Revista SODEBRAS – Volume 14
N° 160 - Abril/2019

DETECÇÃO E RASTREAMENTO DE OBJETOS EM IMAGENS UTILIZANDO DESCRITORES LOCAIS
DETECTION AND TRACKING OF OBJECTS IN IMAGES USING LOCAL DESCRIPTORS

CRISTIAN SIMIONI MILANI
PONTIFÍCIA UNIVERSIDADE CATÓLICA DO PARANÁ
cristiansimionimilani@gmail.com

Resumo – A utilização de descritores locais é uma técnica útil no campo de visão computacional para identificar e rastrear objetos em imagens. Além de ser uma técnica amplamente utilizada na literatura por ser rápida e robusta, esses algoritmos funcionam independentemente da variação de luminosidade, rotação e escala de uma imagem. Esse artigo implementa um sistema utilizando como base o algoritmo SIFT capaz de calcular a distância entre objetos em uma imagem ou a distância inicial e final de um objeto específico em uma cena dinâmica. Apesar de algumas limitações, como a de não considerar a profundidade da imagem e a dificuldade com objetos parcialmente obstruídos o sistema, na maioria dos casos, se mostrou eficiente para calcular a distância entre objetos em imagens.

Palavras-chave: Detecção de Objetos, Rastreamento de Objetos, Visão Computacional, Descritores Locais.

Abstract - The use of local descriptors is an useful technique in computer vision to identify and track objects in images. In addition to being a widely used technique in the literature because it's fast and robust, these algorithms work independently of the variation of luminosity, rotation and scale of an image. This article implements a system based on the SIFT algorithm that is able to calculate the distance between objects in an image or the distance of the initial position and the final position of a specific object in a dynamic scene. Despite some limitations, such as not considering the depth of the image and the difficulty with partially obstructed objects, the system was, in most cases, efficient to calculate the distance between objects in images.

Keywords: Object Detection. Object Tracking. Computer Vision. Local Descriptors.

I. INTRODUÇÃO

Detectar, rastrear e reconhecer objetos em imagens é uma tarefa simples para seres humanos, pois nós somos capazes de reconhecer as mais diferentes formas quase que instantaneamente ao observar uma imagem. Entretanto, dentro da área de visão computacional, apesar de ser um tema amplamente estudado nas últimas décadas, detectar automaticamente objetos em imagens não é uma atividade trivial e requer o uso de algoritmos robustos (KASSIUS, 2015).

Vários problemas práticos requerem a localização e o reconhecimento de objetos dentro de uma sequência de imagens ou vídeos digitais, como por exemplo: vigilância automatizada, pesquisas de conteúdo específico em bancos de dados multimídias ou edição de vídeo (GRACIANO, 2007).

Existem diversas técnicas no campo da visão computacional capazes de detectar e rastrear objetos em imagens, dentre elas destacam-se: SIFT – *Scale-Invariant Feature Transform* (LOWE, 2004) e SURF – *Speeded-Up Robust Features* (BAY, 2008). Ambas são amplamente usadas na literatura pois são robustas em relação a capacidade de detectar pontos notáveis (*keypoints*) em uma imagem, independentemente da luminosidade, rotação ou escala.

Neste cenário, o objetivo desta pesquisa foi desenvolver um sistema capaz de detectar objetos em uma cena utilizando a técnica SIFT e mensurar a distância entre eles em pixels. Além disso, seguindo a mesma ideia também foi possível mensurar o deslocamento de um objeto específico em uma sequencia de imagens considerando um cenário dinâmico, isto é, onde os objetos se movimentam.

Esse sistema pode ser útil para qualquer aplicação que requer a necessidade, em um ambiente dinâmico, de identificar a distância percorrida de um objeto, como por exemplo robôs autônomos ou medição de distância de objetos em imagens aéreas.

Este artigo encontra-se organizado da seguinte forma: a seção 2 apresenta a fundamentação teórica. A seção 3 descreve detalhadamente o método utilizado. Os experimentos e resultados são apresentados na seção 4. Por fim, a conclusão e as referências são apresentadas nas seções 5 e 6, respectivamente.

