UNIVERSIDADE FEDERAL DE SANTA MARIA
COLÉGIO POLITÉCNICO DA UFSM
CURSO DE SISTEMAS PARA INTERNET

Maurissi Albrecht Nunes

# IMPLEMENTAÇÃO DE UM DETECTOR DE MOVIMENTOS À PARTIR DE STREAMINGS DE VIDEO

Santa Maria, RS
2023

Maurissi Albrecht Nunes

# IMPLEMENTAÇÃO DE UM DETECTOR DE MOVIMENTOS À PARTIR DE STREAMINGS DE VIDEO

Trabalho apresentado como requisito parcial para a Conclusão do Curso de Sistemas para a Internet da Universidade Federal de Santa Maria.

Orientador: Prof. Rafael Gressler Milbrasdt

Santa Maria, RS

2023

Nunes, Maurissi.
IMPLEMENTAÇÃO DE UM DETECTOR DE MOVIMENTOS À PARTIR DE STREAMINGS DE VIDEO / Maurissi Nunes.- 2023.

30 p.; 30 cm:
Orientador: Rafael Gressler Milbrasdt
Tese (livre-docência) - Universidade Federal de Santa
Maria, Colégio Politécnico, RS, 2023
1. Visão computacional 2. Remoção de fundo
3. Detecção
de movimento I. Gressler Milbrasdt, Rafael II. Título

**MAURISSI ALBRECHT NUNES**

# IMPLEMENTAÇÃO DE UM DETECTOR DE MOVIMENTOS À PARTIR DE STREAMINGS DE VIDEO

Como requisito parcial para obtenção do
grau de **Tecnólogo em Sistemas para Internet**

**Aprovada em 12 de julho de 2023**

---

**Rafael Gressler Milbrasdt**
**Presidente/Orientador**

---

**Andressa Falcade**
**(UFSM)**

---

**Daniel Lichtnow**
**(UFSM)**

Santa Maria, RS

2023

## AGRADECIMENTOS

Deixo esse espaço para agradecer a todos os profissionais ligados ao curso de Sistemas para Internet. Este documento teve direta ou indiretamente a contribuição de cada um desses profissionais. Em especial aos professores que compõem a coordenação do curso, visto que desempenham um papel além das salas de aula.

Uma das muitas pessoas que merecem um reconhecimento especial é meu orientador Rafael. Ele aceitou no primeiro pedido e sem hesitar a responsabilidade extra como orientador. Acredito que esse trabalho proporcionou uma troca de conhecimentos entre diversas áreas sendo benéfica para os envolvidos.

Por fim um nome que não poderia faltar é dela, Milene. Por ter cedido vários dos seus finais de semana de lazer para que eu pudesse completar essa tarefa. Sem reclamar, muito.

Também fica um agradecimento a você, leitor. Afinal esse trabalho foi feito de certa forma, também, para você. Sendo assim espero que consiga extrair algo valioso e que construa algo genuíno com ele.

## RESUMO

### IMPLEMENTAÇÃO DE UM DETECTOR DE MOVIMENTOS À PARTIR DE STREAMINGS DE VIDEO

AUTOR: Maurissi Albrecht Nunes
ORIENTADOR: Rafael Gressler Milbrasdt

O estudo centrou-se no caso especial de detecção de movimento em um ambiente. Esse ambiente estava sendo registrado por uma câmera fixa. Para aquisição dos dados foi utilizado um streaming de vídeo, captado através de uma câmera com conexão USB. Após a captura dos dados, eles foram decompostos em quadros individuais e tratados como fotografias únicas. A chamada técnica de remoção de fundo foi aplicada a cada uma dessas imagens. Filtros convolucionais de frequência e morfológico também foram utilizados. Eles desempenharam um papel de limpeza e destaque nas regiões-alvo da imagem. Combinados, foi possível destacar a região da imagem original onde ocorreram os movimentos. Sempre nos casos em que a movimentação foi detectada, um evento de texto foi gerado informando o número da fotografia. Foi concluído seu funcionamento e sua possível extensão de aplicabilidade caso seja integrado com outros sistemas. Nesse caso, tanto o armazenamento quanto a transmissão de dados têm o potencial em serem reduzidos. Dessa forma, quando implementado, possibilitará uma economia passiva de recursos. Por fim, foram sugeridos ensaios com outros dispositivos de hardware para melhorar seu desempenho energético.



**Palavras-chave**: Remoção de fundo. Visão computacional. Detecção movimento.

## ABSTRACT

### IMPLEMENTATION OF A MOTION DETECTOR FROM VIDEO STREAMS

AUTHOR: Maurissi Albrecht Nunes
ADVISOR: Rafael Gressler Milbrasdt

The study focused on the special case of detecting movement in an environment. This environment was being recorded by a fixed camera. For data acquisition, a video streaming was used, captured through a camera with USB connection. After capturing the data, they were decomposed into individual frames and treated as single photographs. The so-called background removal technique was applied to each of these images. Frequency and morphological convolutional filters were also used. They played a cleaning and highlighting role in target regions of the image. Combined, it was possible to highlight the region of the original image where the movements occurred. Whenever movement was detected, a text event was generated informing the number of the photograph. Its correct functioning has been completed. The case of its possible extension of applicability if integrated with other systems was also mentioned. In this case, both storage and transmission of data have the potential to be reduced. Thus, when implemented, a passive economy of resources will be possible. Finally, tests with other hardware devices were suggested to improve their energy performance.



**Keywords**: Background Removal. Computer Vision. Movement Detection

## LISTA DE FIGURAS

**SUMÁRIO**

# 1 INTRODUÇÃO

Visando automatizar o processo de informar se houve movimentação em um ambiente e contribuindo passivamente na redução de dados armazenados ou trafegados. Esse trabalho irá informar se houve movimentação em um determinado ambiente. Para isso serão utilizadas técnicas de visão computacional aplicadas em um arquivo de vídeo. Esse arquivo será gerado à partir de uma câmera com conexão do tipo USB[1], posicionada em um local fixo no ambiente.

Essas técnicas e conceitos estão contidos nas obras de Meneses (2012), Zhang (2000), Gonzales (2005) e Colaço (2018). O resultado obtido é uma imagem binaria que destacará os trechos do ambiente que sofreram movimentações. Será gerado um evento informando caso essas movimentações excedam um determinado limite. Sendo assim o trabalho não irá procurar um objeto ou forma em específico, mas captar todas as alterações em um ambiente.

## 1.1 CONTEXTUALIZAÇÃO DO PROBLEMA

Uma empresa de segurança possui uma câmera fixada em cada um dos 16 ambientes monitorados para um cliente. Contratualmente as câmeras devem ficar ligadas 24 horas todos os dias registrando os diversos ambientes. Esses registros de gravações devem ser mantidos por até 5 anos para posteriores averiguações. Outro ponto contratual é informar imediatamente ao cliente sempre que haja movimento nos ambientes em determinados períodos. Esses períodos são compreendidos entre as 19 horas do dia corrente até as 7 horas do dia posterior. Os custos de armazenamento e eventual tráfego de dados são cobrados proporcionalmente a sua utilização.

No caso dessa empresa hipotética, implementar um recurso de software que gere um sinal automático quando há movimentação em cena poderá economizar um cargo de operador. Caso este sinal também seja responsável por habilitar a gravação das câmeras somente quando haja movimento em cena, os custos de armazenamento de dados serão reduzidos próximos à metade. Isso porque o cliente informou um grande período em que não deve haver movimentação.

Nesse caso hipotético o sistema estaria atuando como um sensor de presença e passivamente reduzindo os custos em armazenamento. Caso esse cenário fosse real todos esses

[1] USB abreviação de Universal Serial Bus.

benefícios estariam sendo implantados com o mínimo de custo possível. Pois a infraestrutura existente possivelmente já possuiria câmeras, suportes, fiações e computador.

### 1.1.1 Problema

Esse trabalho aborda o emprego de diversas técnicas computacionais sobre um arquivo de vídeo. O grande desafio é tornar o sistema rápido o suficiente para adquirir as imagens, processá-las e informar o resultado em tempo real. Todas essas atividades sendo executadas em um computador comum. Nesse contexto tempo real seria o processamento de pelo menos 30 imagens por segundo, sem descarte ou armazená-las para processar posteriormente. Entende-se por computador comum os computadores que: No mínimo sejam compatíveis com DirectX 11 ou equivalente; possuam 8 gigabytes de memória principal; contenham pelo menos 120 gigabytes de memória secundaria; tenham processador com velocidade de 2 giga-hertz.

## 1.2 OBJETIVOS

### 1.2.1 Objetivo Geral

Implementar uma ferramenta que faça a detecção de movimentos em vídeo através de mudanças cuja a área total alterada seja superior a 1% da imagem capturada, com um processamento mínimo esperado de 30 imagens por segundo.

### 1.2.2 Objetivos Específicos

Para atingir o objetivo geral serão necessários:

- Conhecer o processo de formação de uma imagem digital e sua constituição.
- Destacar digitalmente pontos de interesse na imagem utilizando filtros.
- Pesquisar técnicas para segmentar o cenário isolando-o dos objetos em movimento.
- Otimizar o processamento de imagens.
- Testar, validar e discutir os resultados e sua aplicabilidade real.

## 1.3 JUSTIFICATIVA

Partindo da contextualização do problema, o trabalho se justifica por validar técnicas de visão computacional. Essas técnicas podem ser implementadas à um sistema online concedendo uma funcionalidade de rotular arquivos de vídeo com índices e períodos de movimentação. Esses períodos tem a finalidade de auxiliar o processo de operadores humanos. Nesse contexto sua aplicação vai além de agregar uma nova funcionalidade. Também possui a capacidade passiva de reduzir o volume no trafego de dados e no seu armazenamento.

## 1.4 ORGANIZAÇÃO DO TRABALHO

Ao todo o trabalho possui seis capítulos sendo o primeiro capitulo responsável pela introdução, contextualização e objetivos do trabalho. Logo após um capítulo dedicado a fundamentação teórica com formação de imagens, explicação de filtros e remoção de fundo. A parte teórica é encerrada com o capitulo três. Nele são apresentadas técnicas e dispositivos reais que fazem uso da remoção de fundo para detecção de movimento.

O capitulo quatro é dedicado inteiramente ao desenvolvimento do projeto sendo o mais extenso e complementado pelo capitulo cinco onde estão mostrados os resultados obtidos pelo ensaio. O trabalho é finalizado pelo capitulo seis onde consta um questionamento quanto à implementação do sistema em um ambiente real e suas considerações finais.

# 2 FUNDAMENTAÇÃO TEORICA

Neste capitulo constam alguns conceitos sobre a formação das imagens e algumas técnicas que podem ser utilizadas sobre essas imagens já formadas. As técnicas e conceitos são extraídas de obras de Pedrini (2007), Gonzales (2012) e Zhang (2005). Outras utilizam conceitos dos trabalhos de Kaehler (2017) e Vacavant (2013).

## 2.1 IMAGENS

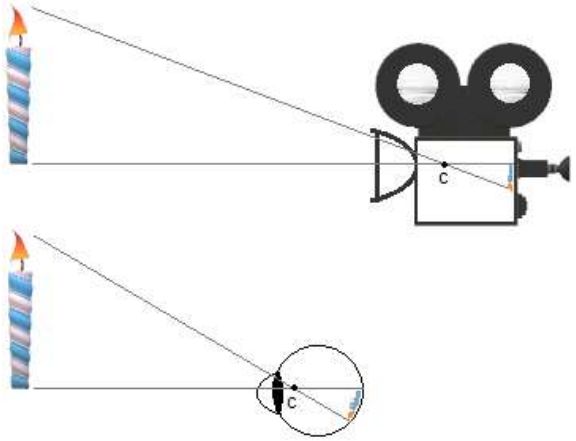
Conforme Pedrini (2007), uma imagem digital pode ser definida como uma função de intensidade luminosa $f(x, y) = i$. Nela o par ordenado $(x, y)$ corresponde respectivamente a linha e coluna do pixel da imagem e $i$ o valor da intensidade naquele ponto. A definição de pixel está nesse documento no item 2.1.1.

Um arquivo de vídeo é composto por várias imagens aninhadas sequencialmente. Dessa forma o cérebro humano interpreta essa sequência de imagens como uma ação em um tempo continuo, Gonzales (2012). Essas imagens em individual são chamadas de *frame* sendo a sequência regida pelo instante de tempo do registro de cada frame. A diferença de tempo entre dois frames compreende o período do vídeo e seu inverso a frequência. Uma unidade de medida denominada frames por segundo (fps) é utilizada para denotar essa frequência do vídeo.

### 2.1.1 Formação de Imagens

Gonzales (2012) compara a visão do olho humano com uma câmera. Ele cita que basicamente os humanos percebem as imagens através de dois tipos principais de células sensíveis a luz denominadas de cones e bastonetes. Destes, os cones ainda são especializados em diferentes comprimentos de onda. Assim são especializados em detectar o nível de intensidade de determinadas cores e estão densamente agrupados em uma região denominada Fóvea que não por acaso é o foco do Cristalino. Essa comparação está representada na Figura 1.

Figura 1 - Olho humano e dispositivo digital.



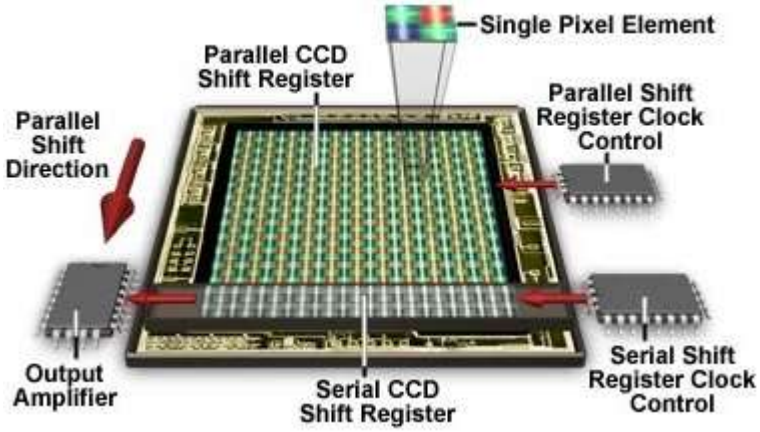

Fonte: Próprio autor.

