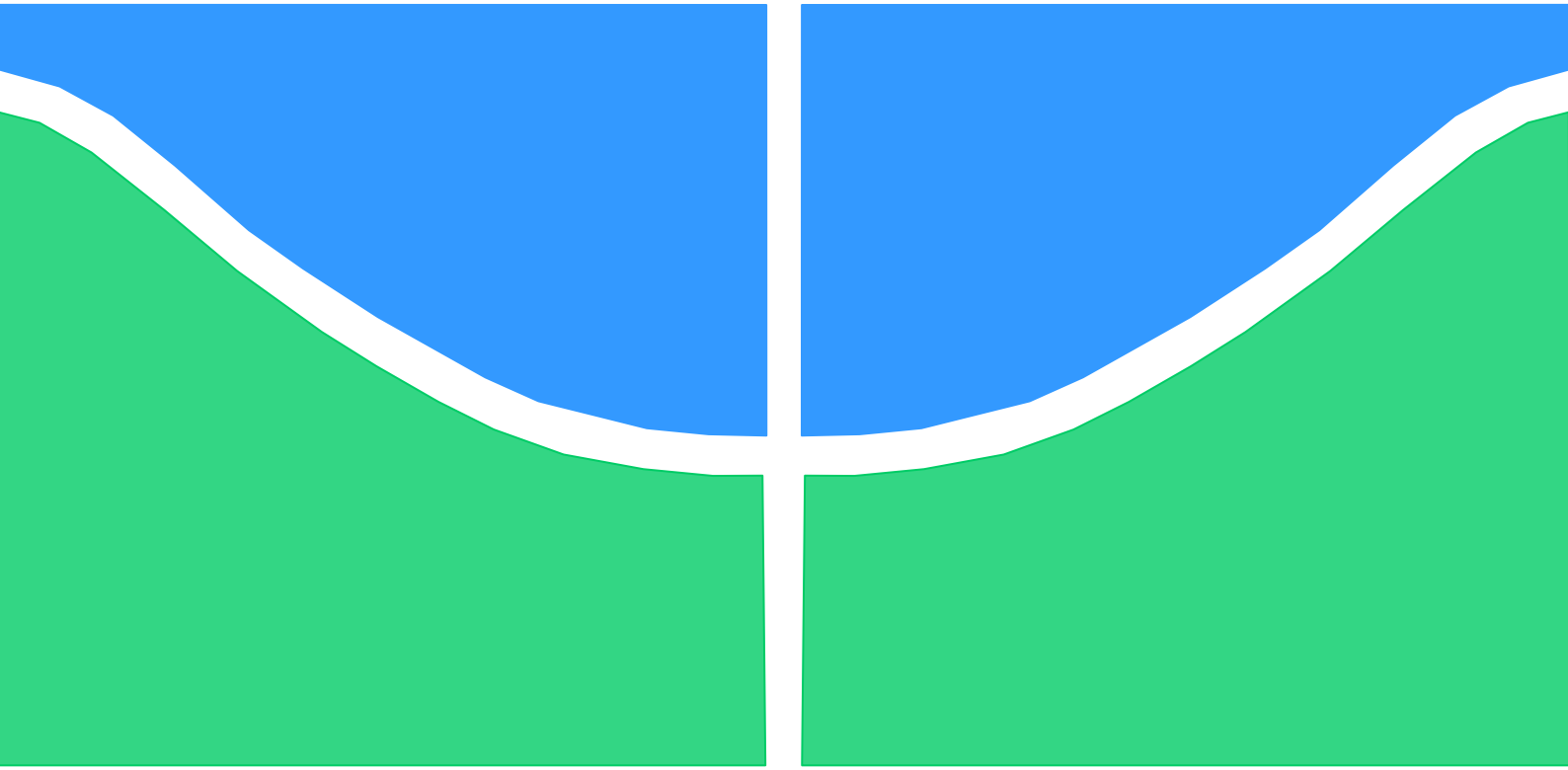

TRABALHO DE CONCLUSÃO DE CURSO 2

ANÁLISE DE MODELOS DE SALIÊNCIA PARA VÍDEOS 360°

Bernardo Magalhães Morales

Brasília, julho de 2019

UNIVERSIDADE DE BRASILIA

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASILIA
Faculdade de Tecnologia

TRABALHO DE CONCLUSÃO DE CURSO 2

Análise de Modelos de Saliência Para Vídeos 360°

Bernardo Magalhães Morales

Relatório submetido como requisito parcial para obtenção
do grau de Engenheiro Eletricista