II. FUNDAMENTAÇÃO TEÓRICA

2.1 – Detecção de Características

Característica é o termo em visão computacional que se refere a uma parte de imagem com alguma individualidade, por exemplo, uma linha, um círculo, pontos, curva, superfícies e regiões de textura (TRUCCO, 1998). As características extraídas de uma imagem podem ser utilizadas em diferentes frentes: detecção de movimento, reconhecimento de objetos, mapeamento, entre outros.

Para que essas características possam ser extraídas e utilizadas existem os algoritmos que são conhecidos como extratores de características (*feature extractors*). Esses algoritmos fornecem como saída um conjunto de descritores (*feature descriptors*), os quais especificam a posição e outras propriedades essências das características encontradas na imagem (KLIPPENSTEIN, 2007).

ISSN 1809-3957

Dentre os algoritmos mais conhecidos e utilizados para a detecção de características destacam-se o SIFT, SURF, ORB, FAST e FREAK, pois são considerados eficientes.

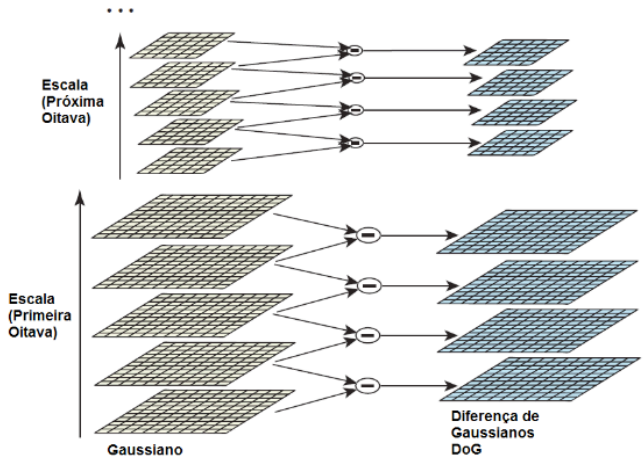
A grande vantagem desses algoritmos está relacionada a capacidade de suportar variações de escala e rotação da imagem. Ou seja, independentemente da escala, luminosidade ou rotação em uma determinada imagem o algoritmo será capaz de detectar os pontos notáveis (*keypoints*).

2.2 – Scale-Invariant Feature Transform (SIFT)

Um dos primeiros descritores locais baseado em gradiente foi proposto por (LOWE, 2004). Esse algoritmo, o SIFT, é dividido em algumas etapas: detecção de extremos, localização de pontos notáveis, atribuição de orientação aos descritores e, por fim, a construção do descritor local.

Durante a etapa de detecção de extremos o algoritmo procura características estáveis em diferentes escalas utilizando a função gaussiana. Basicamente são formadas cinco imagens com cinco níveis de desfoque diferentes que passam a formar uma oitava. O processo é repetido até que se criem quatro oitavas, formando assim o espaço de escalas. A eficácia da busca pelos pontos notáveis é aumentada utilizando a função DoG (*Difference of Gaussian*). Nesse ponto as imagens consecutivas de uma oitava são subtraídas para produzir uma imagem da DoG. Esse processo se repete para todas as oitavas. A Figura 1 demostra essa etapa do processo.

Figura 1 – Representação da obtenção das Diferenças de Gaussianas (DoG)

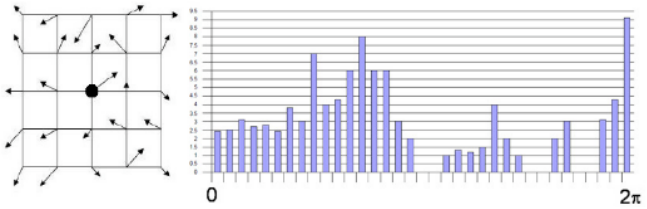


Fonte: (LOWE, 2004)

Nessa etapa do algoritmo inicia-se o processo de busca pelos pontos notáveis. Para cada pixel de uma imagem da DoG, faz-se a comparação dele com seus vizinhos, tanto na imagem atual, quanto nas imagens acima e abaixo, dentro do espaço de escala. Se o pixel examinado é um máximo ou mínimo, então ele é um potencial candidato a ponto notável. Após localizar os candidatos é realizado um refinamento através da aplicação da Série de Taylor. Esse processo elimina os pixels cujo valor de intensidade é menor que o limiar estabelecido. Lowe (2004) aconselha trabalhar com o valor de limiar 0,03, assumindo que os tons de cinza dos pixels estejam normalizados entre 0 e 1.