A Figura 1 ilustra um modelo simplificado do olho humano e de uma câmera digital. Nele o Cristalino está para a lente da câmera, assim como a Fóvea do olho está para a região focal da lente.

### 2.1.2 Imagens Digitais

Imagem digital é constituída de um arquivo que contém a digitalização da região focal de uma câmera, a Figura 2 ilustra com mais detalhes essa região. Conforme analogia de Gonzales (2012) essa região representa a Fóvea contendo os cones dos olhos humanos.

Figura 2 - Matriz de elementos sensores.




Fonte: (HAMAMATSU, 2023).

Ainda seguindo a analogia do autor, esses elementos sensores são minúsculos e estão agrupados na ordem de milhões por centímetro quadrado. Na figura acima está ilustrado um tipo comercial de matriz de elementos sensores. A função básica dessa matriz é a conversão do fóton que atinge cada elemento sensor em uma carga elétrica. Após essa conversão a carga recebe uma amplificação padronizada para ser enviada à um conversor A/D[2]. Esse conversor A/D quantificará a carga analógica em níveis digitais. Portanto essa medida digital está atrelada a intensidade dos fótons e a sensibilidade do elemento que os recebeu. Por isso é importante uma distribuição uniformemente variada dos elementos sensores ao longo da matriz.

Normalmente os elementos sensores são reunidos fisicamente em grupos que respondam à três tipos de cores específicas e para cada cor especifica há uma variação de 256 intensidades diferentes. Esse grupo é denominado de pixel e considerado indivisível, seu ponto físico no espaço é a média aritmética das posições dos elementos do grupo no espaço. Cada cor específica que o pixel seja capaz de mensurar é chamada de canal, sendo assim as câmeras que possuem saída de vídeo colorida devem, ao menos, possuir saída de três canais.

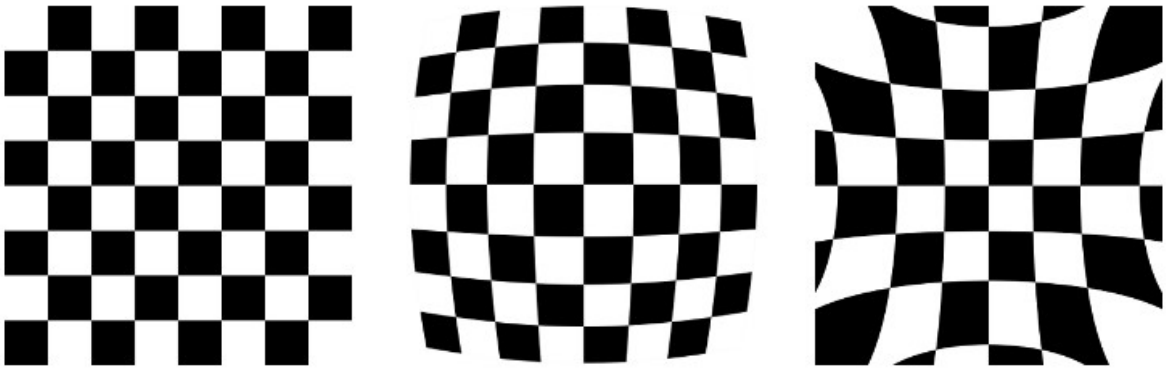
Agora em um domínio digital, uma imagem de três canais é constituída por pixels também de três canais. Cada canal necessita de no mínimo 1 byte para descrever todos os 256 níveis diferentes. Dessa forma é possível armazenar digitalmente um pixel com apenas 3 bytes, no mínimo. Assim com diferentes combinações de intensidades é possível captar mais de 16 milhões de cores distintas. Se corretamente combinadas e mensuradas, essas cores superam a distinção dos olhos humanos contemplando todo o espectro visível, Gonzales (2012).

## 2.2 CORREÇÃO DE LENTES

Uma imagem ao atravessar por uma lente sofre algumas anormalidades. A Figura 3 mostra duas delas denominada de distorção do tipo radial e tangencia. Conforme Zhang (2005) essas duas são as anormalidades que mais distorcem a imagem. Basicamente dependem da posição do pixel na matriz de elementos sensores com os ângulos de abertura das lentes. O trabalho de Zhang (2005) mostra como minimizar essas distorções. O termo “minimizar” foi empregado porque qualquer imperfeição na construção da lente implica em anomalias únicas. Então quanto mais bem fabricada for uma lente melhor ela será corrigida pelo método de Zhang.

[2] A/D refere-se a dispositivos que realizam uma conversão de sinal analógico para digital.

Figura 3 - Imagem Normal, Radial e Tangencial



Fonte: (OPENCV, 2023).

Seu processo consiste em deslocar a origem da imagem para o seu centro e alterar as coordenadas de cada pixel do sistema retangular para o sistema polar. Agora os pixels são referenciados por um raio e dois ângulos ao invés de duas coordenadas cartesianas. O próximo passo é a normalização do raio para o range de -1 a 1 unidade, dessa forma o resultado independe da resolução da imagem. As correções são aplicadas recalculando o valor de cada raio para cada ângulo utilizando 5 parâmetros contidos em uma matriz de correção. Nesse realojamento de pixels podem surgir falhas ou sobreposições. Então utilizando a média das cores ponderada pela localização atual do pixel, eventuais espaços vazios são preenchidos ou aglutinamentos de pixels são substituídos com novas cores.

## 2.3 FILTROS

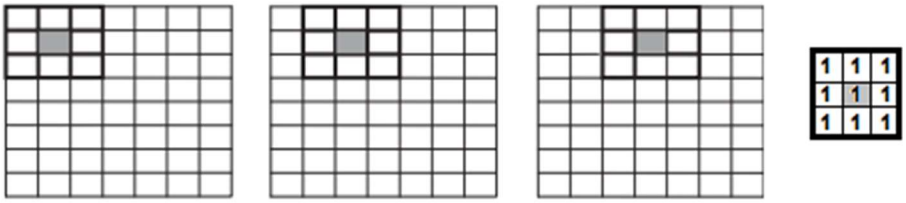
Uma imagem digital pode apresentar diversos tipos de ruídos. Meneses (2012) mostra várias técnicas para tratar ou mitigar esses ruídos através de filtros. Em seu trabalho eles correspondem aos capítulos 7 e 9. É de interesse desse trabalho os filtros espaciais do tipo passa-baixa com janela de convolução e filtro morfológico.

Esse item tem como base Meneses (2012, p.168) que afirma, “em qualquer imagem sempre é observado uma forte interdependência da vizinhança espacial dos valores dos pixels, porque os alvos na natureza tendem a mostrar uma homogeneidade dentro de certos espaços.” Os próximos subitens explorarão essa interdependência, porém cada um com um objetivo divergente.

### 2.3.1 Filtro Espacial Convolucional

Conforme mostra Meneses (2012), os filtros espaciais basicamente recalculam o valor da cor de cada pixel, o filtro utiliza como base as cores dos pixels adjacentes ou vizinhos a ele. Considera-se como adjacente todos os pixels dentro de uma janela de n x n pixels, centrada no alvo. O cálculo para o novo valor do pixel alvo consiste em multiplicar cada respectivo pixel por um respectivo valor contido em uma janela de mesmo tamanho. Logo após somar os respectivos valores. A Figura 4 representa um filtro desse tipo.

Figura 4 - Convolução dos pixels (1,1), (2,1), (3.1) e janela 3x3.



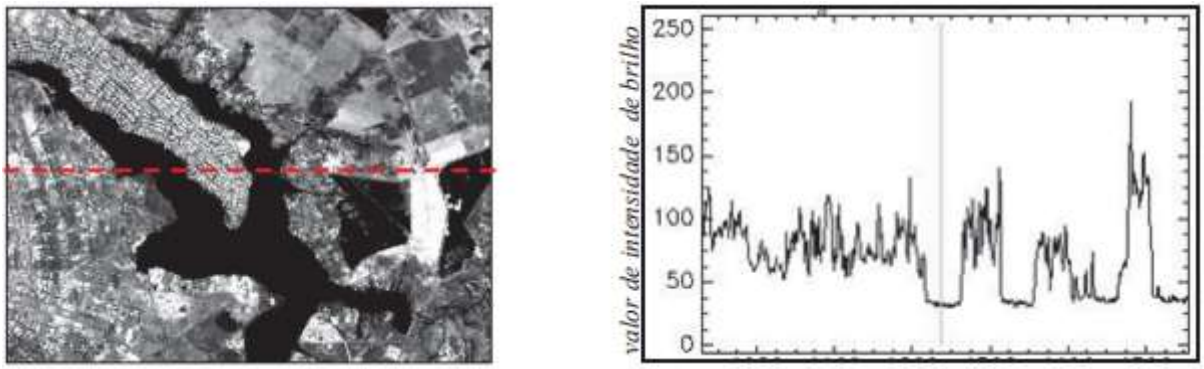

Fonte: Adaptado (MENESES, 2012,p.171).

A figura acima ilustra uma janela de 3 x 3 pixels sendo transladada através dos alvos. A ação de transladar essa janela por todos os pixels da imagem é denominada de convolução.

### 2.3.2 Frequência

Essa palavra aplicada a imagens faz referência a variação de intensidade de cores. Essa variação deve ser entre pixels vizinhos e em uma única direção. Quanto maior for o número de variações em uma região maior será sua frequência e isso indica que essa região possui mais detalhes. Quanto maior for a diferença dessas variações maior será a amplitude dessa frequência. A Figura 5 mostra uma imagem aérea e um gráfico contendo a intensidade luminosa em uma determinada direção na imagem.

Figura 5 - Variação da intensidade da cor.



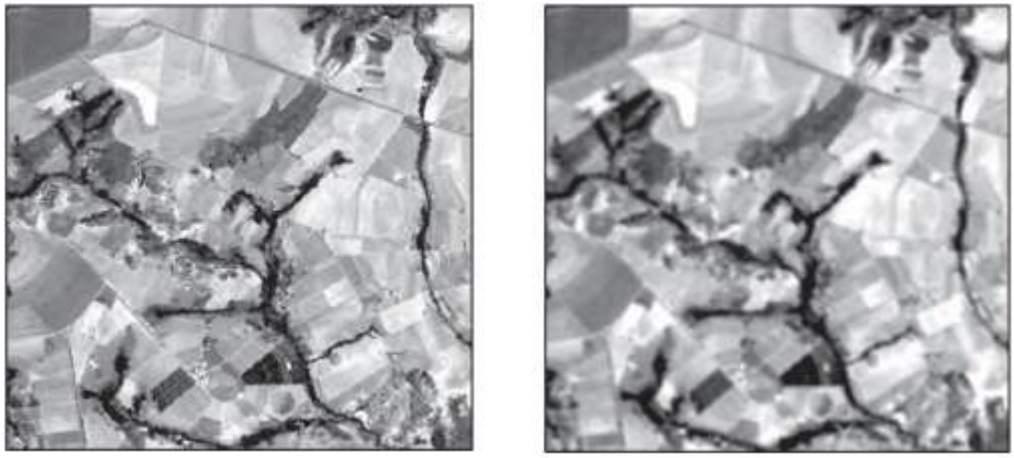

Fonte: Adaptado (MENESES, 2012, p.169).

Pelo gráfico dá imagem acima é possível acompanhar as maiores amplitudes onde há uma transição de terrenos. Durante as regiões dos lagos é possível observar uma baixa frequência, o que indica poucos detalhes ou objetos nessa região. Ao passar pelas regiões onde possuem construções é possível observar um salto nas frequências e suas amplitudes, isso indica mais detalhes e objetos distintos.

### 2.3.3 Filtro Passa-Baixa De Média

Meneses (2012) cita algumas regras para o tamanho e os valores contidos na janela de convolução, esses parâmetros variam de acordo com o tipo de resposta ou característica procurada. Esse trabalho visa a detecção do movimento de objetos e não suas características, para isso será aplicado um filtro que permita um realce das baixas frequências e uma atenuação das altas. Filtros desse tipo tendem a suavizar o contraste da imagem reduzindo erros pontuais e detalhes. A Figura 6 ilustra o resultado final desse tipo de filtro, a imagem à direita é o resultado da aplicação de um filtro passa-baixa de média com janela igual a 5 x 5.

Figura 6 – Filtro passa-baixa de média.



Fonte: (MENESES, 2012, p.175).

Matematicamente esse filtro é aplicado utilizando a formula $g(i,j) = \frac{1}{MN}\sum_{m=1}^{M}\sum_{n=1}^{N} f(m,n)$ , onde m e n correspondem ao tamanho da janela de convolução para o pixel alvo nas coordenadas $(i,j)$. Os valores dessa janela de convolução devem ser positivos e sua soma nunca ultrapassa o produto $MN$. Além disso essa janela deve ser ímpar.

Figura 7 - Máscara 5 x 5 preenchida.

| | | | | |
|---|---|---|---|---|
| +1 | +1 | +1 | +1 | +1 |
| +1 | +1 | +1 | +1 | +1 |
| +1 | +1 | +1 | +1 | +1 |
| +1 | +1 | +1 | +1 | +1 |
| +1 | +1 | +1 | +1 | +1 |

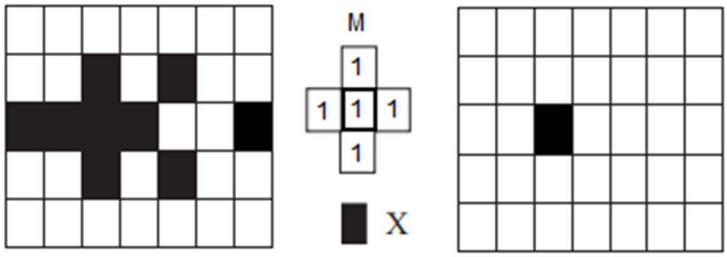
Fonte: Próprio autor.