Banca Examinadora

Profª. Mylène C. Q. de Farias, ENE/UnB
Orientadora

Prof. João Luiz Azevedo de Carvalho, ENE/UnB
Examinador interno

Lucas dos Santos Althoff, UnB/CIC
Examinador interno

Dedicatória

Dedico esse trabalho à minha família. Principalmente aos meus pais, avós, meu irmão e minha namorada que sempre me apoiaram em todas as minhas conquistas

Bernardo Morales

Agradecimentos

Gostaria de agradecer à minha orientadora Mylène, que mesmo morando nos EUA conseguiu me orientar.

Ao GPDS pelo suporte durante o PIBIC e esse trabalho. Aos meus amigos de curso que tornaram as semanas menos chatas.

Aos meus amigos do colégio que nunca deixaram de me apoiar.

A minha terapeuta Ana e meu médico Dr. Mameri.

Principalmente, gostaria de agradecer a minha mãe Maria, meu pai José, meu irmão Danilo, minha namorada Victoria e meus avós Terezinha e Francisco por tudo que já fizeram por mim.

Bernardo Morales

RESUMO

O trabalho aborda o uso do algoritmo de saliência visual GBVS e a rede neural convolucional YOLOv3 aplicados em vídeos 360°. O objetivo é analisar o desempenho de ambos separadamente e avaliar a possibilidade do uso desses algoritmos em um modelo *top-down* de atenção visual aplicado a vídeos 360°. A metodologia utilizada foi submeter um conjunto de vídeos aos algoritmos escolhidos e equacionar o desempenho dos algoritmos a partir de mapas de fixação utilizando a métrica AUC-JUDD. Concluímos que o desempenho desses algoritmos é bom mesmo sem qualquer tipo de compensação.

ABSTRACT

This work discusses the use of the GBVS visual saliency algorithm and the YOLOv3 an object detection algorithm based on a convolutional neural network on 360° videos. The objective is to analyze the performance of both algorithms separately and evaluate the possibility of using them in a top-down model of visual attention applied to 360° videos. The methodology consisted of applying both algorithms to a set of videos and computing their performances using fixation maps (as ground truth) and the AUC_JUDD metric. Results show that the performance of these two algorithms is good, even without any kind of compensation.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

1 INTRODUÇÃO

O campo da neurociência sempre demonstrou enorme interesse em como os animais enxergam. Desde o final do século 19, quando se iniciaram os estudos das estruturas dos olhos dos animais, pesquisas se tornaram cada vez mais abrangentes e serviram de inspiração para o estudo da visão humana, dessa vez não referente aos olhos em si, mas o que ocorre no cérebro. Desde as primeiras décadas do século 20, pesquisadores tentam decifrar os mecanismos por trás da atenção visual. Áreas salientes, conceito inicialmente proposto por James (1890), é definido por Corbetta (1998): "A atenção define a habilidade mental de selecionar estímulos, respostas, memórias ou pensamentos que são relevantes para o comportamento dentre outros que são irrelevantes." Durante sua história, o ser humano foi obrigado a lidar com fluxos cada vez maiores de informações. Seria muito cansativo para o cérebro ter que lidar com todos os estímulos de uma só vez. O cérebro é obrigado a priorizar, por isso que mecanismos evolutivos fizeram o ser humano desenvolver atenção visual seletiva (Judd, 2011).

No entanto, a atenção não está relacionada somente ao que os olhos veem. Yarbus (1967) mostrou que a atenção visual de um indivíduo é fortemente influenciada pela tarefa que esse indivíduo realiza. Isso ocorre como consequência da interação de dois mecanismos principais de atenção visual: o *bottom-up* e o *top-down*. Mecanismos *bottom-up* são automáticos e baseados em estímulos. Em contrapartida, mecanismos top-down são estímulos influenciados pela expectativa, conhecimento e objetivos pessoais (Judd, 2011).

O interesse em projetar algoritmos inspirados em aspectos biológicos aumentou com a evolução das áreas de robótica, visão computacional, inteligência artificial, dentre outras. A partir disso, surgiram algoritmos especializados em modelar atenção visual para que, por meio deles, possa-se melhorar os sistemas de visão computacional ou até encontrar melhores algoritmos de compressão.