Em seguida é atribuído uma orientação para cada um dos descritores, a qual será utilizada mais tarde para se construir descritores invariantes quanto à rotação. Ou seja, mesmo que a imagem seja girada em qualquer direção, os pontos notáveis serão identificados. Para tal, é computando a direção do gradiente e magnitude dos pixels. Então, monta-se um histograma das orientações para os pixels vizinhos ao redor do ponto notável, conforme a Figura 2.

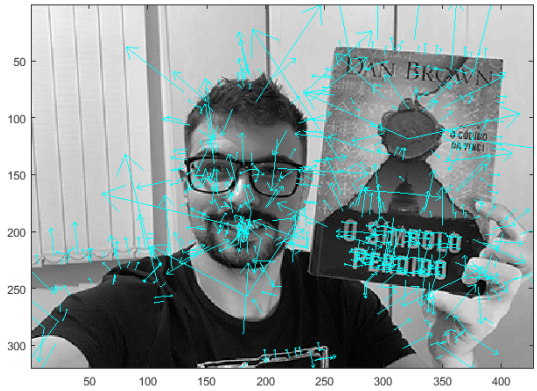
Figura 2 – Histograma de orientações de um ponto notável



Fonte: (LOWE, 2002)

O pico do histograma é utilizado para definir a orientação do ponto notável. No caso de muitos picos de elevada magnitude, o ponto notável pode receber múltiplas orientações. A Figura 3 apresenta diversos pontos notáveis e suas respectivas magnitudes e orientações representadas por vetores.

Figura 3 – Atribuição de orientação e magnitude para pontos notáveis de uma imagem



Fonte: (Autor, 2019)

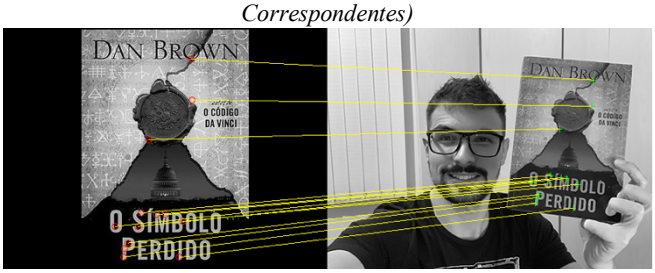
A etapa final do algoritmo consiste na criação do descritor local. Uma janela de 16x16 pixels é criada ao redor do ponto notável, em seguida essa região é dividida em dezesseis janelas de apoio de 4x4 pixels. Dentro de casa janela de apoio são calculadas as magnitudes do gradiente e as orientações. As orientações são colocadas em um histograma de oito divisões, resultando um total de 128 intervalo de valores. Uma vez que se tem os 128 valores, normaliza-se todos, resultando no descritor do ponto notável detectado, também chamado de *feature vector*. Tem-se como resultado final um conjunto de descritores robustos que podem ser usados para fazer a correspondência da imagem em outra imagem.

2.3 – Matching: Pontos Notáveis Correspondentes

O conceito de *matching* baseia-se em extrair pontos notáveis de duas imagens e encontrar os pontos correspondentes em cada uma. Este é um conceito chave para diversas aplicações de visão computacional, como por exemplo a detecção de movimento de objetos em uma série de imagens. A Figura 4 exibe um exemplo do conceito de *matching*.