A Figura 7 contém uma representação da máscara aplicada na Figura 6 e atende todos os requisitos especificados anteriormente.

### 2.3.4 Filtro Morfológico

Filtros morfológicos alteram a geometria do objeto e utilizam conceitos da teoria de conjuntos. Por essa natureza costumam ser aplicados em imagens binárias e Meneses (2012) mostra vários tipos, em especial o tipo erosão que possui a finalidade de excluir formas indesejadas. Para isso é definido um conjunto interceptador X e um elemento estruturante M que será representado no formato de uma janela de convolução. A Figura 8 possui uma imagem à esquerda onde é percorrida pelo estruturante M a passos de X. Sempre que o resultado for igual ao estruturante será acrescido X no resultado.

Figura 8 – Filtro Morfológico.




Fonte: Adaptado (MENESES, 2012, p. 182)

A imagem à direita da figura acima representa o resultado com apenas um X na posição que continha a forma desejada na imagem original à esquerda.

## 2.4 REMOÇÃO DE FUNDO

No capitulo quinze de Kaehler (2017) é feita uma detecção de movimentos em fotos a partir de uma câmera estática. Antes disso um trabalho de Vacavant (2013) executou um trabalho semelhante. Posteriormente esse trabalho foi parcialmente descrito por Domenico Daniele Blois[3] nos documentos oficiais do framework OpenCV[4]. A constante dos 3 trabalhos é a técnica de realizar uma subtração do frame corrente com uma imagem estática. Essa imagem estática contem somente os elementos do cenário e é chamada de fundo. Caso a comparação resulte em zero será atribuído uma cor preta (ausência de cor ou luz). Caso contrário será atribuído uma cor específica. Essa cor representa o movimento na cena, e a ausência de cor

[3] Fonte: https://docs.opencv.org/4.x/d1/dc5/tutorial_background_subtraction.html. Acessado em junho 2023
[4] OpenCV é um framework aberto destinado ao processamento de imagens digitais.

representa o fundo. A lógica consiste em adquirir o mapa de pixels com a cena estática, ou seja, sem movimento. Após isso todos os respetivos pixels dos futuros frames serão comparados um a um. Pelo fato da premissa de utilizar uma câmera fixa, as variações só poderão ser do objeto em movimento.

Em termos matemáticos, dados os conjuntos $L = \{x \in \mathbb{N} / x \geq 0, x < n\}$ representando o comprimento da imagem. $W = \{x \in \mathbb{N} / x \geq 0, x < n\}$ representando a altura da imagem. O mapa de fundo será constituído de um plano ordenado $\mathbb{N}^2$ denominado $\Phi = \{p(x,y) \in \mathbb{R} \forall x \in L, y \in W\}$. Todos os futuros frames capturados $A_i$ com $i \epsilon \mathbb{N}$ serão semelhantes ao fundo, $A \approx \Phi$. Se comparados pixel a pixel de forma $r_i(x,y) = \|a_i(x,y) - p(x,y)\|$. A função módula foi inserida porque nesse contexto a cor representa uma intensidade luminosa e não existe intensidade luminosa negativa. Dessa forma qualquer ponto $r_i \in B / r_i > 0$ será um pixel com alteração. A intensidade dessa alteração é o próprio valor de $r_i$ e significa o quão próximo a cor do objeto em movimento é do fundo naquele ponto. Logo quanto maior o contraste entre objeto e fundo melhor será a detecção.

### 2.4.1 Reconstrução De Fundo

Caso o plano de referência Φ do item 2.4 seja sempre estático, novas alterações serão sempre detectadas como movimento na imagem. Então uma função para reconstruir o plano Φ deve ser chamada em períodos regulares.

A técnica adotada por Kaehler (2017) envolve calcular a média de variação, desvio padrão e variação local. Esses dados são calculados por pixel com a finalidade de constituir uma taxa de variação única para o pixel analisado. Essa taxa é então utilizada para ajustar a imagem estática de fundo, sendo seu objetivo de variar o fundo satisfatoriamente para evitar falsos positivos incorporando novos objetos. Ao mesmo tempo evitar falsos negativos removendo objetos que se desvincularam do cenário.

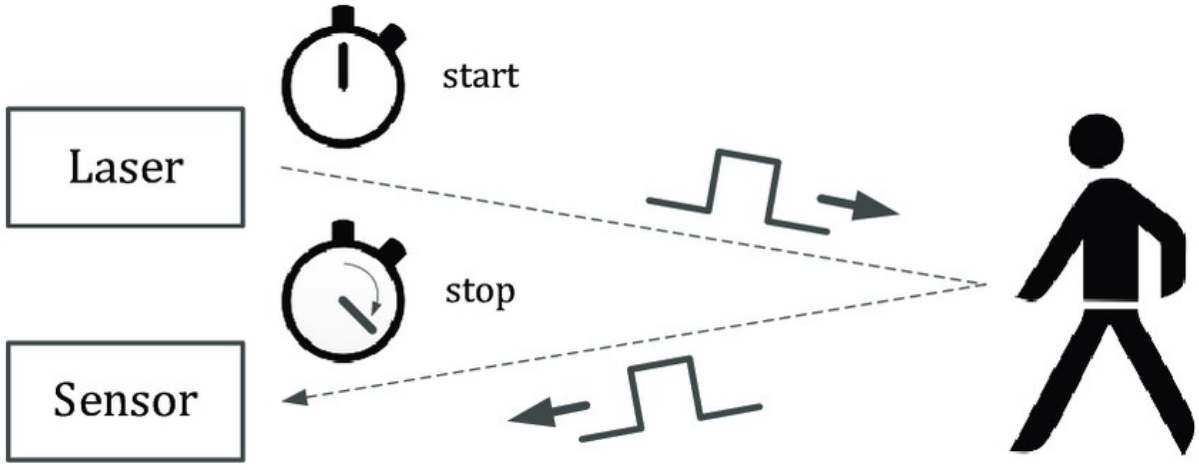
# 3 TECNICAS E DISPOSITIVOS

Nesse capitulo serão abordados técnicas e dispositivos que estão sendo empregados para detectar movimentação e segmentação do cenário em imagens.

## 3.1 TEMPO DE VOO

Utilizando de princípios básicos de física e análogo ao sonar e eco localização. Essa técnica denominada de tempo de voo ou Time Of Flight (TOF) utiliza referências de tempo para medir distâncias. Beer (2018) mostra em detalhes o funcionamento dessa técnica em seu artigo, para esse documento basta saber o ilustrado pela Figura 9. Ela tem início com um flash infravermelho de laser com comprimento de onda maior que 700nm e simultaneamente é iniciado um cronômetro. O feixe de luz do flash viaja até o objeto e o ilumina, ao iluminar esse objeto um feixe é refletido de volta ao emissor que ao detectar esse feixe refletido pausa o cronômetro. Para obter a distância basta dividir o tempo pela metade e multiplicar pela velocidade da luz.

Figura 9 - Técnica TOF.

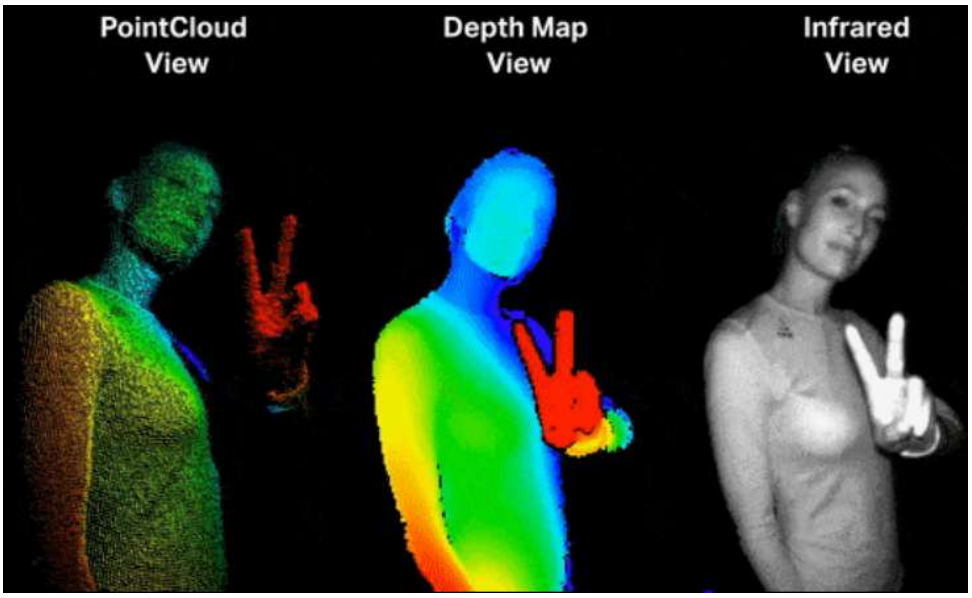



Fonte: (BEER, 2018).

Pelo fato de ser possível trabalhar com luz, empresas como ST, Sony e Toppan se destacam nesse nicho. Conforme informativo em seus sites, elas basicamente substituíram os elementos sensores de câmeras comuns para responderem especificamente ao comprimento de onda do laser. Dessa forma ao detectarem o laser elas armazenam uma referência de tempo que posteriormente será utilizada para calcular a distância do pixel até o alvo. Logo a cor do pixel

será tratada como um comprimento no espaço e seu resultado é um arquivo conhecido como nuvem de pontos. Com essa nuvem é possível criar um mapa de profundidade em que cada cor represente uma distância paralela ao eixo Z da câmera no espaço. A Figura 10 mostra uma nuvem de pontos à esquerda e o mapa de profundidade gerado a partir dessa nuvem. Mais à direita uma foto do mesmo ambiente com uma câmera infravermelha.

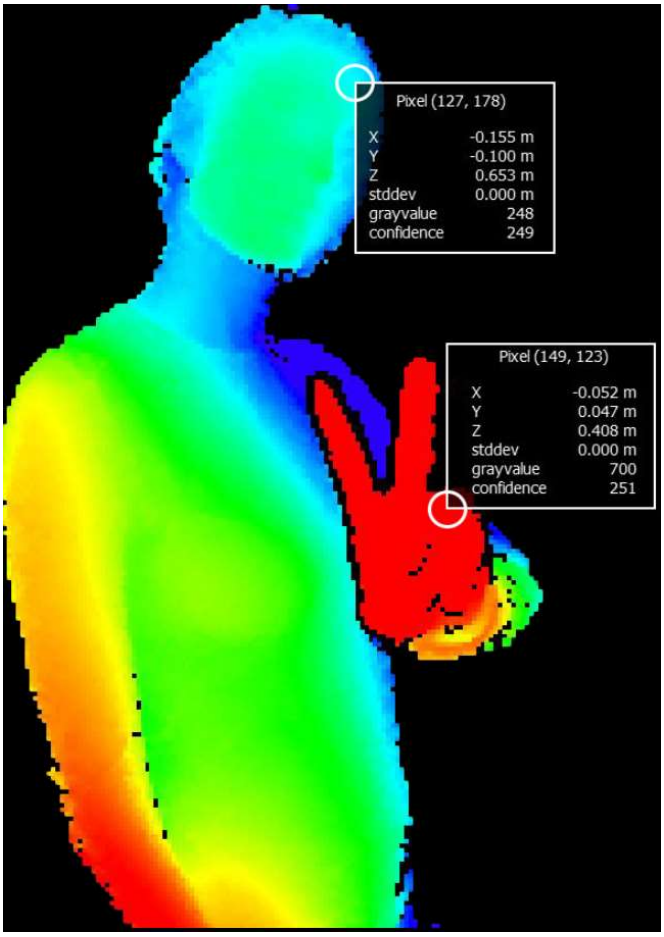
Figura 10 - Nuvem de pontos - Mapa de profundidade - Imagem infravermelha.




Fonte: (PMD, 2023).

Para detectar movimentos com imagens desse tipo, basta realizar a subtração das cores. Uma vez que elas representam as posições dos objetos no espaço é possível detectar velocidade, estimativa de trajetória e aceleração. A Figura 11 mostra um objeto escaneado em três dimensões a partir de um mapa de profundidade. Nela fica claro que as cores, na verdade, representam posições no espaço atreladas a um grau de confiabilidade.

Figura 11 - Objetos gerados a partir de mapas de profundidade.




Fonte: (PMD, 2023).

Hoje sem dúvida é o segmento mais avançado por possuir a menor margem de erro e maior robustez. É amplamente utilizado em carros autônomos na forma de dispositivos LIDAR (Light Detection and Ranging) de longo alcance. Alguns smartphones considerados de última geração também estão começando a utilizar essa tecnologia, um exemplo é o lançamento da Motorola[5]. Seu propósito é justamente identificar elementos do cenário real e misturar com elementos virtuais. A Figura 12 contém um exemplo de dispositivo que utiliza TOF.

[5] Fonte: https://www.motorola.com.br/smartphone-moto-g100/p?idsku=1082. Acessado em junho 2023

Figura 12 – Equipamento de desenvolvimento Pico Flexx2.


Fonte: (ALIEXPRESS, 2023, PICO FLEXX2).

