Insper

Reconhecimento e Extração de Marcas em Vídeos de Partidas de Futebol

Jose Fernando de Melo Cruz(josefmc@al.insper.edu.br)
Gabriel Noal (gabrielno1@al.insper.edu.br)
João Pedro Montefeltro Junqueira Meirelles (joaopmjm@al.insper.edu.br)

Trabalho de Conclusão de Curso

Relatório Final
do Projeto Final de Engenharia

São Paulo - SP
Junho 2022

Jose Fernando de Melo Cruz
Gabriel Noal
João Pedro Montefeltro Junqueira Meirelles

Relatório Final do Projeto Final de Engenharia

Relatório apresentado ao curso de Engenharia, como requisito para o Trabalho de Conclusão de Curso.

Professor Orientador: Prof. Raul Ikeda Gomes da Silva

Mentor na Empresa: Diego Sanchez Gallo

Coordenador TCC/PFE: Prof. Dr. Luciano Pereira Soares

São Paulo - SP
Junho 2022

Jose Fernando de Melo Cruz
Gabriel Noal
João Pedro Montefeltro Junqueira Meirelles

Reconhecimento e Extração de Marcas em Vídeos de Partidas de Futebol

Projeto Final de Engenharia apresentado ao programa Graduação em Engenharia de Computação como requisito parcial para a obtenção do título de Bacharel em Engenharia.

Orientador: Prof. Raul Ikeda Gomes da Silva

Banca Examinadora

_____

Raul Ikeda Gomes da Silva

Insper

_____

Luciano Silva

Insper

_____

Fábio José Ayres

Insper

Sumário

RESUMO 3

INTRODUÇÃO.. 3

1. ESCOPO DO PROJETO 4
2. DESAFIOS 4
3. RECURSOS 7
4. CRONOGRAMA 8
5. MAPEAMENTO DOS STAKEHOLDERS 9
6. REVISÃO DO ESTADO DA ARTE 10
7. DESENVOLVIMENTO 10
   a. ESTRUTURA DE ORGANIZAÇÃO DA EXECUÇÃO DO PROJETO 10
   b. DESENVOLVIMENTO DO PROCESSAMENTO DE IMAGEM 11
8. DESENVOLVIMENTO E COLETA E ANÁLISE DE DADOS 12
9. DESENVOLVIMENTO DO SERVIÇO API 13
10. COLETA DE DADOS SOBRE O PROJETO 14
11. ANÁLISE DOS DADOS COLETADOS 18
12. RESULTADOS PROCESSAMENTO DE IMAGEM 19
13. BENCHMARK 21
14. RESULTADOS API 22
11. CONCLUSÕES E TRABALHOS FUTUROS 23

REFERÊNCIAS 24

‘

Resumo

A área de marketing em eventos esportivos recebe em investimentos altíssimos, porém os meios usados para garantir entrega do contrato dependem de uma pessoa assistindo ao jogo e anotar os dados em um papel ou tabela, e mesmo assim ainda são dados rudimentares e com possíveis erros humanos, e isso apenas para propagandas na tela de emissoras, e sobre as marcas expostas no campo, ninguém faz esse trabalho. Este projeto tem como objetivo desenvolver um SaaS (*Software as a Service*) para auxiliar na análise de investimento em eventos esportivos, usando visão computacional para identificá-las e extrair métricas como localização na tela, tempo de exposição e área ocupada. Para isso, utilizando-se da biblioteca de visão computacional OpenCV, tem-se acesso ao algoritmo SIFT (*scale-invariant feature transform*) para identificação de pontos interessantes chamados *features*, que são usadas para comparar 2 imagens e achar similaridades, e esse software disponível na nuvem através de uma API. Assim conseguimos tirar métricas como o tempo de exposição, proporção da tela ocupou, auxiliando assim ao time que usar a ferramenta com mais dados para tomada de decisões.

Introdução

Marketing em eventos esportivos passam dos bilhões em investimento mundialmente, apenas no Brasil o futebol movimenta 53 Bilhões de reais. Mas para garantir que as empresas que querem colocar suas marcas a vista do telespectador, empresas como IBGE contratam pessoas para assistir os jogos e anotar quando veem a marca ou não, um trabalho manual, que fica suscetível a erros humanos.

Emissoras ofertam um serviço de merchandising digital (marcas aparecendo digitalmente na tela, ou alguma animação que acontece em determinados momentos do jogo), mas para marcas que expõem nos campos e painéis dos estádios ou que pagam para estar na camisa dos jogadores não existem maneiras de acompanhar propriamente seus impactos e esse desafio não temos exemplos de soluções eficazes.

Este projeto tem por objetivo desenvolver um sistema capaz de reconhecer marcas comerciais expostas em partidas de futebol, a fim de extrair métricas como o tempo de exposição, localização na tela e tamanho proporcional a tela. Ideia advinda de um cliente da Dell, mas sem desenvolvimento por falta de validações. Acredita-se que o volume de dados oriundos do sistema, proporcionará um potencial aumento de valor nas análises de investimento e consultorias, assegurando um serviço de maior qualidade.

Além de apenas uma ferramenta, o projeto procura se aproximar a um serviço, portanto desenvolvendo-a em uma API (*Aplication Programming Interface*) com o objetivo de ser de fácil uso.

Utilizando processamento de para extração e identificação da marca em uma partida de futebol, incluindo aparições dela em telões a roupas dos jogadores, podendo definidas escolhidas uma ou mais logos, e a idealização de métricas que atendam os objetivos.

Essas métricas, associadas a outras devem possibilitar, não se limitando, a melhoria da análise, como também a recomendação, de investimento. Portanto, espera-se que a qualidade do serviço aumentará, com baixo, se nenhum, acréscimo de custo, necessitando apenas de um o arquivo de vídeo (muitos disponíveis publicamente e gratuitamente) e a escolha e extração de uma amostra da arte ou logo.

O programa não possui métricas, constantes ou variáveis atreladas ao esporte de futebol, logo, a aplicação para outros eventos esportivos.

‘

Comparando com hoje, as empresas que colocam seus nomes no jogo, olham muito para métricas após a partida como aumento de receita, *feedback* dos compradores (questionando como foi o primeiro contato com a marca), e outros canais, além disso, as emissoras e estádios, quando eles vendem propaganda por tempo de exposição, não é algo garantido nem para mais nem para menos, e uma das métricas consegue contabilizar isso.

1. Escopo do projeto

O principal objetivo é a extração das métricas, e a visualização, sendo as principais métricas o tempo de exposição, a área de exposição e a exposição pelo tempo, para tornar essas informações mais legíveis, utilização de recursos gráficos como gráficos, mapas de calor dão valor ao produto.

O projeto tem 2 marcos, desenvolver um programa que consegue identificar uma logo em um vídeo e retirar métricas de exposição, tempo e posicionamento, e fazer esse *software* ter a estrutura necessária para se tornar escalável e de fácil uso como uma API.