ISSN 1809-3957

Figura 4 – Exemplo do conceito de *Matching (Pontos Notáveis Correspondentes)*



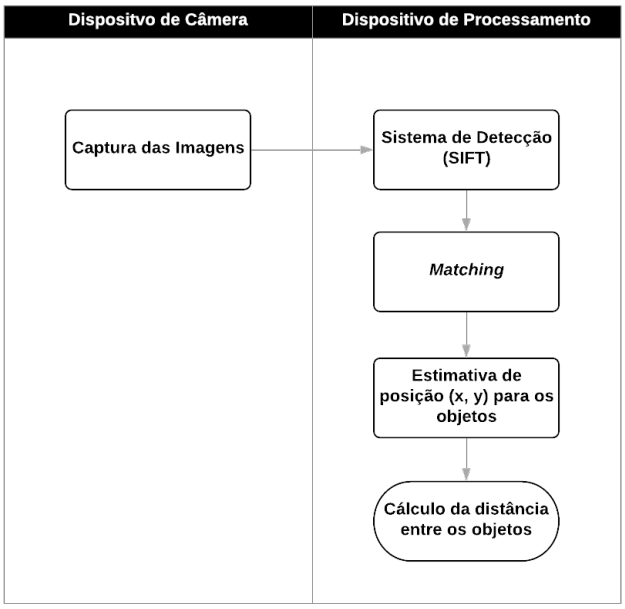
Fonte: (Autor, 2019)

Como citado anteriormente, o algoritmo SIFT retorna os pontos notáveis de uma imagem através de um vetor de descritores locais. Ao extrair os descritores locais de duas imagens é possível comparar seus respectivos vetores utilizando a distância Euclidiana, por exemplo. Lowe (2004) utilizou o algoritmo *Best-Bin-First* (BBF) para identificar os vizinhos mais próximos com elevada probabilidade e sem muito custo computacional. Entretanto, esse método geralmente produz algumas combinações falso-positivas (*outliers*). Visando eliminar o máximo possível dessas combinações falso-positivas, Lowe utilizou um método para comparar a menor distância e com a segunda menor distância e selecionar somente os correspondentes próximos por um limiar. O valor deste limiar sugerido por Lowe é de 0,8, ou seja, toda relação de distância superior a esse valor é automaticamente descartada. Na próxima seção será explicado o método proposto nesse artigo que utiliza o conceito de *matching* para calcular distâncias entre dois objetos.

III. MÉTODO

A possibilidade de detectar e rastrear objetos em imagens se torna interessante para diversas aplicações na área de visão computacional como citado anteriormente. Este artigo propõe-se a apresentar um sistema capaz de calcular a distância entre dois objetos (iguais ou diferentes) em uma série de imagens. A Figura 5 apresenta o fluxo do método proposto.

Figure 5 – Fluxo do método proposto



Fonte: (Autor, 2019)

Na primeira etapa são capturadas diversas imagens de objetos em uma cena dinâmica (objetos variam de posição). A câmera é posicionada em um local fixo, ou seja, todas as imagens são capturadas a partir da mesma posição e com a mesma escala. É importante ressaltar que todas as imagens capturadas têm a mesma dimensão.

Em seguida as cenas capturadas são submetidas ao sistema de detecção que utiliza o algoritmo SIFT para extrair os seus respectivos pontos notáveis. Essa é a parte do processo com maior custo computacional.

Na etapa de *matching* o objeto alvo (aquele que é buscado na cena) também passa pela extração de pontos notáveis e posteriormente é comparado com a cena para determinar a sua localização baseando-se nos pontos notáveis. O método proposto funciona tanto para determinar a distância de um objeto específico em uma série de imagens ou para determinar a distância entre dois objetos diferentes na mesma imagem.

Realizado o processo de *matching* entre o objeto e a cena e, baseando-se na posição (*x,y*) dos pontos notáveis em comum entre a cena e o objeto, calcula-se a média para determinar a localização aproximada do objeto na cena. Vale destacar que a qualidade da localização é bastante dependente do sistema de detecção, uma vez que resultados inconsistentes (como dados *outliers*) impactam em grandes erros na estimativa de localização.

Por fim, com a localização aproximada do objeto na cena, repete-se o processo para outro objeto na mesma cena ou para o mesmo objeto em outra cena e calcula-se por fim a distância (em pixels) aproximada entre os objetos. Uma fragilidade deste método é que ele desconsidera a profundidade (eixo *z*), ou seja, se os objetos estiverem sobrepostos parcialmente no plano, porém distantes, o resultado do algoritmo será zero. Além disso, se os objetos estiverem completamente sobrepostos no plano, o algoritmo não será capaz de localizar o objeto que está atrás.