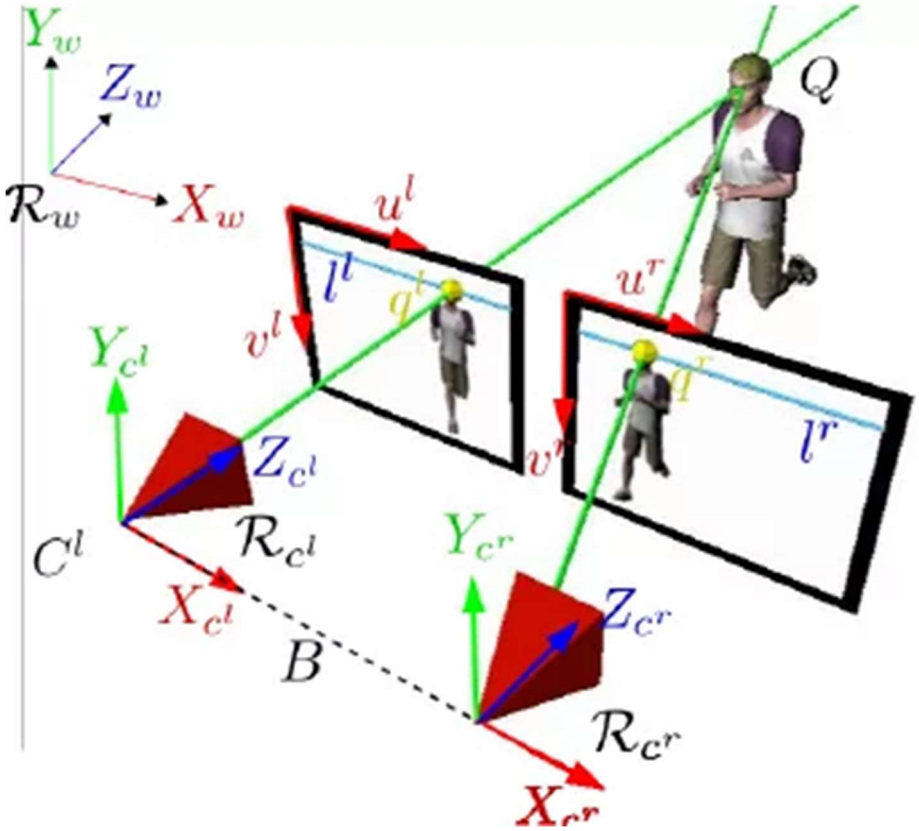
Por trabalhar com grandezas de tempo muito pequenas esses dispositivos se tornam caros em comparativo a outras técnicas. As Figuras 10 e 11 foram retiradas do site da empresa desenvolvedora PMD. Ela vende equipamentos de desenvolvimento e conforme a figura acima superam os 5 mil reais. O hardware possui um alcance modesto de aproximadamente 4 metros com imagens na resolução de 224 x 172 pixels.

## 3.2 VISÃO ESTÉREO

Esse tipo de técnica é baseada na visão dos grandes predadores da natureza, o que inclui os humanos. Ela consiste no emprego de imagens capturadas em diferentes ângulos para estimar a distância dos objetos, Kaehler (2017) aborda completamente esse tema em seu capitulo 19. Conforme está ilustrado na Figura 13, basicamente com um par de imagens transladadas, faz-se uma correlação de pontos das duas imagens para estimar a distância dos objetos. O nome da técnica amplamente utilizada na computação para fazer essa estimativa é denominada Geometria Epipolar. Conforme Meneses (2012) essa técnica é utilizada para reconhecimento topográfico aéreo. Seu erro depende da relação entre a distância real do objeto até as câmeras e a distância e ângulo entre câmeras ou poses. Outro fator crucial para essa técnica é a textura dos objetos. Dificulta ou fica impossível a correlação de pontos em objetos com texturas

homogêneas, objetos com altíssimo ou baixíssimo grau de refletância também são difíceis de medir.

Figura 13 - Visão Estéreo e Geometria Epipolar.




Fonte: (OPENCV, 2023).

Uma ampla gama de dispositivos no mercado utiliza esse tipo de técnica. Entre os mais comuns estavam os dispositivos Kinect[6] para consoles de videogame Xbox. Esses dispositivos foram descontinuados, mas deram origem a linha de câmeras RealSense da fabricante Intel. Para os dispositivos RealSense que utilizam visão estéreo a imagem resultante é uma nuvem de pontos. Essa nuvem possui a capacidade de entregar uma textura colorida referente ao objeto real, conforme fabricante Intel a imagem possui 6 canais sendo 3 canais de cores e 3 canais de posição no espaço (X,Y,Z). Os arquivos são gigantescos e exigem muito recurso computacional após aquisição das imagens. Esse tipo de dispositivo geralmente conta com duas câmeras preto e branco para estimar a profundidade, elas trabalham em conjunto com um projetor de pontos infravermelho para melhorar a acurácia em objetos de baixa textura. Uma câmera colorida

[6] Fonte: https://learn.microsoft.com/pt-br/windows/apps/design/devices/kinect-for-windows. Acessado em junho 2023.

completa o dispositivo para aquisição de imagens com textura em 3 canais coloridos. A Figura 14 mostra uma nuvem de pontos extraída com esse tipo de dispositivo. É perceptível o altíssimo nível de ruido se comparado com os dispositivos que utilizam tempo de voo. Por possuir baixa fidelidade esses dispositivos são mais utilizados na indústria de robótica para evitar colisões e estimar distâncias até objetos a serem manipulados.

Figura 14 - Brinquedo de pelúcia captado por câmera de visão estéreo

Fonte: (INTEL, 2023).

A Figura 15 representa um equipamento desse tipo, da fabricante Intel. Nesses casos para verificar movimentações em cena basta adotar técnicas semelhantes ao tópico anterior, fazendo uso de mapas de profundidade e objetos isolados.

Figura 15 - Câmera da linha RealSense com visão estéreo.


Fonte: (ALIEXPRESS, 2023, REALSENSE D435).

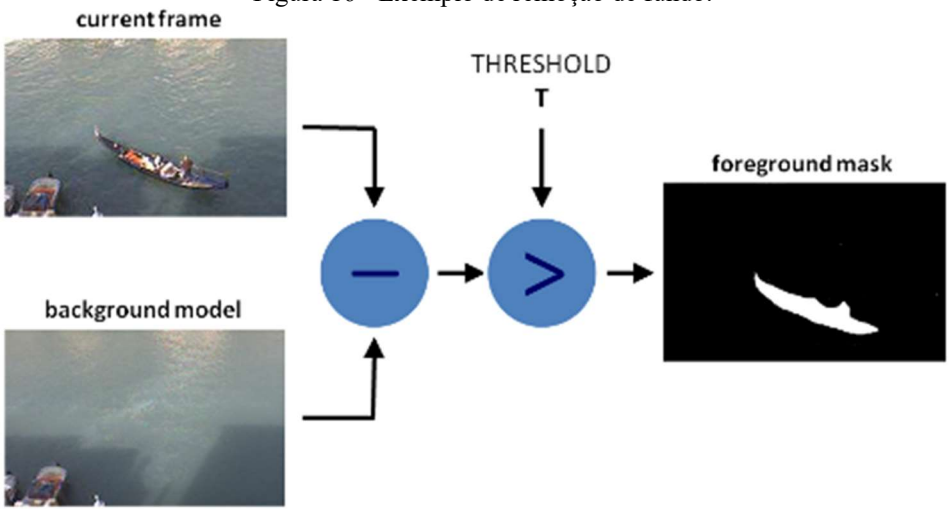
Embora possuam as desvantagens citadas até agora. O valor de dispositivos que utilizam essa técnica costuma ser menos que a metade de um dispositivo do item anterior. A Figura 15 foi retirada no mesmo site de vendas da Figura 12, no mesmo dia e evidencia o preço.

## 3.3 FILTRO POR COR

Em alguns casos é possível segmentar o fundo dos demais objetos em movimento utilizando somente a cor. Meneses (2012) realizou um experimento desse tipo com imagens aéreas, porém para aplicar esse método é necessário conhecer as cores dos objetos que estarão em movimento. Um método mais genérico é implementado por Kaehler (2017) em seu capitulo 14, a remoção de fundo. Nele o autor traz uma coletânea de métodos e cita autores que o utilizaram antes dele, seu funcionamento já foi descrito no tópico 2.4 deste documento.

A Figura 16 foi retirada da documentação oficial do framework OpenCV e ilustra o resultado desejado pela aplicação desse método através de um equipamento simples ilustrado pela Figura 17.

Figura 16 - Exemplo de remoção de fundo.




Fonte: (OPENCV, 2023, BACKGROUND SUBTRACTION).

Figura 17 - Câmera colorida USB.


Fonte: (ALIEXPRESS, 2023, WEBCAM FULL HD).

Sua utilização requer uma câmera colorida, em termos de valores essa técnica é a mais acessível podendo custar mais de 100 vezes menos que as demais. Para comparativo a Figura 17 contém o preço de um equipamento desse tipo no mesmo site de vendas e no mesmo dia das Figuras 12 e 15.

# 4 DESENVOLVIMENTO

Esse capitulo contém a lógica e o desenvolvimento da aplicação sendo fundamentadas em filtro por cores. Essa decisão é baseada por questões de custo e acessibilidade aos equipamentos analisados do capítulo anterior.

## 4.1 FRAMEWORK UTILIZADO

Para o desenvolvimento da aplicação será utilizada a plataforma de Unity, ela foi escolhida por já possuir nativamente acesso aos periféricos através de interfaces de alto nível. Com isso várias funções de acesso à câmera já serão abstraídas, outro ponto facilitado será a programação em paralelo na placa de vídeo dedicada. Como seu motor gráfico está nativamente integrado com uma grande variedade de placas de vídeo, o processamento do vídeo não será executado pela CPU[7] e sim pela GPU[8].

Conforme o site oficial[9] da desenvolvedora Unity. Ela teve sua origem centrada em um motor gráfico para jogos da empresa Unity Technologies. Posteriormente evolui para soluções que contemplam desde desenvolvimento web, arquitetos da construção civil, desenvolvimento automotivo, inteligência artificial e realidade aumentada. Hoje grandes empresas utilizam a solução Unity Enterprise, ela representa uma plataforma “fim a fim” de desenvolvimento eletrônico, simulação e integração com modelos de objetos em 3D. A Unity vem buscando divulgar sua marca com licenças e programas educacionais gratuitos como o caso da licença Unity Community. Esse trabalho utiliza uma licença gratuita desse tipo.

A linguagem utilizada pela Unity é uma versão do C#, essa versão é executada dentro de containers compilados pela própria plataforma. Dessa forma o usuário pode utilizar um simples editor de textos ou uma IDE[10] de sua preferência para gerar os códigos. Posteriormente tudo será compilado e otimizado pela plataforma Unity. Para realizar cálculos diretamente na GPU deve ser utilizada a linguagem HLSL. A linguagem HLSL é estruturada, baseada no C e surgiu juntamente com o DirectX9[11]. Ela foi projetada para performance em programação paralela.

---

[7] CPU - Central Processing Unit, representa a unidade central de processamento do computador.
[8] GPU - Graphic Processing Unit, representa a unidade responsável pelo processamento gráfico.
[9] Endereço https://docs.unity3d.com/ (acessado em 18 de junho de 2023)
[10] Integrated Development Environment, representa um ambiente de desenvolvimento de recursos digitais.
[11] É uma coleção de bibliotecas desenvolvidas inicialmente para padronizar requisitos e programações para jogos digitais.

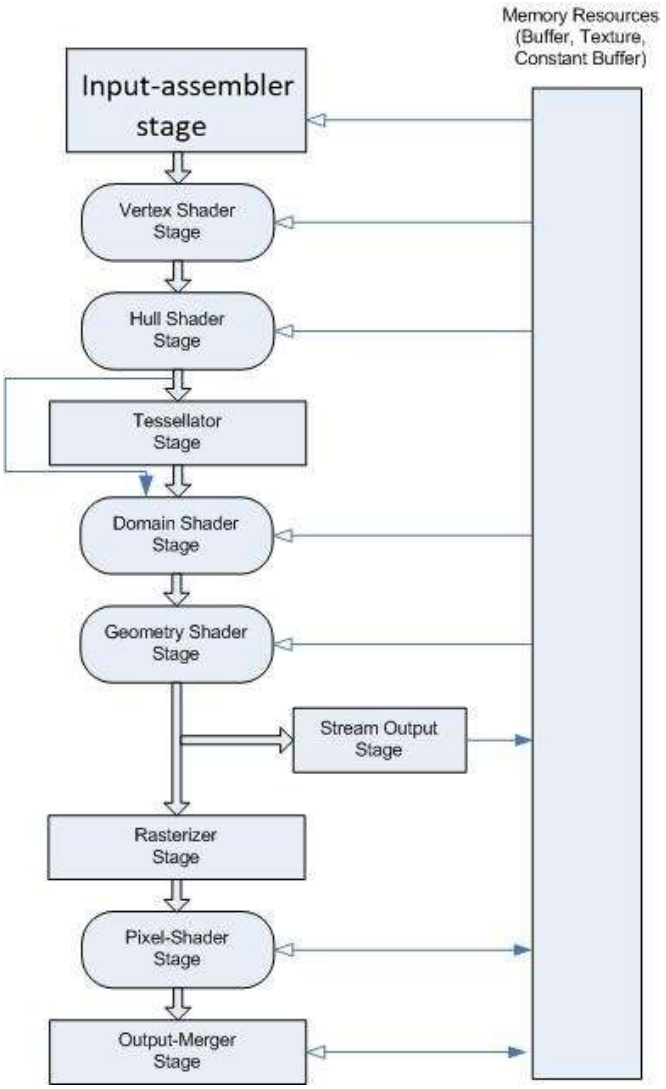
## 4.2 PROGRAMAÇÃO PARALELA

Foi utilizado programação em paralelo porque há muitos dados a serem processados e em estágios bem específicos. Dentro desses estágios os dados não requerem uma ordenação de processamento específica. Além disso foi escolhido a programação paralela dentro da placa dedicada de vídeo pelo baixo número de instruções dentro de cada estágio. Essa escolha não só melhora a performance como libera tempo de processamento da CPU. A garantia de melhora em performance reside no pré-requisito de compatibilidade da placa de vídeo com o DirectX11[12]. Isso garante à aplicação uma quantia de até 1024 threads para o processamento desses estágios, cada thread com um núcleo físico dedicado funcionando na mesma frequência da placa de vídeo. Esses estágios bem definidos se concretizam dentro da Unity como *Comput Shaders*.