2. Desafios

Alguns obstáculos do projeto, como, processamento de um vídeo, todo o projeto ocorre a volta de algum vídeo de eventos esportivos, mas essa precisa de alguns cuidados, como, a formatação que o vídeo deve ter (ex.: mp4, avi, mkv). E o principal, o CODEC (compressão) do vídeo, isso se deve dado que alguns métodos que comprimem vídeos fazem cortes em informações importantes para salvar espaço, portanto necessita-se de um CODEC que consiga entregar todos os frames por inteiro ao programa, alguns acabam entrando apenas atualização do frame anterior, e isso pode causar muitos empecilhos já que o frame teria que ser montado.

Definição das métricas, existem as métricas quais são critérios do projeto, e as métricas que se deve ter para medir acurácia, performance entre outros dados para trazer confiança que o software traz um resultado consistente e aceitável.

Detecção na imagem, principal desafio do projeto, como vai ser feito a detecção e o tracking da logo durante o vídeo. Qual algoritmo será necessário e o melhor para a necessidade, e quais tratamentos nos dados e imagens devem ser feitos.

Disponibilidade do software, deve-se definir qual a melhor ou mais prática maneira de ser ofertado e de ser usado.

E o *benchmark*, temos que verificar a acurácia do software, comparando-o com uma referência que sabemos que está correta.

‘

3. Recursos

Lista de recursos, serviços e ferramentas que foram necessárias para desenvolver o projeto, e uma breve descrição do seu funcionamento ou utilidade, repositório para organização e versionamento de software, a BitBucket, escolha da Dell, é uma plataforma para armazenamento, gerenciamento e versionamento de software, dando várias ferramentas para grupos trabalharem juntos no mesmo projeto com facilidade em resolver conflitos de desenvolvimento.

Para organização interna do time Kanban Board, Notion provê um site customizável e com vários *templates* de páginas prontos, incluindo de organização em Kanban Board, usado para gerenciar pipelines de projetos de software.

Python é uma linguagem em crescimento, mundialmente usada para processamento de imagens[1] e junto a ela, python jupyter notebook que provê uma interface que permite executar códigos em python em células separadas, porém pertencentes ao mesmo kernel.

Biblioteca de processamento de imagem, OpenCV[2] com Algoritmo de *feature matching*, o SIFT, significa *Scale-Invariant Feature Transform* (Transformação de recurso invariável de escala), criado e patenteado por David Lowe, professor em ciência da computação na universidade de British Columbia e na Universidade de Nova Iorque. Biblioteca de processamento de imagem, OpenCV[32] com Algoritmo de *feature matching*, o SIFT, significa *Scale-Invariant Feature Transform* (Transformação de recurso invariável de escala), criado e patenteado por David Lowe, professor em ciência da computação na universidade de British Columbia e na Universidade de Nova Iorque. SIFT é um algoritmo que detecta, descreve e compara os recursos locais de uma imagem, tem a funcionalidade de reconhecer objetos e/ou formas, navegação e mapeamento robótico, para o projeto interessa a função de reconhecimento. Como sugestivo pelo nome, tem a funcionalidades de ser invariantes com relação a escala e a rotação da imagem, pode-se destrinchar esse processo em 4 partes, sendo elas: Detecção de extremos, localização de ponto-chave, atribuição de orientação e descritor de ponto-chave (*keypoints*). Para a detecção de espaço e escala um objeto só pode ser definido se estivermos olhando pela escala certa, esse processo tem como objetivo dimensionar a imagem que estamos analisando. A função gaussiana de espaço-escala da imagem é criada a partir da convolução de uma escala variável, essa tem o efeito de borrar a imagem. Com as imagens em diversas escalas podemos fazer a diferença gaussiana de cada ponto na imagem original e nas escaladas. Após gerar as diferenças gaussianas, é preciso detectar extremos locais que são possíveis ponto-chave, a partir de um pixel da imagem e seus vizinhos são comparados com seus respectivos locais nas outras imagens, se a escala se mantiver esses pixels são candidatos a ponto-chave.

---

[1] Top 3 Programming Languages For Implementing a Computer Vision System
[2] https://opencv.org/

‘

Após localizar os ponto-chave é possível observar que seu número é bem elevado, sendo muitos deles nos cantos da imagem e em pontos com pouco contraste. Usando a série de Taylor para retirar os pontos de contraste, é sugerido 0.03. Para os cantos utiliza-se uma matriz Hessiana 2x2 para computar a curvatura principal.

Com a invariância de escala resolvida é necessário agora dar uma orientação para cada ponto-chave para que ele seja também invariante em rotação. Pegando uma área vizinha ao ponto-chave da escala, cria-se um histograma de orientação de 36 partes para cobrir os 360° envolta do ponto-chave, ponderando pela magnitude e pela ponderada gaussiana do ponto-chave de escala com a proporção de 1.5 vezes. Os picos no histograma correspondem a direção dominante do gradiente local, picos com mais de 80% do ponto-chave de escala se torna um ponto-chave dessa orientação.

Posteriormente, o descritor é criado pegando uma vizinhança de 16x16 em torno do ponto-chave e o divide em blocos de 4x4. Esses 128 valores descrevem o ponto-chave por um vetor, ele guarda esse grande número de informações para dar robustez contra mudanças de rotação e iluminação.

Ao comparar ponto-chave de imagens é possível ocorrer uma segunda correspondência devido a uma similaridade entre os ponto-chave. Para esses casos é calculado uma razão entre a distância mais próxima e a segunda mais próxima, se for maior que a proporção 0.8 o ponto é rejeitado, isso reduz em 90% das falsas correspondências (LOWE, 2004, p. 20).

Algoritmo de *feature matching*, *flann knn*, como descrito anteriormente, o SIFT calcula pontos que podem ser considerados pontos chave em uma imagem que chamam *features*, e em ordem de achar padrões nesses pontos que caracterizaria áreas parecidas, chamada de correspondências ou *matches*, usando o *flann knn* que utiliza as *features* e faz uma comparação entre dois conjuntos delas usando distância euclidiana, e de acordo com um limite *threshold*) dessa distância, ele considera algo como match ou não.

Servidor da API, também feito em python, mas com a biblioteca FastAPI, servidor responsável por executar o software desenvolvido no projeto e salvar o resultado no banco de dados, também receber requisições do cliente.

Banco de dados para vídeos e imagens, AWS S3, utilizado para armazenamento de arquivos volumosos e que não são editados, apenas adicionados ou apagados.

MongoDB, um banco de dados não relacional para armazenamento de dados e resultados das iterações, e para a comunicação das partes, Prisma ORM facilita a integração com o banco de dados, principalmente por ser uma biblioteca compatível com python.