IV. EXPERIMENTO E RESULTADOS

Utilizando o método descrito anteriormente foram capturadas algumas imagens de objetos (livros) utilizando uma câmera fixa. A Figura 6 mostra um exemplo de cena capturada. A dimensão de todas as imagens capturadas é de 1000x750 pixels.

Figura 6 – Exemplo de cena capturada

Fonte: (Autor, 2019)

ISSN 1809-3957

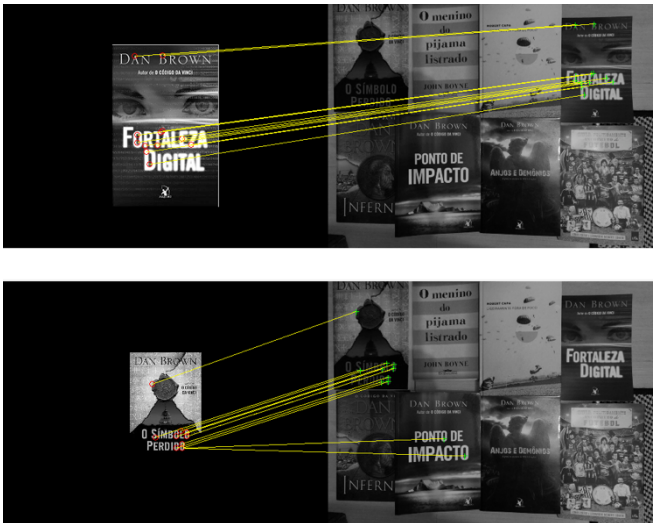
O experimento realizado foi concebido com a ideia de calcular a distância entre os objetos em uma mesma cena e, posteriormente deslocar os objetos (tornando a cena dinâmica) para calcular a distância de descolamento (entre cenas) para um objeto específico.

Para tal, foi desenvolvido um algoritmo no MATLAB utilizando como base as implementações do algoritmo SIFT (LOWE, 2005) e do algoritmo de *SIFTMatch* para determinar a correspondência de pontos notáveis entre duas imagens (ZHANG, 2018).

4.1 – Distância entre Objetos

Considerando o primeiro caso (cálculo de distância entre objetos) e a cena apresentada anteriormente na Figura 6, são extraídos 15 (quinze) pontos notáveis de dois objetos previamente conhecidos da imagem. Posteriormente é realizado o *matching* entre os objetos e a imagem, tendo como resultado a Figura 7. O limiar utilizado para determinar se dois pontos notáveis são correspondentes foi de 0,8.

Figura 7 – *Matching* dos dois objetos fornecidos como entrada com a cena



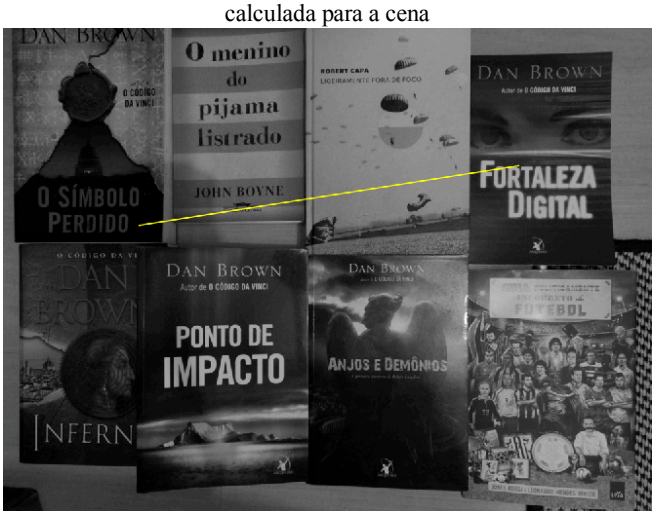
Fonte: (Autor, 2019)