Os dados à serem processados em uma GPU seguem um fluxo constante denominado *pipeline*. Primeiro a CPU carrega os dados para uma região de transição com a GPU, logo após carrega as instruções em assembler para cada grupo de etapas. Elas podem já estarem compiladas ou serem compiladas em runtime para aproveitamento de recursos. Depois a CPU indica que o *pipeline* pode ser iniciado e aguarda a sua conclusão para ler o resultado na memória de transição. O *pipeline* dentro da GPU segue o fluxo conforme representado na Figura 18. As únicas sincronizações garantidas estão nas saídas de cada bloco. No meio deles as instruções assembler são executadas em paralelo por todo o conjunto de dados.

[12] Especificações de hardware e recursos disponíveis. Fonte: https://learn.microsoft.com/pt-br/windows/win32/direct3d11/direct3d-11-advanced-stages-compute-shader. Acessado em junho 2023.

Figura 18 - Fluxo do pipeline dentro da GPU.



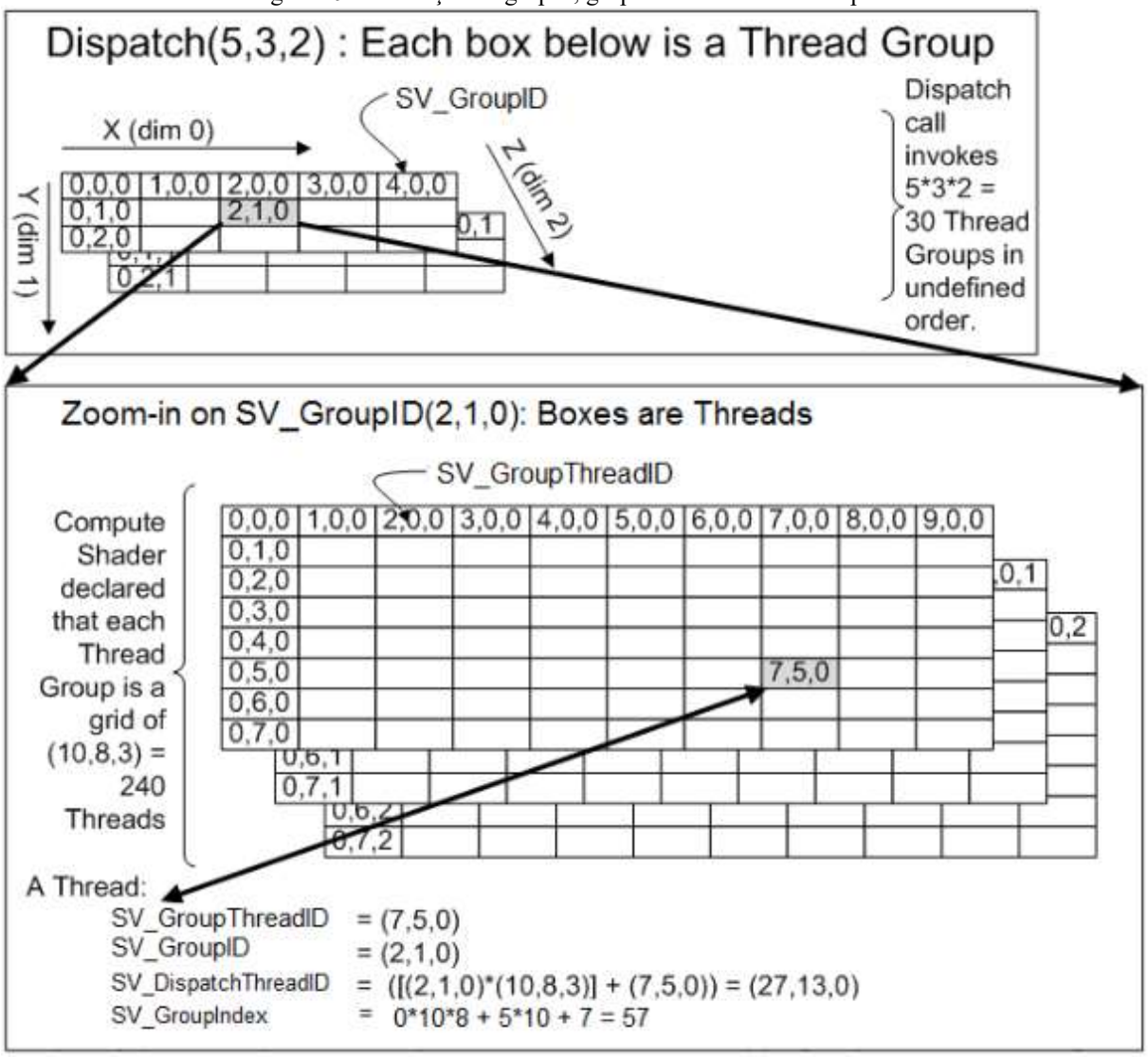

Fonte: (MICROSOFT, 2023, DIRECTX11-LEARN).

Conforme a documentação oficial[13] do DirectX11 no site da empresa Microsoft, ao criar um *comput shader* ele será empilhado no pipeline padrão. Todos esses threads em paralelo possuem severas restrições e gerenciadores. A Figura 19 ilustra a hierarquia e como é possível identificar um thread entre todos. Essa figura mostra a reserva para o processamento de 7200 threads, nessa ilustração temos a execução de 240 fluxos de processamento em paralelo sendo instanciados 30 vezes. Primeiro é declarado quantas instancias de grupos serão implementados, elas serão alocadas em uma matriz tridimensional e dependem principalmente do volume de dados a serem processados. Posteriormente para cada elemento dessa matriz será associado um

[13] https://learn.microsoft.com/pt-br/windows/win32/direct3d11/direct3d-11-advanced-stages-compute-shader (acessado em 18 de junho de 20023)

grupo de threads. Esse grupo também é representado por uma matriz e possui a finalidade de reservar os recursos de hardware. Para identificar um thread dentro do grupo é calculado o que a documentação chama de índice plano. Na Figura 19 ele está identificado como “SV_GoupIndex”. Para identificar um thread entre todos é utilizado um índice de três dimensões denominado “SV_DispachThreadId”. Esse índice é encontrado multiplicando o índice do elemento da matriz de instancias com a capacidade total do grupo de threads e somando seu índice. A plataforma Unity aceita chamadas passando esses valores para uma variável também de três dimensões.

Figura 19 - Ilustração de grupos, grupos de threads e index plano.



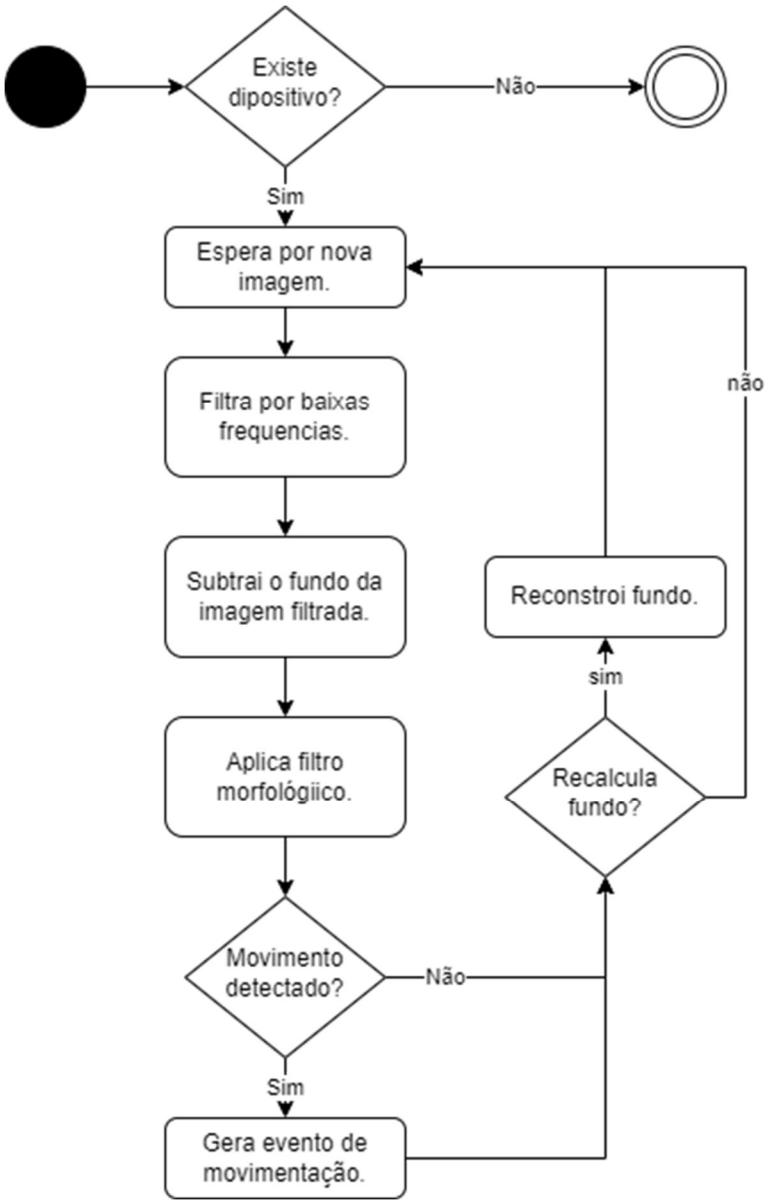

Fonte: (MICROSOFT, 2023, DIRECTX11-LEARN).

Quanto às restrições, a primeira está relacionada ao uso de memória para gravação. Um thread poderá gravar diretamente em sua região que está limitada a 256 bytes. Acesso a regiões de memória compartilhada pelo grupo devem ser feitas indiretamente através de métodos, sendo necessário informar o endereço de memória desejado juntamente com o índice plano do thread destino. Essa memória do grupo é limitada a 16 KB, quanto a memória global da GPU é permitida somente acesso de leitura. Essa memória varia dependendo do hardware instalado, é mais lenta que as demais e sempre será atualizada no final do *pipeline*. Essa memória espelha ou alimenta a memória de transição.

## 4.3 LOGICA E ALGORITMOS

Esse tópico é destinado e explicar com mais detalhes a lógica e os algoritmos utilizados no desenvolvimento, em síntese estão representados na Figura 20. Desse ponto em diante quando o trabalho se referir a imagens binárias ele estará se referindo a imagens de apenas um canal discreto. Nesse caso os valores das cores de seus pixels poderão assumir somente o valor 0 ou 1. Quando mencionar imagens em tons de cinza estará se referindo a imagens de apenas um canal. Quando se referir a imagens coloridas estará se referindo a imagens com 3 canais. Cada um desses canais pode assumir valores inteiros de 0 a 255, caso necessário sua conversão para ponto flutuante é a simples divisão do valor do pixel por 255. A conversão inversa será somente considera a parte inteira da multiplicação do valor flutuante pelo escalar 255.

Figura 20 - Logica Geral




Fonte: Próprio autor.

Conforme mencionado no tópico do framework utilizado, o desenvolvimento desta aplicação foi segmentado em estágios ou etapas bem definidas. Elas estão ilustradas na Figura 20 sendo que muitas dessas etapas serão computadas em *shaders* especiais. Os próximos subtópicos deste documento detalham cada uma delas.

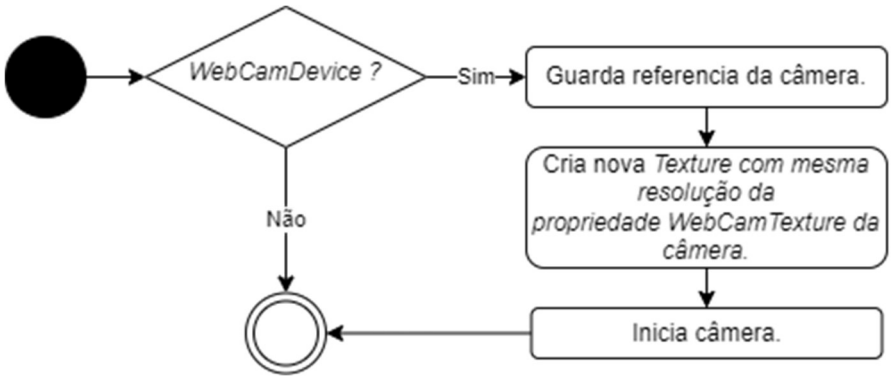
### 4.3.1 Aquisição de Imagens

Aqui o trabalho aborda as duas primeiras etapas sendo ilustradas pela Figura 21 e Figura 22. Conforme visto no capítulo 2 no processo de digitalização a câmera irá converter, amplificar

e mensurar a luz visível. Essa informação é disponibilizada de forma serial via interface padrão USB. A informação é transmitida como um fluxo de bits e começa com as cores do pixel superior à esquerda da matriz dos elementos sensores. Termina com o pixel inferior direito sempre sendo executado em linhas sequenciais e o tipo de informação é um byte por canal com intensidades variando de 0 a 255 inteiros. A câmera utilizada captura até 30 imagens por segundo sendo que a cada imagem um novo fluxo de bytes é gerado e necessita ser lido.

A plataforma Unity encapsula a gestão da câmera e disponibiliza uma interface de alta ordem denominada *WebCamDevice*. Com ela é possível acessar, iniciar, pausar e liberar a utilização da câmera. *WebCamDevice* possui uma propriedade do tipo *WebCamTexture* com a finalidade de conter as informações da última foto registrada pela câmera. Por serem interfaces nativas da plataforma é fácil sua conversão para o tipo *Texture*. Esse tipo será utilizado para guardar uma cópia estática e editável da foto original. Esse tipo também será utilizado ao longo do trabalho para reter as informações e trabalhar em conjunto com os *comput shaders*.

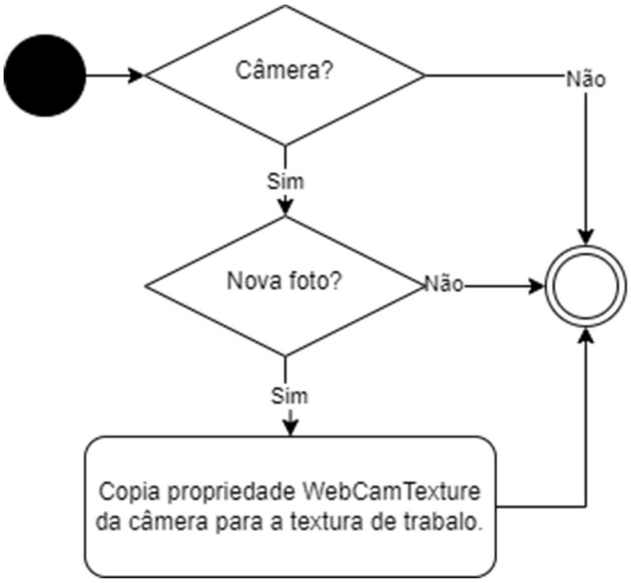
Figura 21 – Existe dispositivo




Fonte: Próprio autor.