4. Cronograma

A Figura 1 é o fluxo de desenvolvimento do projeto, divido em 3 macro áreas localizadas a esquerda em quadrados. A superior, processamento de imagem, *Frame Id* é quebra dos frames de um vídeo, montagem do software para executar essa tarefa automaticamente. *Detecção em um frame* se refere a conseguir identificar um logo em um frame apenas. *Rastreamento no vídeo* é fazer *n* interações identificando os logos em cada frame, onde *n* é a quantidade de frames. *Múltiplos logos* capacitar o software a fazer identificação de mais de

‘

um logo na mesma execução. *Refaturamentos e melhorias* foco em melhorar a performance e velocidade de execução.

A sessão central da Figura 1, *Extração de métricas*, se divide em *Métricas de um frame* definir e desenvolver quais dados que devem ser calculadas a cada frames, área, proporção da tela e contorno de posição, usado para o mapa de calor.

Métricas do vídeo relaciona a juntar as métricas de cada frame em uma geral para o vídeo (exposição média na tela, montar o mapa de calor de posicionamento na tela), somando a isso, métricas que dependem de vários, como o tempo que a logo ficou exposta.

Métricas de erro que se predispõem em desenvolver e implementar métricas para os erros do match achado, então diferenças no histograma de cores e a razão de *matches* com a quantidade de *keypoints* são métricas de foram desenvolvidas nessa parte.

Cálculo de acurácia usar dados confiáveis e bater com o resultado do software, descobrindo a porcentagem de acerto.

E a sessão de baixo, *Serviço API* que tem apenas 2 sessões, *Infraestrutura* que se preocupa com a preparação da infraestrutura necessária para a API, incluindo, mas não preso à, servidor cliente para receber requisições, servidor responsável pelo tracking, banco de dados para armazenamento de resultados e banco para armazenamento de dados volumosos. E a segunda sessão *Adaptação do código para API* são ajustes necessários, para um código que antes era na máquina do programador, poder funcionar em um servidor, com conexão de banco de dados e serviços externos (AWS, Bucket, etc.).

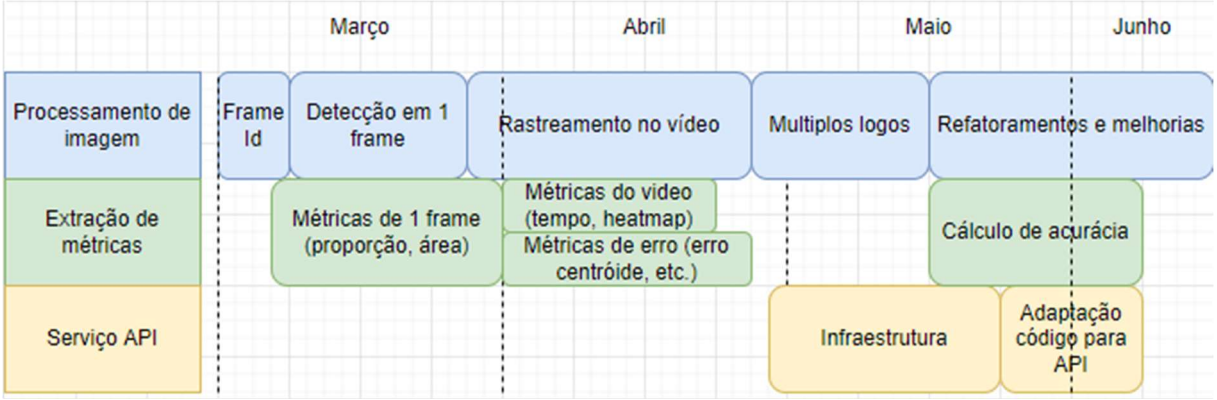

Figura 1. Cronograma de desenvolvimento

5. Mapeamento dos stakeholders

Stakeholder	Posição	Papel no Projeto
João Meirelles	Desenvolvedor	Relatório e programador
Gabriel Noal	Desenvolvedor	Programador
Jose Fernando	Desenvolvedor	Estudos e programador

‘

Diego Gallo	PO Dell	Dono do projeto

6. Revisão do Estado da Arte

Identificações de logos em fotos de *marketplaces*. A Amazon por possuir um catálogo enorme, e diariamente serem feitos upload de milhares de fotos de produtos, desenvolveu um software que usando visão computacional e *machine learning*, ensinou a identificar possíveis logos nessas fotos. (Fehervari & Reddy, 2019).

Uma comparação entre os algoritmos de *feature* concluiu que o SIFT possui maior acurácia, com o ORB e o SURF sendo melhores em casos específicos. (Karami at al, 2015).

Uma comparação entre os algoritmos incluindo o BRISK, AKAZE e KAZE, concluiu também que o SIFT tem uma acurácia melhor em escala e rotação em transformações multidimensionais (em mais de 1 eixo). (Tareen & Saleem, 2018)

Projeto que gera relatórios para marcas sobre sua exposição usando puramente Rede Neural (Romaniuk et al, 2019).

7. Desenvolvimento

a. Estrutura de organização da execução do projeto

A metodologia *Scrum*, com sprints semanais, um dos métodos ágeis mais conhecidos, e amplamente empregado no mundo do desenvolvimento e tecnologia por ser mais focado em entregas pequenas e de baixa complexidade, com o objetivo de no final gerar um resultado juntas. O *Scrum* se baseia em uma janela, comumente semanais, de tempo onde metas são estabelecidas, essas decidias pelo time, e não por um gerente ou chefe, deixando portanto, o tempo de desenvolvimento nas mãos de quem vai executa-lo, somando a isso, ocorrem sessões semanais, às terças-feiras, entre o grupo e o orientador para tomarem decisões mais estratégias e técnicas, enquanto as sextas ocorrerem as reuniões de finalização da *sprint*, onde se reunião todos *Stakeholders* e técnicos de visão computacional da Dell para apresentar os avanços feitos, e discutir próximos passos. Ambas as reuniões são registradas em atas de reuniões que podem ser encontradas no *Notion*, site usado para organização de documentos e tarefas, do grupo junto com documentos e as tarefas que foram feitas.

b. Desenvolvimento do processamento de imagem

O *software* de processamento precisa de imagens que possam ser processadas, para isso um se desenvolver outro *software* mais simples que faz o trabalho de percorrer todos os frames de um vídeo e salvá-los em ordem em um diretório. Esse programa usa a própria biblioteca OpenCV que tem métodos de leitura de vídeos.

‘

Com os frames separadas, a parte externa ao processamento está pronta, agora internamente ao software, algumas coisas precisam ser definidas, são essas, para cada logo que deve ser encontrada, se aplica o SIFT para extrair as *features* de cada uma, e a segunda a ser, o software depende de um parâmetro, para considerar se algo que ele considera correspondência realmente deve ser considerado uma, o número mínimo de correspondências, e isso deve ser dinâmico, ou seja, variar para cada logo, já que que a quantidade de *features* que cada logo vai ter varia muito, e essas que vão ter correspondências então a quantidade máxima seria a quantidade de *features*, então o número mínimo deve ser proporcional.