Vale salientar que algoritmos de atenção visual mais eficientes poderiam auxiliar no processo de compressão/transmissão de vídeos na internet, por exemplo, juntamente com os algoritmos de codificação de fonte. Idealmente, seria possível identificar com precisão quais pontos da imagem de um filme são mais relevantes para os mecanismos de atenção. Esses pontos seriam transmitidos com uma resolução maior do que pontos menos importantes e, assim, utilizar a banda de transmissão com mais eficiência sem comprometer a experiência do usuário. No entanto, grande parte dos modelos de atenção visual são baseados em fatores *bottom-up*, como referido acima. Modelos baseados nesses fatores costumam ser mais simples (Judd, 2011). Preferencialmente, modelos de atenção visual devem unir fatores *bottom-up* e *top-down* se pretenderem se aproximar mais da visão humana.

Juntamente com o avanço da visão computacional, houve um aperfeiçoamento significativo nas redes neurais. Na última década, a disponibilidade de hardware, dados e o desenvolvimento de novas técnicas favoreceram o rápido desenvolvimento das redes neurais. Em anos recentes, algoritmos de inteligência artificial atingiram níveis tão altos de desempenho que hoje são amplamente utilizados na indústria e pesquisa acadêmica.

Para a atenção visual especificamente, redes neurais e *deep learning* são utilizadas em alguns modelos como uma forma de incorporar fatores *top-down* a modelos antes exclusivamente dependentes de fatores *bottom-up*. Se considerarmos que os aspectos que compõem uma cena não são aleatórios, mas são compostos por elementos que já são do conhecimento do expectador (Chun, 2005), seria possível extrair contexto acerca do que ocorre a partir dos objetos que fazem parte da composição da cena. Portanto, algoritmos de *deep learning* de reconhecimento de objetos podem ser usados no desenvolvimento de modelos com fatores *top-down*. Por esses motivos, o foco da pesquisa se voltou para essas redes com a perspectiva de que elas pudessem auxiliar no desenvolvimento de algoritmos de atenção visual em que o contexto do que está sendo mostrado seja levado em consideração.

Um algoritmo que tem ganhado proeminência é o algoritmo YOLOv3 (do inglês "*you only look once*"), em especial sua terceira versão (Redmon e Farhadi, 2018). Esse algoritmo identifica classes de objetos em vídeos e imagens com extrema rapidez em comparação com outras soluções existentes e tem se

mostrado promissor. Parte disso se deve ao seu processo de divisão do quadro (ou *frame*) em pequenas caixas com diferentes pesos atribuídos a elas, conceito que será explorado mais adiante. Pelas suas qualidades, o YOLOv3 pode se configurar como ferramenta eficiente para introduzir contexto a modelos *bottom-up* a partir da detecção e classificação de objetos presentes na cena.

No entanto, na literatura há poucos algoritmos de atenção visual e *deep learning* projetados para vídeos 360°, apesar de ser economicamente atraente pesquisar sobre técnicas de processamento de vídeo 360°. O mercado desse segmento da indústria pode atingir até 50 bilhões de dólares até 2025 (IMARC Group, 2020). Portanto, é cada vez mais importante pesquisar sobre atenção visual em vídeos 360°.

Com base nesse contexto, um dos objetivos desse trabalho é analisar o desempenho de um dos algoritmos de atenção visual baseado em fatores *bottom-up* mais tradicionais, o algoritmo GBVS, em vídeos 360°. O segundo objetivo é analisar o desempenho da rede YOLOv3 em vídeos 360° para se conheça o potencial desse algoritmo na elaboração de algoritmos de atenção visual para que, futuramente, algoritmos especialistas em vídeos 360° possam ser projetados

Para tanto, foi necessário escolher um *dataset* específico para que pudessem ser realizados os experimentos e a métrica com qual o desempenho desses algoritmos seria avaliado, bem como analisar que métricas poderiam ser utilizadas para medir o desempenho do algoritmo *top-down* proposto. Por fim, foi necessário gerar mapas de saliência e classificação dos objetos em cada frame e compará-los com resultados obtidos por esses algoritmos em imagens e vídeos tradicionais.

Esse trabalho está dividido em cinco capítulos. O Capítulo 2 fará uma breve revisão dos conhecimentos básicos necessários para compreender o problema a ser estudado. O Capítulo 3 descreve os experimentos e simulações realizadas neste trabalho. Os resultados dos experimentos se encontram no Capítulo 4 que também inclui uma análise destes resultados, seguido das conclusões no Capítulo 5.