A Figura 21 ilustra o início do processo. Nele é necessário verificar se existe alguma câmera disponível. Caso exista será guardada a primeira referência encontrada e logo após é criado uma textura de trabalho do tipo *Texture* que seja compatível com o tipo de foto gerado pela câmera. Para isso é necessário que as dimensões e número de canais sejam iguais, por último a câmera é iniciada. Essa função é chamada somente uma vez no início da aplicação e reserva a câmera para essa aplicação em especifico.

Figura 22 – Espera por nova imagem.




Fonte: Próprio autor.

Após iniciar a câmera, uma outra função é chamada repetidamente. Sua lógica está ilustrada na figura acima. Consiste basicamente de verificar se a referência para câmera continua válida e se houve o recebimento de nova fotografia. Em caso afirmativo ela irá realizar a cópia da nova foto para a textura criada no início do programa.

### 4.3.2 Correção das Distorções das Lentes

Como foi mencionado no tópico de correção de lentes, toda imagem sofre com alguma distorção das lentes que afetam a forma dos objetos. Como o foco deste trabalho está centrado no movimento sendo indiferente quanto à forma, a correção destas distorções pode ser ignorada sem comprometer o resultado. Portanto não serão implementadas.

### 4.3.3 Filtro de Frequência

Esta etapa tem por objetivo reduzir eventuais ruídos que possam incorporar a imagem. Tais ruídos são caracterizados por detalhes dos objetos, reflexos em bordas, objetos muito pequenos que podem ser desprezados e entre outros. Esses conceitos estão contidos no subtópico de filtros.

Sua implementação foi dividida em 2 arquivos. O primeiro é o *comput shader* escrito na linguagem HLSL e representado na Figura 23. Essa figura é um registo da tela da IDE utilizada. Nela as linhas estão indexadas por um número exibido no canto esquerdo de cada linha. Os conceitos matemáticos contidos no subtópico do filtro passa-baixa de media estão abstraídos pelas linhas 23 a 34. É visível que a abstração consiste da soma dos "n" elementos realizada por dois laços "for". Em cada interação é incrementado um contador para registrar efetivamente o número de elementos somados. Por fim a média é calculada através da divisão da soma pelo número de elementos. Essas variáveis são do tipo *float4*[14]. Os três primeiros elementos desse vetor constituem a cor do pixel em cada respectivo canal e normalizada de 0 a 1. O quarto elemento define a opacidade do pixel, ele foi implementado com o único propósito de visualizar a imagem dentro da plataforma Unity. Sendo assim seu valor é atribuído manualmente na linha 35. As linhas 10 a 21 contém a lógica para não ultrapassar tanto os limites da imagem, quanto os limites da janela onde os laços serão aplicados. O conceito de convolução do subtópico 2.3.1 foi implementado através da injeção de dependência na linha 8. A variável "id" armazena o valor corrente do índice do thread no contexto global, esse índice varia de acordo com o tamanho do grupo de threads informado na linha 7. Nesse caso foram implementados 900 threads em paralelo.

A Figura 24 foi retirada do segundo arquivo escrito na linguagem C#, sendo o responsável por chamar a execução do *shader* dentro da Unity. A linha 41 localiza uma referência para a função compilada do arquivo anterior. As linhas 42 a 46 carregam os dados a serem processados. Essas informações são retidas dentro da memória de transição. Antes de disparar sua execução será necessário informar quantos grupos serão criados. Para uma convolução de toda a área da imagem o *shader* deve ser executado para cada pixel, isso é feito dividindo as dimensões da imagem pelas dimensões do grupo de threads.

[14] *Float4* são matrizes unidimensionais constituídas por 4 elementos do tipo *float*. Caso o contexto seja aplicado a cor os elementos são referenciados respectivamente pelas letras r,g,b,a. Caso o contexto seja posição as letras mudam respectivamente para x,y,z,w.

Figura 23 - Comput Shader Passa-Baixa

```
1  //Filro para deixar somente as baixas frequencias passarem
2  #pragma kernel PassaBaixa
3
4  RWTexture2D<float4> resultado,texturaEntrada;
5  int janela,width, height;
6
7  [numthreads(30,30,1)]
8  void PassaBaixa (uint3 id : SV_DispatchThreadID)
9  {
10     uint MX = (uint)width;
11     uint MY = (uint)height;
12
13     uint paradaX = id.x+janela+1;
14     uint paradaY = id.y+janela+1;
15     if(paradaX>MX) paradaX=MX;
16     if(paradaY>MY) paradaX=MY;
17
18     uint inicioX = id.x-janela;
19     uint inicioY = id.y-janela;
20     if(inicioX<0) inicioX = 0;
21     if(inicioY<0) inicioY = 0;
22
23     float4 soma = float4(0,0,0,0);
24     uint interacoes=0;
25     for(uint DX=inicioX; DX<paradaX; DX++){
26         for(uint DY=inicioY; DY<paradaY; DY++){
27             interacoes++;
28             soma+=texturaEntrada[uint2(DX,DY)];
29         }
30     }
31     float4 media = float4(0,0,0,0);
32     if(interacoes>0){
33         media = soma/((float)interacoes);
34     }
35     media.a = 1;
36     resultado[id.xy] = media;
37 }
```

Fonte: Próprio autor.

Figura 24 - Chamada da execução de comput shader

```
40     public void Filtrar(){
41         var id = passaBaixa.FindKernel("PassaBaixa");
42         passaBaixa.SetTexture(id, "texturaEntrada", entrada.texture);
43         passaBaixa.SetTexture(id, "resultado",  filtrada.texture);
44         passaBaixa.SetInt("janela", tamanhoJanela);
45         passaBaixa.SetInt("width", entrada.texture.width);
46         passaBaixa.SetInt("height", entrada.texture.height);
47         passaBaixa.Dispatch(id, entrada.texture.width / 30, entrada.texture.height / 30, 1);
48     }
```

Fonte: Próprio autor.

No caso a imagem a ser processada possui dimensões 1920 x 1080, logo foram instanciados $64 * 36 * 1 = 2304$ grupos com 900 threads cada. Essa ação é chamada na linha 47, ao chamar essa linha a aplicação ficará esperando até todos os grupos serem executados. Seu resultado será lido da memória de transição utilizando a referência passada pela linha 43.

### 4.3.4 Remoção do Fundo

Tem por objetivo subtrair o fundo ou cenário da imagem e basicamente será feita a subtração de duas imagens e pelo fato de a câmera ser estática os itens remanescentes serão os objetos em movimento.

A Figura 25 contém o código da lógica do tópico 2.4. A linha 15 realiza a subtração da foto atual com a referência de fundo. As linhas 16 a 18 realizam a função módulo do valor, ou seja, retornam sempre o número positivo. A linha 19 realiza a soma das diferenças que são comparadas na linha 22. Caso exceda um determinado limite significa que houve movimentação nesse pixel e seu retorno será o valor 1, caso contrário retornará 0. A diferença nesse *shader* é o fato dele possuir apenas um canal e retornar valor 0 ou 1.

Figura 25 - Shader Subtrator

```
13  void Subtrator (uint3 id : SV_DispatchThreadID)
14  {
15      float4 dif = texturaAtual[id.xy] - fundo[id.xy];
16      dif.r = (dif.r>0) ? dif.r : -dif.r;
17      dif.g = (dif.g>0) ? dif.g : -dif.g;
18      dif.b = (dif.b>0) ? dif.b : -dif.b;
19      float variacao = dif.r+dif.g+dif.b;
20      float res = 0.0f;
21
22      if (variacao>limiar*3) {
23          res = 1.0f;
24      }
25
26      resultado[id.xy] = res;
27  }
```

Fonte: Próprio autor.

Todos os shaders necessitam que algum método os instancie. As imagens dessas chamadas serão omitidas por serem muito semelhantes a Figura 24, basicamente é alterado somente o texto passado como parâmetro para o método *FindKernel* para o nome do shader.

### 4.3.5 Reconstrução do Fundo

Esta etapa tem como objetivo incorporar mudanças no cenário, seja incluir novos elementos ou remover elementos ausentes. Conforme mencionado no subtópico de reconstrução de fundo, os métodos convencionais envolvem o cálculo de médias, desvio padrão

e variação local por pixel. Esse trabalho adotou uma simplificação dessa técnica. Como o intuito é sempre reduzir a intensidade da diferença entre a imagem atual da imagem de fundo. O trabalho introduziu uma taxa fixa $t$ que varia de 0 a 1. Isso será feito recalculando o novo valor do fundo multiplicando a diferença da imagem atual pela taxa. Em termos matemáticos será recalculado o valor de $p(x, y)$ do tópico 2.4 utilizando a fórmula $t * \big(a_i(x, y) - p(x, y)\big) + p(x, y)$ . Quanto mais próxima de 1 for a taxa de reajuste, mais rápido a imagem de fundo será readequada às variações do cenário.

Figura 26 - Reconstruir Fundo

```
12  void FazFundo (uint3 id : SV_DispatchThreadID)
13  {
14      float4 dif = texturaAtual[id.xy] - fundo[id.xy];
15      float4 correcao = dif * fator;
16      float4 novo = fundo[id.xy] + correcao;
17      novo.a = 1.0f;
18      fundo[id.xy] = novo;
19  }
```

Fonte: Próprio autor.

A Figura 26 mostra a implementação dessa taxa de variação fixa em um *comput shader*. A linha 14 realiza a subtração da cor da imagem atual pela cor do fundo atual. A linha15 aplica a taxa como fator de correção sobre a diferença e posteriormente a linha 16 soma essa correção ao pixel atual do fundo. A linha 18 armazena essa nova informação.

### 4.3.6 Filtro Morfológico

Aqui o trabalho faz uso dos conceitos do subtópico de filtro morfológico. A intenção é remover pequenas variações para não serem computadas como movimentação. Para isso foi utilizado um filtro do tipo erosão. A forma escolhida para o filtro foi o formato de cruz de tamanho “n” centrada no pixel alvo.

A Figura 27 contem sua implementação que começa pelo cálculo dos limites da cruz entre as linhas 13 a 18. Em seguida é verificado se os limites da cruz não irão exceder os limites da imagem. O laço da linha 29 percorre a cruz no sentido horizontal enquanto o laço da linha 37 no sentido vertical. Caso algum pixel possua o valor 0 significa que a forma analisada não

corresponde a uma cruz. Por fim caso a execução do programa ultrapasse os dois laços será atribuído o valor 1 ao pixel correspondente na imagem resultado.

Figura 27 - Filtro Morfológico Cruz

```
11  void Morfologico (uint3 id : SV_DispatchThreadID)
12  {
13      uint MX = (uint)width;
14      uint MY = (uint)height;
15      uint paradaX = id.x+janela+1;
16      int inicioX = id.x-janela;
17      uint paradaY = id.y+janela+1;
18      int inicioY = id.y-janela;
19
20      if(inicioX<0 || inicioY<0
21          || paradaX>MX || paradaY>MY) {
22          resultado[id.xy] = 0.0f;
23          return;
24      }
25
26      float pixel = 0.0f;
27      uint DX = inicioX;
28      uint DY = id.y;
29      for(DY = id.y; DX<paradaX; DX++){
30          pixel = subtraida[uint2(DX,DY)];
31          if(pixel == 0 ){
32              resultado[id.xy] = 0.0f;
33              return;
34          }
35      }
36      DY=inicioY;
37      for(DX = id.x; DY<paradaY; DY++){
38          pixel = subtraida[uint2(DX,DY)];
39          if(pixel.x == 0){
40              resultado[id.xy] = 0.0f;
41              return;
42          }
43      }
44
45      resultado[id.xy] = 1.0f;
46  }
```

Fonte: Próprio autor.

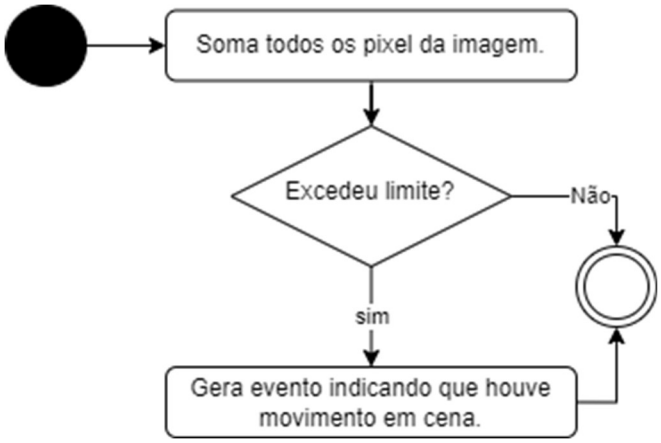
A lógica da Figura 27 é aplicada na imagem binária resultante do subtópico 4.3.4 e seu resultado também é uma imagem binária pois seus pontos de retorno externam 0 ou 1.

### 4.3.7 Evento de Movimento

Após a aplicação do filtro morfológico o resultado é uma imagem binária em que todos os pixels diferentes de 0 representam uma movimentação. Essa etapa final consiste em somar

todos os valores dos pixels da imagem resultante do subitem anterior e comparar com um limite. Caso o número exceda esse limite significa que houve alterações suficientes na imagem para considerar uma movimentação. Quando a movimentação for detectada um evento deve ser disparado, a Figura 28 ilustra essa lógica.

Figura 28 - Gera Evento por Movimento




Fonte: Próprio autor.