Com as preparações feitas, o programa segue um processo de 9 passos, que serão repetidos para cada frame que foi extraído, os passos se constituem em:

1. carregar o frame do vídeo que será processado.

2. usar o SIFT para pegar as *features* (pontos de interesse) na logo que desejamos encontrar.

3. usar o SIFT no frame extraído para buscar *features*.

4. usar *flan knn* para achar correspondência entre as *features* do Frame e do logo.

5. Dado que a quantidade de correspondências seja maior que o mínimo necessário (que foi calculado previamente), segue os passos, caso contrário, termina essa iteração e volta ao passo 1 com o próximo frame.

6. Achar o contorno do logo (se faz isso buscando pegar a logo base, e aplicando transformações para que ela se encaixe nos *keypoints* da logo encontrada).

7. Calcular métricas baseados nessa correspondência e no contorno.

8. Pintar a logo encontrada de branco, para permitir que outras sejam encontradas

9. Repetir o passo 4 até não achar mais correspondências, então voltar ao passo 1 com o frame seguinte.

8. Desenvolvimento e coleta e análise de dados

Métricas que procuram se complementar foram pensadas, dado que cada uma delas pode ter incapacidades em casos diferentes, são essas, comparar duas histogramas de cores buscando similaridades na variação nas cores e nas quantidades que elas aparecem. O OpenCV possui um método para criar um histograma de cores (usando cada canal de cor separadamente *RGB*), para compará-los necessita-se ser baseados em algum método, o escolhido foi *Hellinger distance,* que quantifica a similaridades dentre duas distribuições (usando o quadrado da diferença das derivadas das duas como cálculo principal), mostrado
‘

graficamente pelo Figura 2, onde se tem duas curvas para cada espectro de cor, sendo uma do logo benchmark e outro do encontrado.

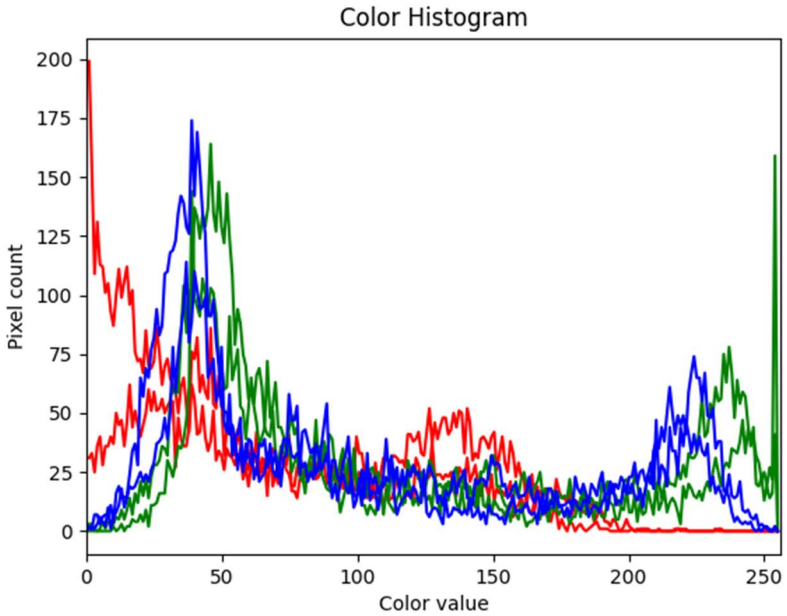

Figura 2.: Histograma de cores dos canais *RGB*

Outra métrica é razão entre pontos usados e existentes, o SIFT provê uma quantidade de *features*, então a razão entre quantidade de pontos interessantes que o logo possui (o que é usado como benchmark), e quantos desses pontos foram utilizados para considerar algo no frame como match, exemplificado pela Figura 3, onde *keypoints* se refere aos da logo que está sendo buscada e os *matches* quantos desses *keypoints* foram usados para achar um *match*, assim, quão maior for essa razão maior a chance do logo ser a mesma, obedecendo a função:

$$razão = \frac{Nf}{Nm}$$

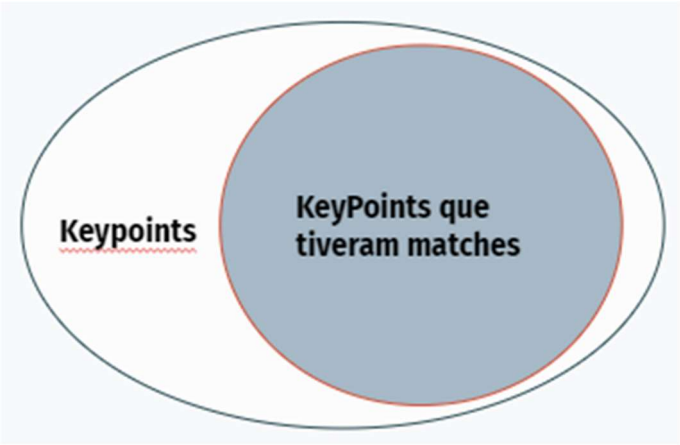

Figura 3.: Razão entre *keypoints* totais e matches.

‘

Sendo, *Nf* (Número de *features* da logo original) e *Nm* (Número de *matches* encontrados).

Porcentagem da área dos logos presentes no frame em relação ao todo da tela, métrica que consiste em somar as áreas dos logos presentes no frame e dividi-los pela área total:

$$\%\ da\ área\ dos\ logos\ = \frac{área\ ocupada\ por\ logos}{área\ total}$$

Tempo de exposição, consiste em pegar a quantidade de frames que houve alguma detecção de logo e dividi-la pela velocidade do vídeo (FPS frames por segundo).

$$Tempo\ = \frac{n}{fps}$$

Sendo, *n* o número de frames que houve detecção de uma ou mais logos.

Mapa de calor da localização agrega visualmente, dado que ele compila em uma imagem algo que acontece por um número grande de tempo ou iterações, assim, o mapa de calor do posicionamento do logo.

Acurácia do *software* para saber se o consegue identificar uma logo com um nível satisfatório de precisão, precisamos de um *benchmark* para poder compará-lo, portanto, foi contado visualmente em quantos frames os logos realmente aparecem, considerando qualquer pedaço mínimo do logo, com esses se calcula o tempo usando a velocidade do vídeo e compara-se esse resultado manual com o resultado do *software*.

9. Desenvolvimento do serviço API

Um dos principais desafios é o da disponibilidade da ferramenta. Existem meios de fornecer e distribuir uma ferramenta, como por exemplo softwares com licenças, softwares livres ou por consumo de requisições via HTTP.