As coordenadas extraídas para a primeira objeto (livro "Fortaleza Digital") são: [*780.02, 243.99; 764.27, 223.56; 789.40, 281.04; 764.05, 237.63; 764.05, 237.63; 857.63, 229.74; 764.27, 223.56; 782.34, 256.86; 819.14, 63.48; 805.40, 243.60; 811.87, 217.09; 816.89, 215.12; 877.12, 241.72; 761.97, 65.65; 780.02, 243.99*], já para a segunda (livro "O Símbolo Perdido") são: [*202.29, 268.41; 165.75, 308.62; 357.50, 482.83; 167.47, 269.03; 185.22, 308.33; 420.52, 536.30; 182.19, 250.56; 167.47, 269.03; 357.50, 482.83; 182.19, 250.56; 86.74, 93.15; 202.33, 268.13; 185.83, 294.53; 98.78, 272.91; 202.53, 255.19*].

Neste ponto vale o destaque para os dados *outliers* (aqueles que não pertencem ao objeto fornecido com entrada para o algoritmo) apresentados na localização do segundo objeto. São as duas coordenadas ([*86.74, 93.15; 98.78, 272.91*]) mais distantes que foram identificadas de forma equivocada e que estão localizadas sobre o livro "Ponto de Impacto". Como o sistema se baseia na média das coordenadas (*x,y*) dos pontos notáveis para determinar a posição estimada do objeto na cena, o segundo objeto teve a sua posição (*x,y*) estimada parcialmente comprometida por conta dos pontos *outliers* apresentados.

A Figura 8 exibe o resultado final do sistema, onde as duas coordenadas estimadas para cada um dos objetos são interligadas por uma linha.

Figura 8 – Resultado final com a distância entre os objetos calculada para a cena



Fonte: (Autor, 2019)

A distância aproximada entre os dois objetos para essa cena é de 592 pixels. Nota-se, que a posição estimada do objeto é próxima aos pontos notáveis extraídos da imagem. Conclui-se então que é possível utilizar o algoritmo SIFT para calcular a distância entre dois objetos em uma imagem, porém se a aplicação exigir uma precisão maior, existirá a necessidade de utilizar métodos adicionais para melhorar a acurácia do sistema. Além disso, trata-se de uma estimativa para os eixos *x* e *y*, ou seja, como já citado anteriormente se o objeto estivesse sobreposto ao outro, mesmo que houvesse uma movimentação no plano o resultado do sistema erroneamente retornaria zero como distância entre os objetos.

4.2 – Movimento de Objetos em uma Série de Imagens

Utilizando o mesmo princípio usado para calcular a distância entre dois objetos em uma imagem também é possível calcular a distância percorrida de um objeto em uma cena dinâmica a partir de uma série de imagens. Considerando a Figura 9 deseja-se descobrir a movimentação do livro "Fortaleza Digital".

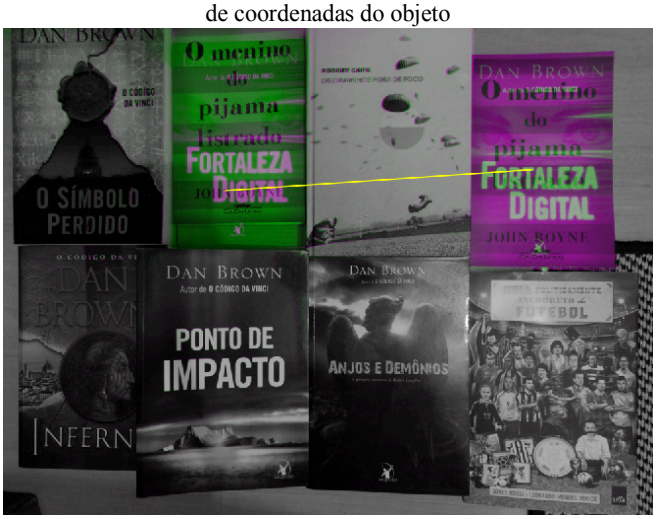
Figura 9 – Cenas utilizadas com entrada para calcular a distância percorrida de um objeto

Fonte: (Autor, 2019)

É possível notar que o objeto alvo se encontra no canto direito superior na imagem (a) e mais ao centro na imagem (b). Repetindo o mesmo processo de *matching* o objeto é localizado em ambas as cenas e posteriormente é calculada a distância "percorrida". O resultado é apresentado na Figura 10.