Pelo fato de ser um trabalho com fins acadêmicos, será utilizado como evento uma simples mensagem de texto para informar a porcentagem e o número de pixels alterados no frame corrente. O próximo capitulo corresponde a um ensaio realizado utilizando o software produzido.

# 5 RESULTADO

Nesse capitulo serão expostos os resultados de ensaios práticos obtidos a partir da implementação do aplicativo construído no capítulo anterior. Essa implementação ocorreu através de uma câmera fixada próximo ao teto de uma residência. Também foi utilizado um arquivo de vídeo gerado pela mesma câmera. Esse arquivo serviu para comparar resultados utilizando métricas diferentes sobre os mesmos dados de entrada.

## 5.1 AQUISIÇÃO DE IMAGEM

Uma câmera semelhante ao ilustrado na Figura 17 foi fixada ao alto de um ambiente externo. Ela foi utilizada para registrar o ambiente em vídeo implementando assim o software desenvolvido no subtópico 4.3.1. A Figura 29 ilustra uma dessas imagens. Esse item produz uma imagem digital ordenada na forma de uma matriz bidimensional. Essa matriz possui tamanho de 1920 pixels na horizontal por 1080 pixels na vertical. Cada pixel composto de 3 canais com 256 níveis distintos cada. Pelo fato desses níveis estarem normalizados no range de 0 a 1. Eles utilizam variáveis do tipo *float* com 2 bytes cada. Dessa forma a imagem produzida ocupa um total de $(1920 * 1080) * 3 * (2 * 8) = 99532800$ bits.

Figura 29 - Imagem Capturada.

Fonte: Próprio autor.

Utilizando as imagens capturadas pela câmera foi produzido um arquivo de vídeo contendo 31 minutos e 40 segundos. O vídeo foi registrado a uma taxa de 30 quadros por segundo, está na mesma resolução das fotos e possui tamanho total de 3,78GB.

## 5.2 FILTRO FREQUÊNCIA

A implementação do subtópico 4.3.3 utilizou como base a imagem capturada pelo subtópico anterior. Seu resultado está ilustrado na Figura 30 onde foi aplicado um filtro de frequência do tipo passa-baixa de média.

Figura 30 - Resultado Filtro Passa-Baixa

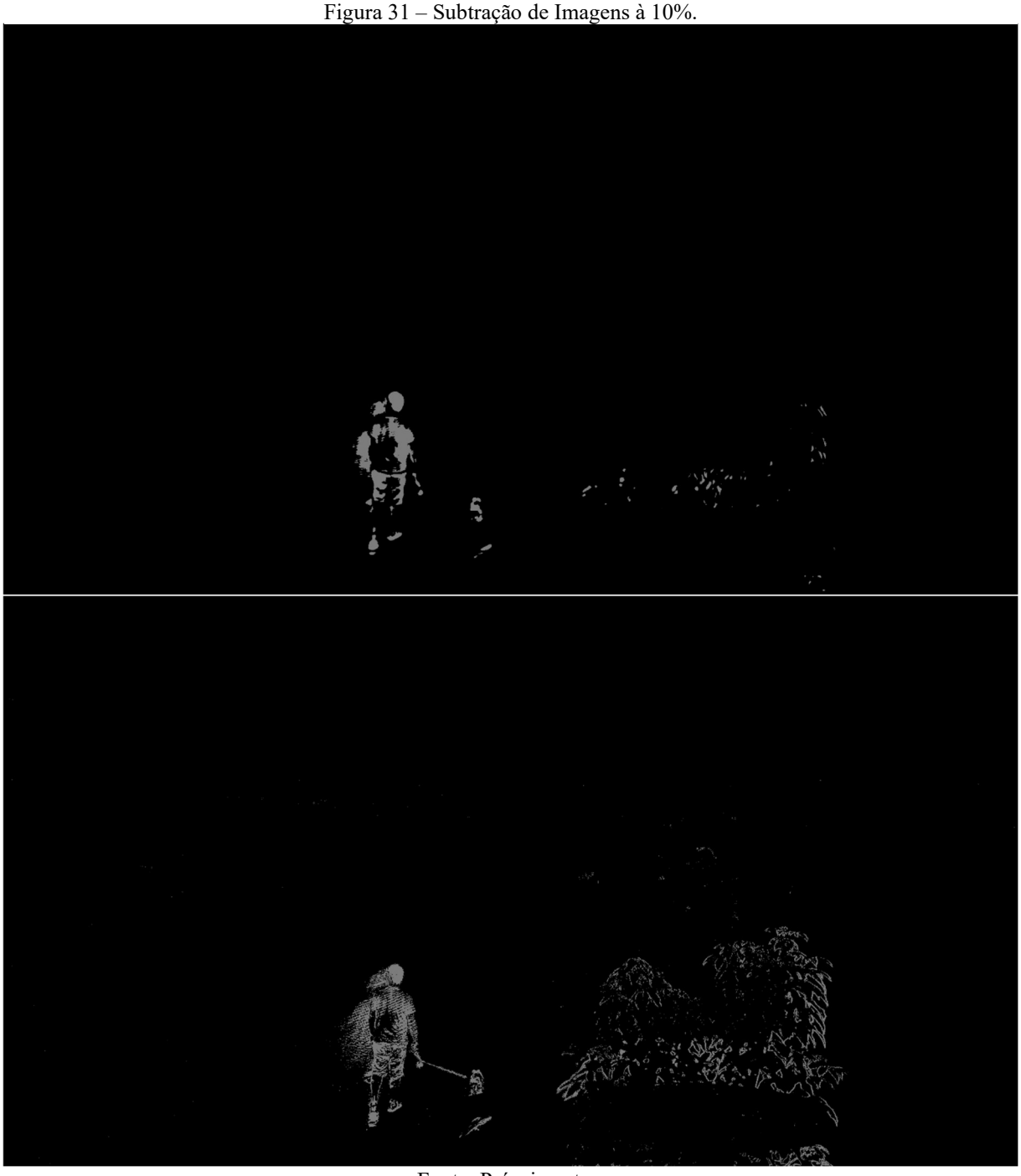
Fonte: Próprio autor.

Na figura acima foram utilizados 7 pixels como tamanho da janela do filtro de convolução. É visível a redução do nível de detalhes após aplicação do filtro e seu aspecto é semelhante à uma imagem fora de foco com uma maior suavização das bordas e interseção dos objetos.

## 5.3 SUBTRATOR

A Figura 31 contém duas imagens binárias, ambas representam o resultado da implementação do subtópico 4.3.4 tendo 10% como limite de tolerância na variação das cores. A imagem de cima mostra o resultado aplicando o filtro com as configurações descritas no tópico anterior 5.2.

Figura 31 – Subtração de Imagens à 10%.



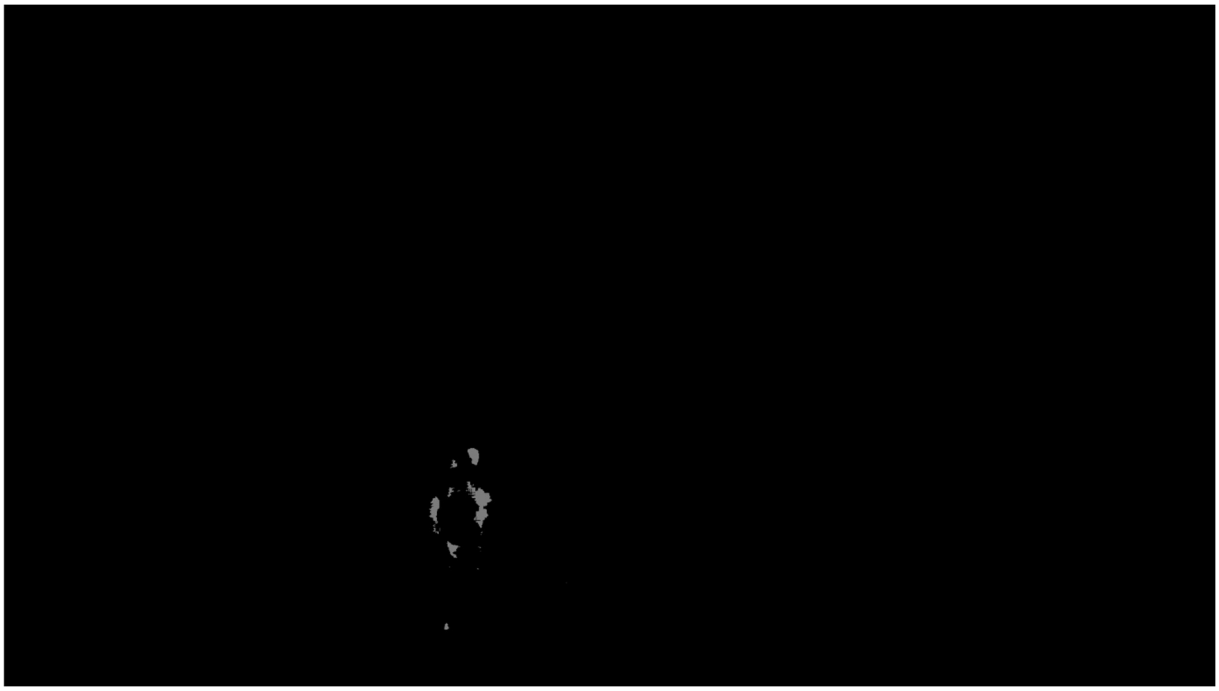
Fonte: Próprio autor.

A imagem de baixo da Figura 31 foi gerada apenas para comparação posterior. Nela foi removido o filtro de frequência alterando o tamanho da janela do tópico anterior para 1. Em ambas, os pixels mais claros representam as alterações da foto atual com a referência de fundo.

## 5.4 EROSÃO

Foi implementado o subtópico do filtro morfológico com tamanho de 6 pixels na imagem binária do tópico anterior e seu resultado pode ser observado na Figura 32.

Figura 32 - Região Considerada Com Movimento



Fonte: Próprio autor.

Os únicos pontos com alteração remanescentes continuam representados por cores mais claras na figura acima. Esse trabalho considera essa região como movimento.

## 5.5 FUNDO

O subtópico de reconstrução de fundo foi aplicado a cada novo frame com uma taxa de correção de 0,034. A Figura 33 representa a imagem de fundo em um dado momento.

Figura 33 - Imagem de Fundo

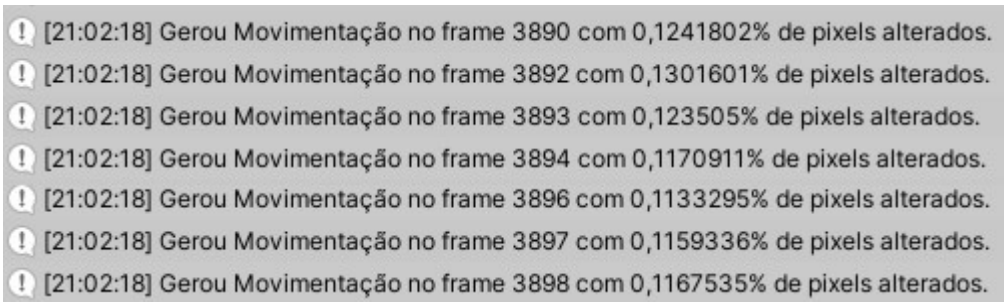
Fonte: Próprio autor.

Na figura acima é possível verificar um rastro deixado pelo objeto em movimento decorrente de estados passados. Esse rastro é dissipado conforme o subtópico de reconstrução do fundo é aplicado repetidas vezes.

## 5.6 EVENTO

O subtópico de evento de movimento foi aplicado com um limite de 0,10% a cada nova imagem do resultado gerado pelo tópico erosão. O evento somente ocorre se o limite for excedido e produz uma informação no formato de texto. Esse texto contém um número que representa o frame em que houve a geração do evento e a porcentagem de pixels com movimento. A Figura 34 mostra alguns eventos gerados em um determinado instante de tempo.

Figura 34 - Eventos Gerados



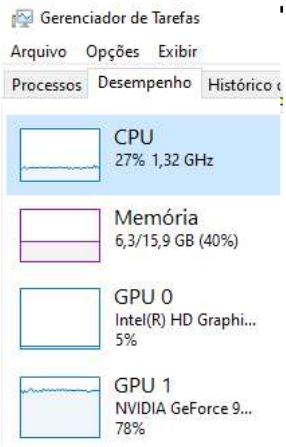

Fonte: Próprio autor.

A figura acima mostra eventos com o frame que ocorreu o movimento e sua taxa de variação. O próximo tópico abordará questões relacionadas ao desempenho desses eventos.

## 5.7 PERFORMANCE

O software executou em compasso com a câmera sem descarte de quadros. Durante o ensaio com duração de 31 minutos e 40 segundos foram processados 57013 quadros. Sendo gerado 1863 eventos de movimento. Foi capturado um instante de tempo do Gerenciador de Tarefas do sistema operacional no momento da execução. Esse instante mostra a utilização de todos os recursos do computador e parte dele está na Figura 35.

Figura 35 - Gerenciado de Tarefas




Fonte: Próprio autor.