Foi escolhida via API, principalmente pela capacidade de ser utilizada por qualquer serviço sem exigir dependências, apenas sabendo as rotas existentes (disponíveis na documentação) e o que passar pela requisição, você consegue fazer uso completo da solução desenvolvida.

Com esse objetivo em mente, foi montada a infraestrutura representada pela Figura 4. Montada inteiramente nos serviços web da Amazon, e consiste em um banco de dados (*MongoDB*) para armazenamento dos resultados, um *bucket* de arquivos do S3 (Serviço da AWS) para armazenamento de dados maiores como o vídeo a ser processado e armazenar os frames que são extraídos e um servidor de processamento, montado com a biblioteca *FastAPI* do python, fica responsável pelo processamento do vídeo.

‘

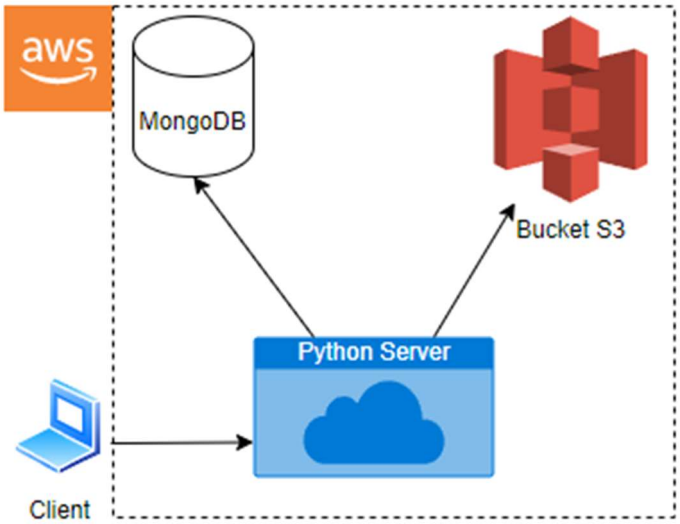

Figura 4.:Diagrama de infraestrutura da API

10. Coleta de dados sobre o projeto

Foram usados 2 tipos de métricas, as requeridas pelo cliente, e as de qualificação do próprio software, como apresentado na introdução, e na sessão 1.7.2, os gráficos que se seguirão, para cada caso, um em ordem crescente e outros em ordem dos frames, essa diferença procura mostrar diferenças conforme os frames vão se distanciando do inicial, o qual é retirada a logo que será usada para treino, pode aparentar alguma dificuldade maior do algoritmo de fazer seu trabalho, os dados completos foram colocados em um Google Sheets.

A Tabela 1 e os Figuras 5 e 6 são referentes a métrica de razão das correspondências das *features* com a quantidade delas na logo base.

Razão *Matches* / *KeyPoints*				
Mediana	Média	Máxima	Mínima	Desv. Padrão
70.29%	69.406%	95.47%	52.45%	9.112%

Tabela 1.: Resultados das razões de *Keypoints* e *Matches*

‘

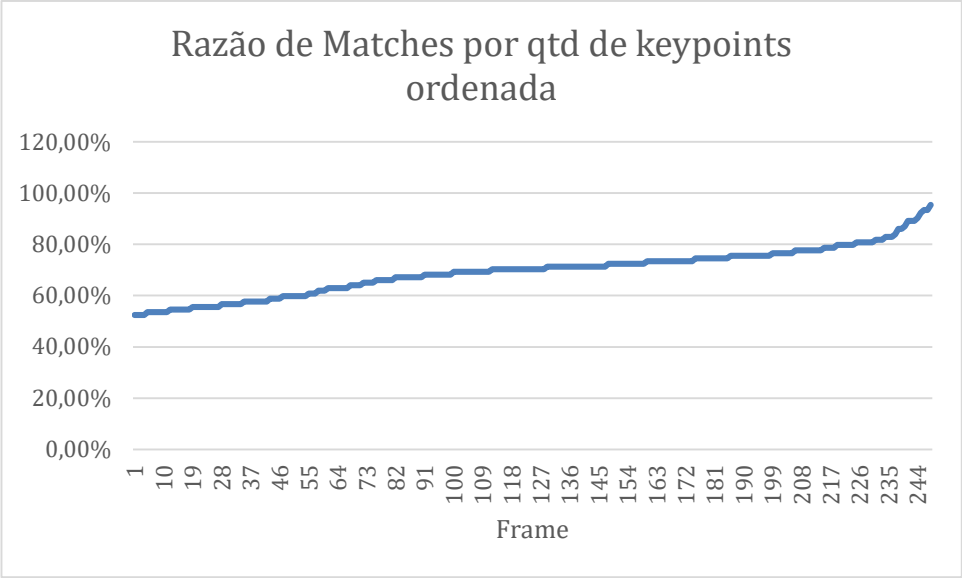

Figura 5.: Razão com Ordenação crescente

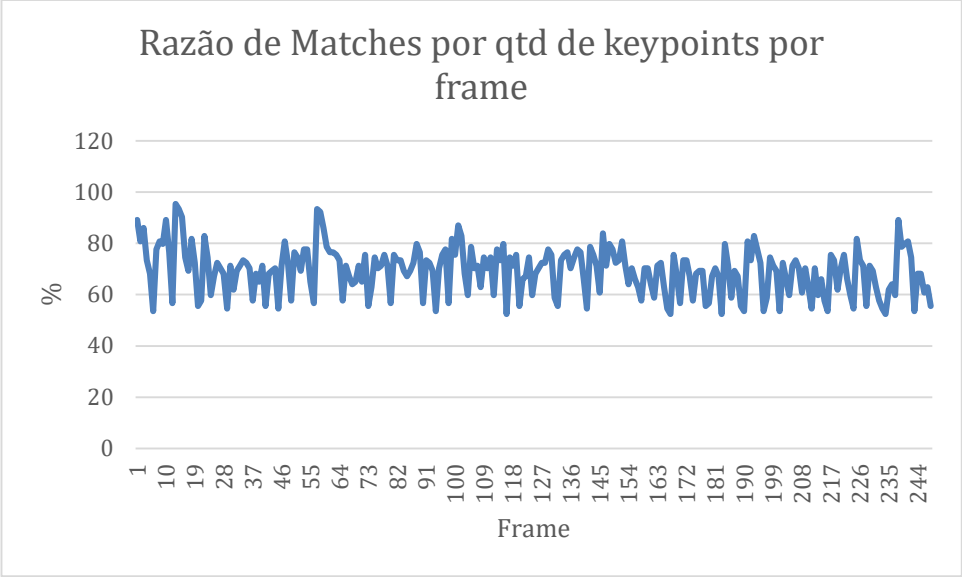

Figura 6.: Razão ordenado por frame

A Tabela 2 e os Figuras 7 e 8, são referentes a comparação do histograma de cores dada pela distância de *Hellinger*