ISSN 1809-3957

Figura 10 – Cenas sobrepostas com uma linha entre as estimativas de coordenadas do objeto



Fonte: (Autor, 2019)

A distância aproximada percorrida pelo objeto na cena é de 471 pixels. O exemplo exposto pode ser utilizado, como passo inicial para um sistema capaz de, a partir de frames de um vídeo, medir a distância percorrida por um determinado objeto e também calcular a velocidade do mesmo.

V. CONCLUSÃO

Neste artigo foi desenvolvido um sistema capaz de detectar objetos em imagens e calcular a distância entre eles. O sistema se mostrou eficaz considerando um ambiente controlado, onde não havia obstrução aos objetos procurados. Além disso, é necessário reafirmar o que a literatura apresentou: os pontos notáveis extraídos pelo algoritmo SIFT são um excelente referencial para localizar objetos em imagem pois independem da escala da imagem, iluminação e rotação.

As distâncias adotadas nesse sistema foram em pixels, mas sabendo-se a dimensão real da imagem poderiam ser calculadas em uma medida "mais legível", como centímetros ou metros. Isso depende da real necessidade das aplicações que possam vir a utilizar esse sistema. Vale, mais uma vez, deixar explícito que as distâncias calculadas são aproximadas e que para se criar algo mais preciso seria necessário informações adicionais como tamanho do objeto, centro de massa, entre outros.

Como trabalho futuro sugere-se a melhoria na identificação dos pontos notáveis identificados erroneamente o que aumentaria a precisão da coordenada (x,y) estimada para o objeto e, consequentemente, melhoraria a precisão da distância calculada. Também é possível utilizar esse sistema como ponto de partida para mensurar a velocidade de objetos em um vídeo, extraindo as posições do objeto a cada frame e calculando suas respectivas distâncias pelo tempo, por exemplo.

VI. REFERÊNCIAS

BAY, H.; ESS, A.; TUYTELAARS, T.; GOOL, L. Speed-up Robust Features (SURF). **Computer Vision and Image Understanding (CVIU)**, v. 110, p. 346–359, 2008.

BEZERRA, K.; AGUIAR, E. Casamento de padrões em imagens e vídeos usando características de imagens. **XXVI SIBGRAPI – Conference on Graphics, Patterns and Images,** 2013.

GRACIANO, A. **Rastreamento de objetos baseado em reconhecimento estrutural de Padrões**. Dissertação de Mestrado. 2007

JASMINE, J.; DEEPA, S. M. Tracking and Recognition of Objects using SUFR Descriptor and Harris Corner Detection. **International Journal of Current Engineering and Technology**, v. 4, p. 775-778, 2014.

KLIPPENSTEIN, J.; ZHANG, H. Quantitative Evaluation of Feature Extractors on Visual Slam. **Fourth Canadian Conference on Computes and Robot Vision**, p. 157–164, 2007.

LOWE, D. Object recognition from local scale-invariant features. **Proceeding of the International Conference on Computer Vision**, p. 1150–1157, 1999.

LOWE, D. Distinctive Image Features from Scale-Invariant Keypoints. **International Journal of Computer Vision**, p. 91–110, 2004.

LOWE, D. **Demo Software: SIFT Keypoint Detector**. Disponível em: http://www.cs.ubc.ca/~lowe/keypoints. Acesso em 03 de maio de 2019.

MACEDO, M.; AMARAL, E. Sistema de Detecção e Rastreamento de Um Alvo Baseado em Visão Computacional Utilizando um Filtro de Partículas. **Mostra Nacional de Robótica (MNR)**, 2017.

TRUCOO, E.; VERRI, A. Introductory Techniques for 3-D Computer Vison, **Prentice Hall PTR**, 1998.

ZHANG, Z. **SIFT-MATLAB: Extract and match features using SIFT descriptors**. Disponível em: https://github.com/DzReal/SIFT-MATLAB. Acesso em 04 maio de 2019.