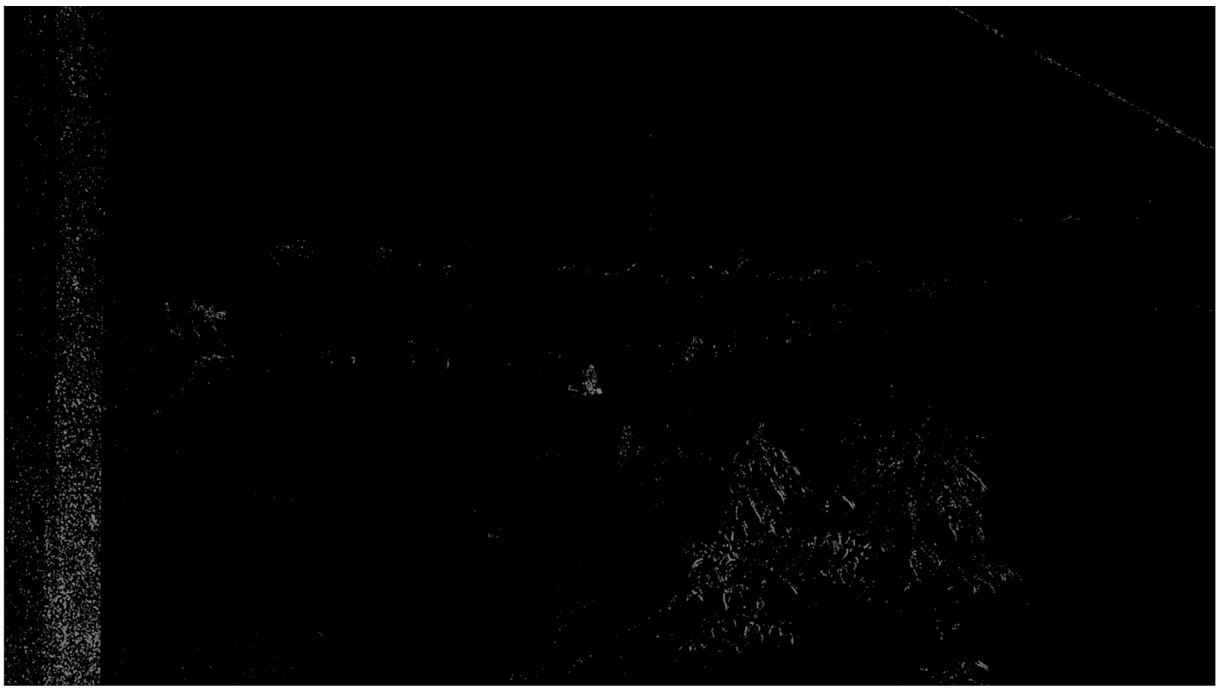
É possível observar a CPU operando a 27% de sua capacidade total enquanto uma GPU está operando a 78%. Por último foi observado uma redução drástica no nível da bateria. Normalmente o computador utilizado no ensaio opera até 2 horas com uma única carga de bateria, mas após 18 minutos de execução foi necessário conectar o computador à uma fonte de energia externa.

### 5.7.1 Falso Positivo

Esse trabalho entende falso positivo como o evento gerado pelo tópico anterior em momentos quando não há movimentação no ambiente. Foram observados falsos positivos no início da execução do programa até o frame 82. Sempre na retirada de um elemento do cenário ocorriam falsos positivos na região antiga de ocupação do elemento. Ao reduzir o tamanho

mencionado no tópico 5.4 para 2, variações muito bruscas de iluminação começaram a gerar falsos positivos. Esses falsos eventos tinham durações de aproximadamente 20 frames e estão representados na Figura 36.

Figura 36 - Falso Positivo com Variação de Iluminação



Fonte: Próprio autor.

As regiões em branco na figura acima surgiram após a diminuição das janelas do filtro morfológico e do filtro de média. A soma das pequenas áreas gerou um evento de movimentação quando não deveria, caracterizando um falso positivo.

### 5.7.2 Falso Negativo

Nesse contexto entende-se por falso negativo quando houve movimentação em cena, mas não houve geração de eventos pelo tópico anterior. Foram observados falsos negativos em objetos muito pequenos e carros distantes.

## 5.8 ANÁLISE DOS RESULTADOS

Através dos resultados apresentados nesse capítulo, fica claro que o processo de aquisição de imagens está funcionando corretamente. Seguido do filtro de frequências que praticamente elimina todas as bordas e detalhes finos. Esses detalhes poderiam gerar muito ruido para etapas posteriores de processamento. Dessa forma é possível destacar melhor somente os objetos desejados. A Figura 31 exemplifica essa diferença de detalhes e ruído. Nela é possível, também, observar vegetações próximas à câmera sendo detectadas. Essa detecção possivelmente ocorreu pelo movimento da vegetação através do vendo. Porém objetos mais distantes também se movem e não são detectados. Então esse fato mostra uma possível relação entre a distância do objeto e a câmera. Outro fato que reforça essa relação de distância é o subtópico de falso negativo. Se pensado de forma analítica essa relevância da distância realmente existe, pois à medida que os objetos se afastam da câmera eles ocupam uma área menor dentro da imagem. Sendo assim chega um momento em que são descartados por não atingirem o mínimo esperado. O centro da Figura 36 exemplifica essa redução de área através da distância. Nela a mesma pessoa e animal da Figura 31 estão presentes, mas são descartados por possuir pouca área.

A Figura 32 mostra como o filtro de erosão atua de forma satisfatória removendo os itens pequenos considerados como ruído. Dessa forma regiões compreendia pelo animal e a vegetação próxima a câmera foram excluídas. Um efeito negativo é a redução da área do objeto alvo. Um método para minimizar esse efeito poderia ser implementado. Ele seria através do uso de outro filtro, denominado por Meneses (2012) de filtro morfológico de dilatação. Dessa forma o objeto em movimento representado na Figura 32 teria seu tamanho mais semelhante ao real.

A lógica para geração da imagem de referência do fundo através de uma taxa fixa se mostrou simples e eficiente. É visível na Figura 33 como esse método tem facilidade em incorporar novos objetos do cenário. No entanto foi relatado um falso positivo no início da aplicação e com longo período. Isso se deve pelo fato de a imagem da referência do fundo começar totalmente em preto, sendo assim uma melhoria para o software seria a implementação de uma condição de início. Nessa condição a primeira imagem seria utilizada já como referência de fundo. A lógica do subtópico de evento de movimento também pode ser melhorada a fim de reduzir falsos eventos. Sua implementação conta apenas com um limite inferior para disparar o evento, mas um limite superior também poderia ser inserido. Caso atingido esse novo limite a taxa de atualização do fundo seria aumentada momentaneamente e o evento não seria gerado.

Essa alteração visa principalmente eventos de variações bruscas na iluminação, pois neles uma grande quantidade de área da imagem é alterada como mostra a Figura 36.

Quanto ao tópico de performance é possível observar que aproximadamente 96,7% do tempo de vídeo não possui movimento. Realizando uma conversão simples dessa porcentagem sobre o tópico 5.1. Significa que um arquivo de 3,78 gigabytes pode ser substituído por um arquivo de apenas 126 megabytes. Em termos de armazenamento e transporte de dados é uma relação bem favorável. Em termos de recursos humanos, pode proporcionar que uma única pessoa monitore vários ambientes. Também fica claro que a aplicação suporta a aquisição e o processamento de 30 quadros por segundo. Sendo o recurso mais utilizado a GPU com 78% e liberando tempo de processamento da CPU. Um fato não previsto foi o acentuado consumo de energia. O computador passou a consumir mais de 6 vezes o usual, possivelmente por essa utilização elevada da GPU. Dependendo das condições esse fato pode ser impeditivo de sua implementação. Ele pode ser contornado utilizando um hardware mais eficiente. Um exemplo seria a placa de desenvolvimento Raspberry Pi-4B. Conforme site do fabricante[15] essa placa teria condições de executar o processamento consumindo menos de 15 watts.

[15] https://www.raspberrypi.com/ (Acessado em 18/06/2023)

# 6 CONSIDERAÇÕES FINAIS

Neste trabalho foram abordados como as imagens digitais são formadas, como são representadas digitalmente e como detectar movimento em uma cena. Conceitos matemáticos foram aplicados sobre essa representação digital de imagem. Eles destacaram pontos de interesse, removeram ruídos e alteraram a forma de elementos digitais. Esses conceitos foram implementados através de algoritmos e plataformas de desenvolvimento de software. Através da plataforma foi possível gerar um software executável onde foram realizados ensaios utilizando equipamentos reais. Complementando a área de formação de imagens digitais. Foi realizada uma pesquisa específica sobre o tema de segmentação de fundo de cenário. Essa pesquisa indicou as principais técnicas e dispositivos empregados na atualidade. Inclusive o empregado nesse trabalho.

Através dos ensaios ficou claro que todos os objetivos propostos foram atingidos. Dentre os objetivos principais ficou evidente que o software cumpre o proposto de informar quando há movimento em cena com processamento em tempo hábil. É altamente aplicável em um ambiente real, visto que não demanda de muitas necessidades especiais. Em caso de aplicação tem potencial de otimizar recursos humanos, diminuir armazenamento e transmissão de dados.

A principal desvantagem observada reside no aumento de consumo de energia pelo computador que executa o software. Com isso surgiu a hipótese de utilizar uma placa de desenvolvimento Raspberry Pi-4B de menor consumo. Portanto fica como sugestão para novo trabalho a validação dos métodos desenvolvidos aqui sobre esse novo hardware sugerido.

# REFERÊNCIAS


ALIEXPRESS. PICO FLEXX2. Disponível em: https://pt.aliexpress.com. Acessado em: 18 de julho,2023. Palavra chave de pesquisa: “Pico Flexx2”.

ALIEXPRESS. REALSENSE. Disponível em: https://pt.aliexpress.com. Acessado em: 18 de julho,2023. Palavra chave de pesquisa: “Realsense D435”.

ALIEXPRESS. WEBCAM FULL HD. Disponível em: https://pt.aliexpress.com. Acessado em: 18 de julho,2023. Palavra chave de pesquisa: “Webcam Fhd”.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TECNICAS. **NORMA BRASILEIRA:** ABNT NBR 6024. Rio de Janeiro: ABNT, 2012 ISBN: 978-85-07-0347-2

ASSOCIAÇÃO BRASILEIRA DE NORMAS TECNICAS. **NORMA BRASILEIRA:** ABNT NBR 14724. Rio de Janeiro: ABNT, 2011 ISBN: 978-85-07-02680-8

ASSOCIAÇÃO BRASILEIRA DE NORMAS TECNICAS. **NORMA BRASILEIRA:** ABNT NBR 15287. Rio de Janeiro: ABNT, 2011 ISBN: 978-85-07-02681-5

BEER, Maik & Haase, Jan & Ruskowski, Jennifer & Kokozinski, Rainer. (2018). **Background Light Rejection in SPAD-Based LiDAR Sensors by Adaptive Photon Coincidence Detection. Sensors**. 18. 4338. 10.3390/s18124338.

COLAÇO, Kenion César Michelato. **Computação Gráfica e Processamento de Imagens**. Londrina: Editora e Distribuidora Educacional S.A., 2018 ISBN: 978-85-522-0323-0

GONZALES, Rafael C. **Processamento de imagens digitais**. São Paulo: Editora Edgard Blucher LTDA, 2005.

HAMAMATSU. **Concepts in Digital Imaging Technology:** Full-Frame CCD Architecture. Disponível em: https://hamamatsu.magnet.fsu.edu/articles/fullframe.html. Acessado em 18 junho, 2023.

INTEL. **REALSENSL-SDK.** Disponível em: https://www.intelrealsense.com/sdk-2/. Acessado em 18 junho, 2023.

KAEHLER, Adrian; Bradski, Gary. **Learning OpenCV3: COMPUTE VISION IN C++ WITH THE OPENCV LIBRARY**. Boston: O´REILLY, 2017 ISBN: 978-1-491-93799-0

MENESES, Paulo Roberto; Almeida, Tati de. **Introdução ao Processamento de Imagens de Sensoriamento Remoto.** Brasília: E-book. Disponível em: https://www.academia.edu/40388021/INTRODU%C3%87%C3%83O_AO_PROCESSAMENTO_DE_IMAGENS_DE_SENSORIAMENTO_REMOTO, 2012. Acesso em 01 maio, 2022.

MICROSOFT. **DIRECTX11-LEARN**. Disponível em: https://learn.microsoft.com/pt-br/windows/win32/direct3d11/direct3d-11-advanced-stages-compute-shader. Acessado em: 18 de junho, 20023.

MICROSOFT. **KINECT**. Disponível em: https://learn.microsoft.com/pt-br/windows/apps/design/devices/kinect-for-windows. Acessado em: 18 de junho, 2023.

OPENCV. **Open Source Computer Vision.** Disponível em: https://docs.opencv.org/4.5.2/d9/d0c/group__calib3d.html. Acessado em: 18 de junho, 2023.

PEDRINI, Hélio; SCHWARTZ, William R. **Análise de imagens digitais: princípios, algoritmos e aplicações**. São Paulo: Cengage Learning Brasil, 2007 ISBN 978-85-221-2836-5.

PMD. **3D Vision for your Product**: The Flexx2. Disponível em: https://3d.pmdtec.com/en. Acessado em: 18 de junho, 2023.

RASPBERRY. **Raspberry Pi-4B**. Disponível em: https://www.raspberrypi.com/. Acessado em 18 junho, 2023.

SCHWALBEL,2005. **GEOMETRIC MODELLING AND CALIBRATION OF FISHEYE-LENS.** Disponível em: https://www.semanticscholar.org/paper/GEOMETRIC-MODELLING-AND-CALIBRATION-OF-FISHEYE-LENS-Schwalbe/cd0333e5cd89f7b58a58083d1a2fffbadbeb804d?p2df. Acesso em 06 novembro, 2022

ST. **Time-of-Flight ranging sensor: DocID029104 Rev 5**. Revisão E-book. Disponível em: https://www.st.com/resource/en/datasheet/vl53l0x.pdf. Acesso em 01 janeiro,2023

UNIVERSIDADE FEDERAL DE SANTA MARIA. **MANUAL DE DISSERTAÇÕES E TESES DA UFS**M: Estrutura e Apresentação Documental para Trabalhos Acadêmicos. Santa Maria: UFSM, 2021. ISBN: 978-65-5716-052-7

USB-IF. **DOCUMENT LIBRARY**. Disponível em: https://www.usb.org/documents . Acesso em: 18 junho 2023.

VACAVANT, Antoine; Chateau, Thierry Alexis Wilhelm; Lequièvre, Laurent. **A benchmark dataset for outdoor foreground/background extraction**. WorkshopVision-ACCV 2012, páginas 291–300. Primavera, 2013.

ZHANG, Zhengyou. **GEOMETRIC-MODELLING-AND-CALIBRATION-OF-FISHEYE-LENS.** IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 22, N. 11, NOVEMBER 2000, pag. 1330 - 1334.