Histograma de cores				
Mediana	Média	Máxima	Mínima	Desv. Padrão
98.42%	98.63%	100%	97.74%	0.59%

Tabela 2.: Resultados da comparação dos histogramas de cores

‘

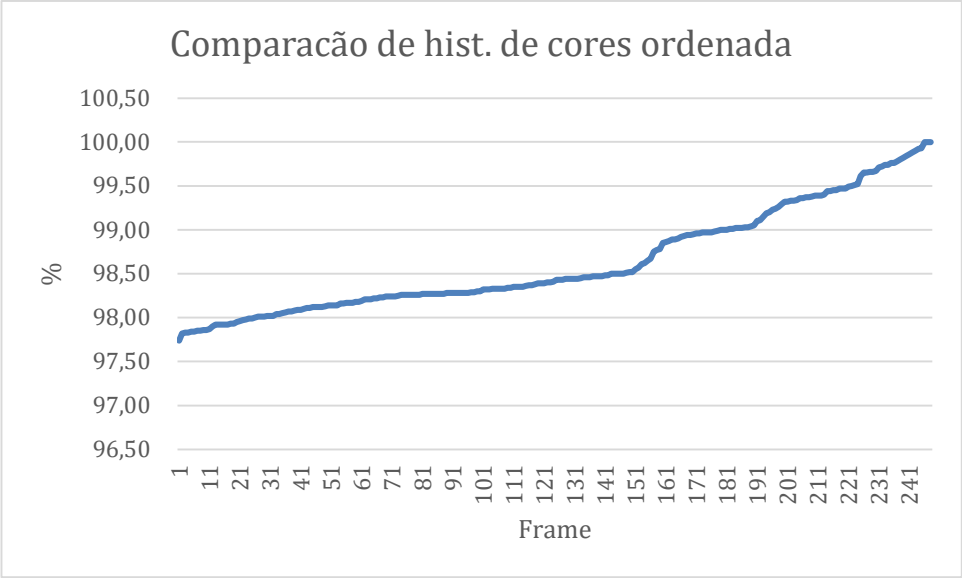

Figura 7.: Comparação de histograma de cores por ordem crescente

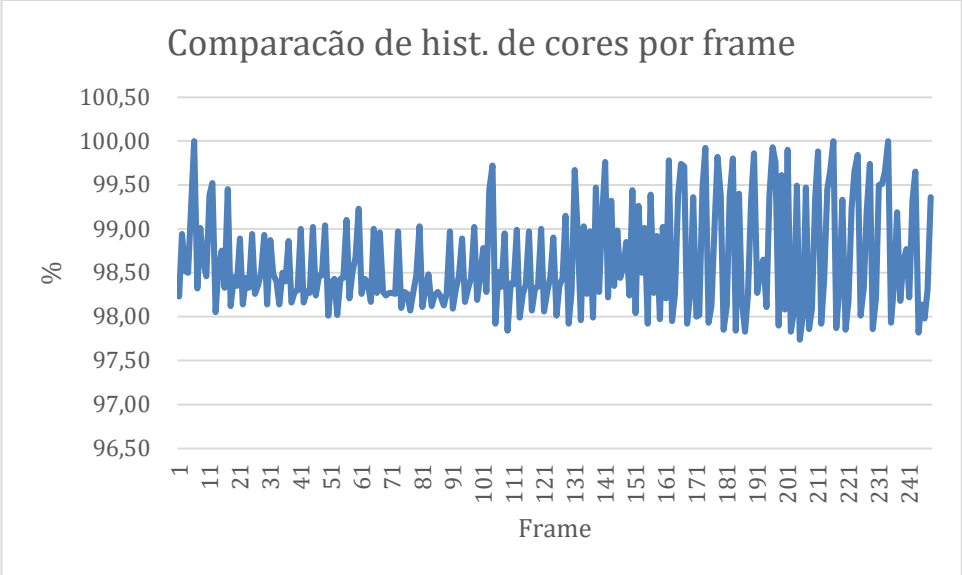

Figura 8.: Comparação de histograma de cores por ordem de frames

A tabela 3 e os Gráficos 9 e 10 são referentes a métrica de porcentagem da área que os logos presentes no frame ocupam.

Distância de *Hellinger*				
Mediana	Média	Máxima	Mínima	Desv. Padrão
0.29%	0.2898%	0.59%	0.13%	0.0818%