2 REVISÃO BIBLIOGRÁFICA

Esse capítulo faz uma revisão dos conceitos necessários para o estudo aprofundado do tema. Inicia-se com uma descrição de vídeos 360°, passando por uma revisão de atenção visual e do algoritmo GBVS e terminando com uma descrição do algoritmo YOLOv3.

2.1 VÍDEOS 360°

Vídeos 360°, panorâmicos ou em realidade virtual, como o nome já indica, são aqueles projetados em um espaço tridimensional que colocam o espectador imerso no conteúdo. Cada vídeo é capturado utilizando um conjunto de câmeras, no qual cada câmera aponta para uma direção diferente (El-Ganainy e Hefeeda, 2016), ou ainda por uma câmera com múltiplas lentes. Em ambos os casos, a imagem capturada deve compreender todo o espaço 360º a partir de um único ponto de vista. Uma sobreposição entre as imagens geradas por câmeras ou lentes com ângulos de visão diferentes é utilizada para permitir que as imagens possam ser "costuradas" umas às outras em um processo conhecido como *stitching*. A Figura 1 ilustra o processo de *stitching*. Finalmente, as imagens resultantes da captura são sincronizadas, passando por um tratamento de cor (de modo que cada imagem gerada seja consistente com as imagens vizinhas) e então projetadas em uma esfera.

Esse tipo de vídeo não pode ser visto dessa maneira por monitores convencionais. É necessário que o usuário utilize um dispositivo conhecido como *Head Mounted Display*, abreviado por HMD (sem tradução para o português). O visor desses dispositivos é geralmente uma pequena tela para ambos os olhos (HMD biocular) ou pode conter uma tela para cada olho (HMD binocular), como o Playstation VR. Alguns visores mais simples chamados de monoculares possuem uma única tela para somente um dos olhos do usuário.

Por meio do HMD, o usuário pode então explorar o espaço 3D ao movimentar a cabeça. Os movimentos do usuário são quantificados por meio de ângulos conhecidos como eulerianos. Esses movimentos são realizados ao redor de 3 eixos: empinamento (do inglês, *pitch*), cabeceio (do inglês, *yaw*) e balanceio (do inglês, *roll*) (Marcato, 2010), correspondentes às direções (x, y, z), como mostra a Figura 2. No entanto, o usuário somente visualiza uma pequena porção do vídeo a cada instante de tempo, conhecida como *viewport* (janela de exibição, em português). O tamanho dessa janela depende do campo de visão (*field of vision* , FOV, em inglês) do visor, apesar de que na literatura é comum usar *viewport* e FOV como termos sinônimos.

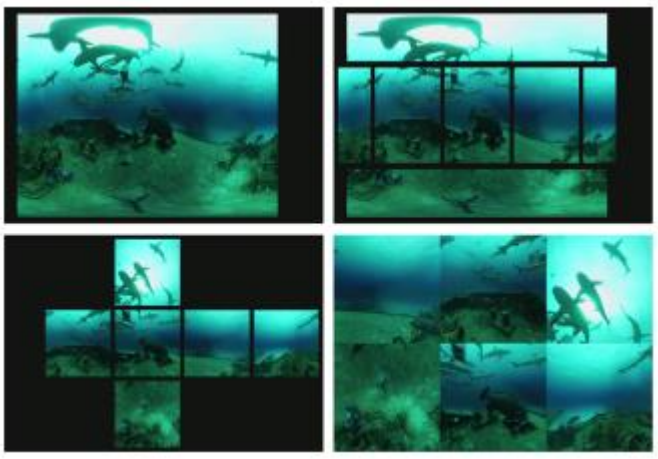

Figura 1. Exemplo de *stitching* (El-Ganainy e Hefeeda, 2016).

Figura 2. Usuário utilizando um HMD e as direções possíveis de movimento (El-Ganainy e Hefeeda, 2016).

