizado 0.5. O método `findHomography` irá encontrar e retornar a transformação de perspectiva entre os pontos chave correspondentes da imagem original e a que está sendo detectada.

Para calcular a posição que o objeto está, foi utilizado o método `perspectiveTransform` do componente `Core` do OpenCV, que recebe como primeiro parâmetro uma matriz de pontos flutuantes do objeto original. O segundo parâmetro é uma matriz de pontos flutuantes que irá receber o resultado da transformação, essa matriz deve ter o mesmo tamanho do elemento original. E por fim, recebe como último parâmetro a matriz de homografia através do método `findHomography`.

REVISADO

S a ampliar o seu caráter científico, todos os TCCs devem apresentar e discutir resultados não limitados os trabalhos correlatos. Devem ser apresentados os casos de testes do software, destacando objetivo realizada a coleta de dados e a apresentação dos resultados obtidos, preferencialmente em forma de gráficos ou tabelas, fazendo comentários sobre os mesmos. Também é sugerida a comparação com os trabalhos correlatos apresentados na fundamentação teórica.

5 CONCLUSÕES

As conclusões devem refletir os principais resultados alcançados, realizando uma avaliação em relação aos objetivos previamente formulados. Deve-se deixar claro se os objetivos foram atendidos, se as ferramentas utilizadas foram adequadas e quais as principais contribuições do trabalho sociais ou práticas para o seu grupo de usuários bem como para o desenvolvimento científico e ou tecnológico da área.

Deve-se incluir também as limitações e as pos

REFERÊNCIAS

ARUCO. **Detection of ArUco Markers**. [S.l.], [2020]. Disp <https://docs.opencv.org/trunk/d5/dae/tutorial_aruco_detection.html>. Acesso em: 14 set. 2020.

AVA. **Aplicaciones de la Vision Artificial**. [S.l.], [2018?], I

Acesso em: 14 set. 2020.

BRITO, Agostinho. **8. Detecção de bordas com o algoritmo Canny**. <https://agostinhobritojr.github.io/tutorial/pdi/#_detec%C3%A7%C3%A3o-de-bordas-com-o-algoritmo-canny>. Acesso em: 15 nov. 2020.



a/node/26>.

Todas as referências bibliográficas devem ser citadas no texto. E todas as citações devem ter uma referência bibliográfica.