Tabela 3.: Resultados da distância de *Hellinger*

‘

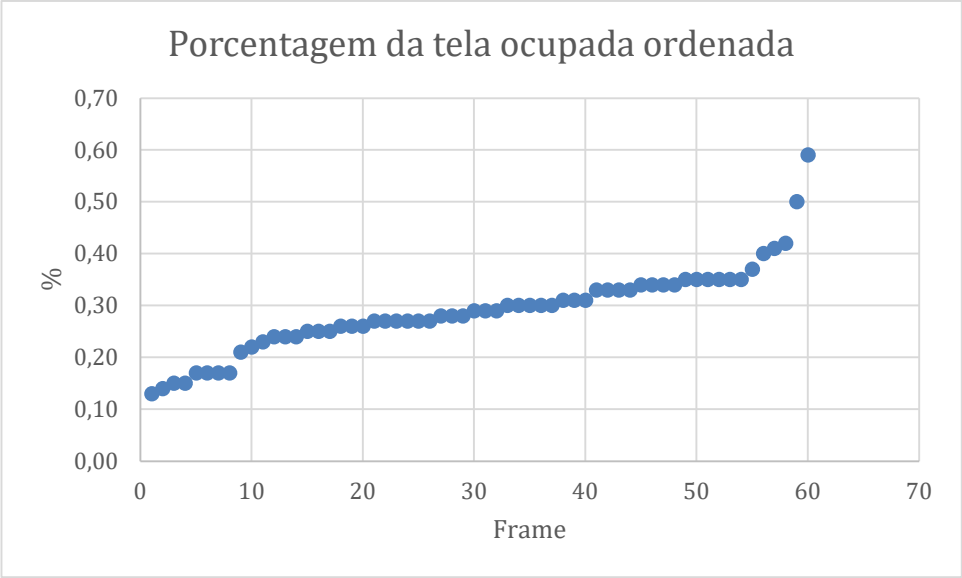

Figura 9.: Porcentagem da tela ocupada por ordem crescente

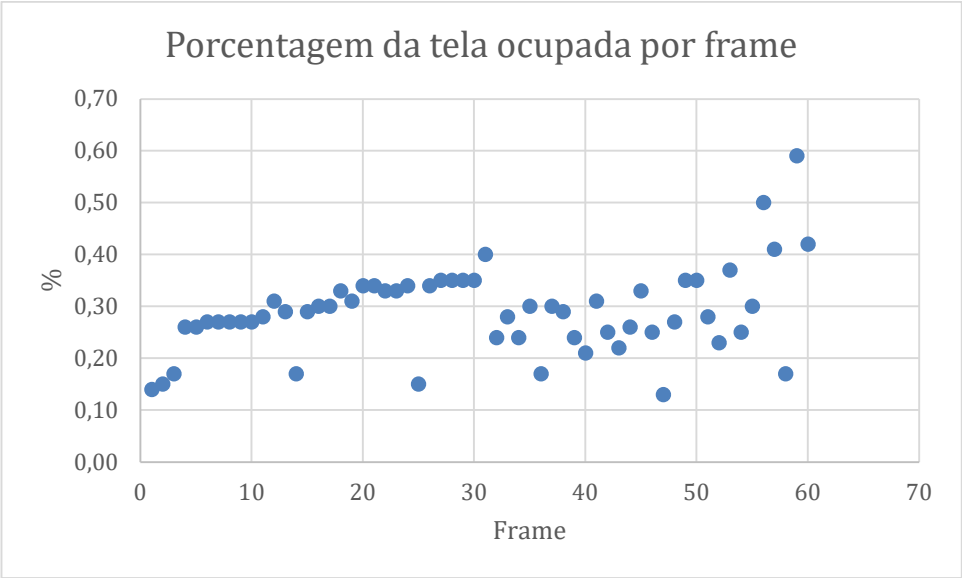

Figura 10.: Porcentagem da tela ocupada por ordem crescente

11. Análise dos Dados Coletados

Com as Figuras 6, 8 e 10, podemos ver que a partir de um certo frame, o comportamento mais ao final fica mais instável. Por exemplo, no Figura 11, fica bem aparente uma dispersão maior a partir do frame 30, assim como observado também na Figura 6 aproximadamente no frame 229 e na Figura 8 a partir do frame 131. Esse problema pode ser minimizado atualizando a referência, logo que é usado como base para identificação, dado que conforme o vídeo passa.

A comparação do histograma de cores parece uma métrica ótima, mas ela pode causar dúvidas em casos, principalmente em logos que dependem mais do formato do que a cor. A usada nos testes, *Binance*, possui logos com fundo branco, e logos com fundo preto, mas a escrita sempre aparece em amarelo, já a Dell por exemplo, o logo aparece em azul, as vezes

em prateado, fazendo com que, em casos, a comparação de histograma de cores seja desinforma.

Figura 11 mostra um caso em que a logo do *Tim Hortons* está sendo obstruída, mas ainda foi identificada, esse caso a comparação de histograma de cores seria muito diferente, principalmente pelo fato do *software* procura encaixar o logo *benchmark* baseado nos *matches* dentro do frame, isso feito para extrair um contorno real.

Figura 11.: Negativo do contorno do logo

A Figura 6 mostra que a razão se manteve estável, deixando o entendimento que a quantidade de *keypoints* achados para montar o match, não foi afetado.

12. Resultados processamento de imagem

O programa que consegue identificar as com uma boa precisão, como mostrado na coleta de dados, a Figura 12 é o frame que será usado para buscar logos, a Figura 13 é a logo usada para treino, a qual foi retirada do frame inicial, na Figura 12 tem-se as Figuras 12 e 13 juntas e as linhas verdes representam os matches encontrados, conectando *features* que foram consideradas semelhantes.

Figura 12.: Frame do vídeo

‘

Figura 13.: Logo de *benchmark*

Figura 14.: Figura mostrando conexões dos *keypoints* com seus respectivos *matching*

A Figura 14 demonstra o polígono que é feito contornando a logo encontrada, esse é usado para ser preenchido de branco, como dito no passo 6 na sessão de metodologia, e para calcular a área dessa logo.

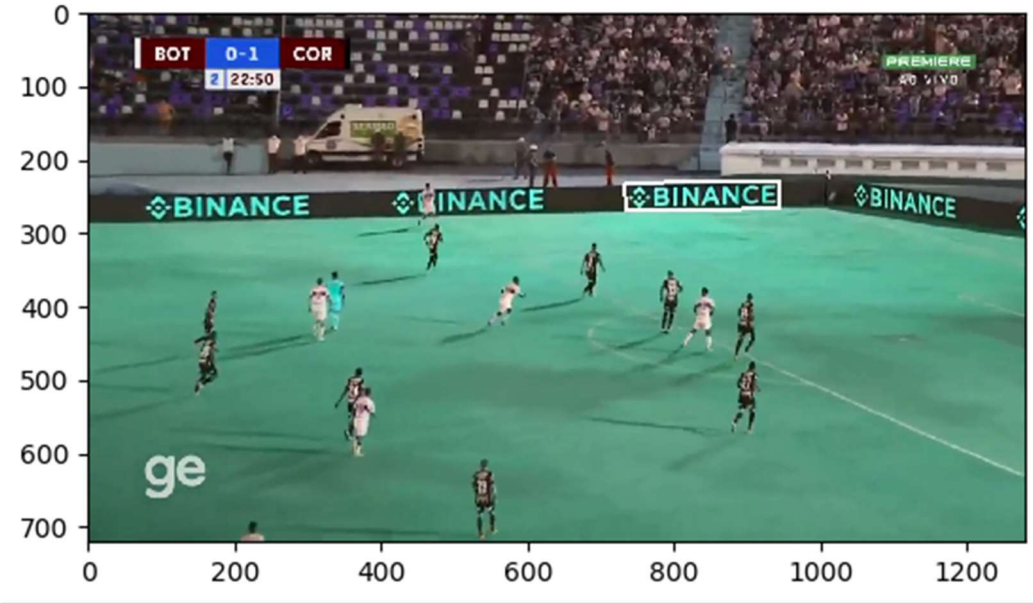

Figura 15.: Identificação de uma logo

Quando achado o logo, ele o obstrui pintando de branco, assim, na próxima interação não vai achar ele, permitindo que o próximo logo seja encontrado. Na Figura 15 já foi identificado um logo como o da Figura 13 e obstruído. Na Figura 17, pode-se identificar a interação em cima dos logos aparentes, fazendo os mesmos passos em múltiplas aparições.

‘

Figura 16.: Após removido uma logo, identificação de outra diferente

Figura 17.: Após removido 2 logo, identificação de outra diferente

13. Benchmark

Não basta apenas gerar erros baseados nos dados obtidos, isso não garante a acurácia do software. Para calcular isso precisamos de dados confiáveis de que estão corretos, e assim, pegar os mesmos vídeos que foram usados para pegar os dados confiáveis, e aplicar o software neles e comparar os resultados.

A obtenção dos dados confiáveis de tempo de exposição consistiu em quebrar o vídeo em frames, e manualmente ir olhando os primeiros 300 frames para ver onde a logo aparecia, foi considerado qualquer pedaço do logo, sendo apenas o símbolo (que aparece muitas vezes ao lado do nome escrito), ou até a última letra, as Figuras 18, 19 mostram casos que o software não conseguiu, sendo a Figura 18 pouca parte da logo aparecendo e a Figura 19 em uma transição de câmeras, onde a Logo foi aparecendo através de um efeito *fade*, a Figura 20 onde há uma identificação aconteceu exatamente um frame após a Figura 19.

‘

Figura 18.: Logo cortado não identificado com destaque

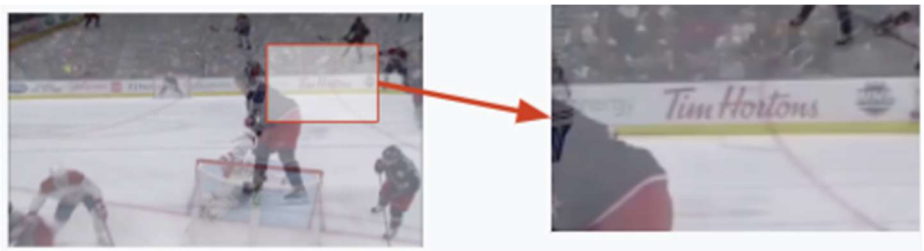

Figura 19.: Logo em transição de câmera não identificado com destaque

Figura 20.: Logo em transição de câmera identificado com destaque

A Tabela 4 corresponde a diferença em porcentagem dos resultados do *software* e do benchmark criado, a Tabela 5 traz dados gerais sobre os dados da Tabela 4, essas diferenças respeitam à fórmula:

$$dif = 100 * (1 - \frac{Tempo\ Software}{Tempo\ Benchmark})$$

Nome vídeo	**diferença (%)**
Baseball	-0.81
Convocação	0
Futebol 1	12.98
Futebol 2	21.81
Hockey	38.46
Seleção Fem	6
MotoGP	28.29
Super Truck	23.35

Tabela 4.: Resultados acurácia

Média	17.4
Desvio padrão	14.09

‘

Maior	38.46
Menor	-0.81

Tabela 5.: Resumo acurácia

14. Resultados API

A API consegue prover dados de duas formas. Foi desenvolvido um arquivo python *Notebook* para usar de referência, caso tenham preferencias por receber dados mais "crus" e fazer sua própria ciência dos dados. A Figura 21 exempla um arquivo JSON desses dados, esse método é necessário o próprio cliente fazer as quebras dos Frames, e mandar cada um desses para a API para ela fazer a localização e extração das métricas "cruas".


```
[
  {
    "area": 6359.5,
    "points": [
      [[153.17343139648438, 222.5863494873047]],
      [[154.8256072998047, 252.0750274658203]],
      [[338.43048095703125, 252.60260009765625]],
      [[338.89239501953125, 223.90753173828125]]
    ],
    "contours": [
      [[245,220]],
      [[338,221]],
      [[340,223]],
      [[340,252]],
      [[338,254]],
      [[154,254]],
      [[152,252]],
      [[151,236]],
      [[151,222]],
      [[153,220]]
    ],
    "exposure": 0.006900499131944444,
    "matchPercentage": 0.005763319672131148,
    "centroideErr": 273.86245053928366,
    "colorHistogramComparison": 98.91746430355128
  }
]
```


Figura 21.: Exemplo de retorno da API para 1 frame

15. Conclusões e trabalhos futuros

O projeto atingiu o que foi esperado pelo cronograma, sendo esse, um *software* capaz de identificar 82.6% verdadeiros positivos em média, e raramente identifica um falso positivo (isso é importante pois falsos positivos seria identificação onde não existe), a solução desenvolvida gera métricas consistentes e que trazem valor, e conseguiu prover uma

‘

infraestrutura que permite que ser implementado de uma maneira fácil e simples (Uso de API é mundialmente usada).

Próximos passos são:

Melhorar performance aplicando mais dinamismo nos parâmetros do SIFT, durante a segunda parte do semestre focamos na API e não tivemos tempo para melhorar o SIFT. Paralelizar processamento de imagens em caso do processamento de vídeo completo para melhorar a performance. Implementar um sistema de múltiplos logos mais completo, com separação de dados entre eles, para facilitar a comparação entre diferentes logos. E criar uma interface *user friendly,* atualmente é preciso um conhecimento de programação para trabalhar com a API.

Referências

1. Noema, Yaniv. Top 3 Programming Languages For Implementing a Computer Vision System. Dev.to, 2021. Disponível em:https://dev.to/imagescv/top-3-programming-languages-for-implementing-a-computer-vision-system-4jk5#:~:text=As%20one%20of%20the%20most,based%20tasks%20at%20high%20speeds. Acesso em: 06/06/2022
2. Fehervari, Istvan. Reddy, Arun. Detecting Brand Logos at Scale Using Computer Vision and Machine Learning. Youtube.com, 2019. Disponível em:https://www.youtube.com/watch?v=6tDoxaDzC3E Acesso em: 06/06/2022
3. Karami, Ebrahim. Prasad, Siva. Shehata, Mohamed. Image Matching Using SIFT, SURF, BRIEF and ORB: Performance Comparison for Distorted Images, 2015. Disponível em: https://www.researchgate.net/publication/318194587_Image_Matching_Using_SIFT_SURF_BRIEF_and_ORB_Performance_Comparison_for_Distorted_Images Acesso em: 06/06/2022
4. Romaniuk ,Michal. Budek, Konrad. _Romaniuk ,Michal. Budek, Konrad. Logo detection and brand visibility analytics, 2019. Disponível em: https://deepsense.ai/logo-detection-and-brand-visibility-analytics-example/ Acesso em: 06/06/2022

‘

5. https://docs.google.com/spreadsheets/d/1QmLzZNNIMBeWeaYzXU0N1InyFPP2jHphnsGWQ9TSUBA/edit?usp=sharing
6. Tareen, Shaharyar. Saleem, Zahra. A comparative analysis of SIFT, SURF, KAZE, AKAZE, ORB, and BRISK, 2018. Disponível em: https://ieeexplore.ieee.org/abstract/document/8346440, Acesso em: 06/06/2022
7. Mattos, Rodrigo. Futebol movimenta R$ 53 bi na economia do Brasil, mas só gera 1% de imposto..., 2019. Disponível em: https://rodrigomattos.blogosfera.uol.com.br/2019/12/13/futebol-movimenta-r-53-bi-na-economia-do-brasil-mas-so-gera-1-de-imposto/, Acesso em: 06/06/2022.
8. Lowe, David G. Distinctive Image Featuresfrom Scale-Invariant Keypoints, 2004. Disponível em: https://www.cs.ubc.ca/~lowe/papers/ijcv04.pdf , Acesso em: 06/06/2022

‘