Para que os vídeos possam ser visualizados por meio do HMD, o conteúdo do vídeo 360° deve ser projetado em alguma superfície plana. Essa projeção gera alguns problemas, como distorções que levam às imagens a adquirirem um aspecto de terem sido esticadas ou até mesmo a perda na qualidade da imagem. Nas últimas décadas, diversos tipos de projeção foram propostos, para propósitos diferentes, sejam para atenuar alguma característica ou ressaltar outra. Elas podem ser divididas em duas categorias:

I. **Projeção de Qualidade Uniforme**: Todas as partes do plano esférico são mapeadas para o plano 2D com a mesma qualidade. Estas são as projeções mais comuns comercialmente e incluem a projeção equirretangular (nas áreas de cartografia é conhecida como vista cilíndrica equidistante), a cúbica, a dodecaédrica rômbica e o esquema de segmentação em *tiles*.
II. **Projeção de Qualidade Variável**: O mapeamento confere maior qualidade às regiões do plano nas quais o usuário possui maior probabilidade de visualizar ou às áreas que ele está visualizando atualmente. Estes tipos de projeções são as menos utilizadas atualmente e incluem a projeção piramidal e a projeção cúbica com desvio (*offset cubic map*).

As projeções equirretangular e cúbica são as mais utilizadas por vídeos 360° atualmente. Aos poucos, a projeção cúbica tem se tornado mais popular devido a sua implementação em serviços como Facebook e YouTube. No entanto, a projeção equirretangular foi predominante nos primórdios das tecnologias de vídeos panorâmicos. Porém, uma vez que nenhum dos vídeos utilizados nesse trabalho possui projeção cúbica, ela não será abordada nos tópicos posteriores.

A seguir descrevemos em mais detalhes a projeção equirretangular utilizada neste trabalho, com uma discussão acerca de suas principais caraterísticas e problemas associados a sua implementação.

2.2 PROJEÇÃO EQUIRRETANGULAR

Esse tipo de projeção é compatível com a maioria dos HMD e reprodutores de vídeo. Corresponde ao mapeamento da esfera em um plano de dimensões $2\pi r$ x πr, no qual r é o raio da esfera (El-Ganainy e Hefeeda, 2016). O exemplo mais comum da utilização desse tipo de projeção são os mapas cartográficos, como ilustrado na Figura 3.

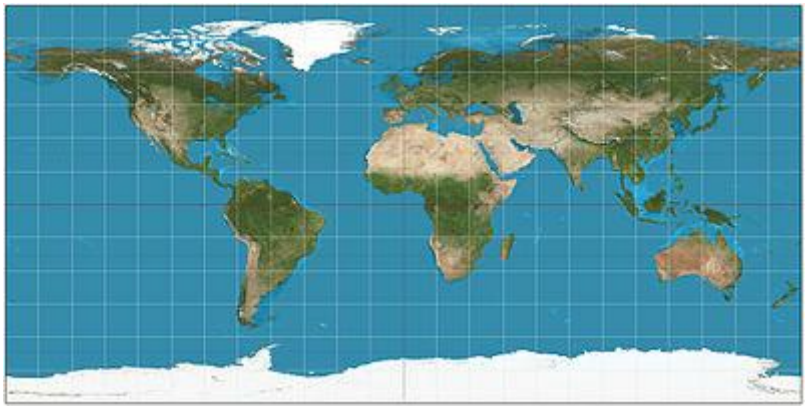

Figura 3. Exemplo de projeção equirretangular (El-Ganainy e Hefeeda, 2016).

Projetar uma esfera em um plano 2D retangular gera lacunas, cuja quantidade aumenta conforme se aproxima dos polos. Não só isso, esse processo resulta em distorções e artefatos visíveis nessas regiões devido ao fato de que objetos próximos aos polos se achatam e se deformam horizontalmente, de forma que os tamanhos desses objetos são sobrerepresentados. Outra desvantagem de utilizar essa projeção é o aumento de pixels redundantes nessas regiões, que causam um desperdício de banda de transmissão para o usuário (El-Ganainy, e Hefeeda, 2016).

2.3 MODELOS COMPUTACIONAIS DE ATENÇÃO VISUAL

Algoritmos de atenção visual são modelos computacionais que tentam simular o funcionamento dos mecanismos de atenção do sistema visual humano. Atualmente algoritmos de atenção visual têm aplicações nas mais diversas áreas, como visão computacional e publicidade.

Como já discutido, os algoritmos de atenção visual são baseados nos estudos sobre o sistema visual humano. O desenvolvimento desse tipo de algoritmo auxilia tanto a entender como os seres humanos olham para uma imagem, como a melhorar técnicas de visão computacional Judd (2011).

A maior parte dos algoritmos de atenção visual possui uma estrutura semelhante, como Judd (2011) afirma em sua tese, “A ideia principal é computar vários atributos (em inglês: *features*) em paralelo e unir suas saliências em uma única representação comumente chamada de mapa de saliência.” Desta forma, um mapa de saliência é uma representação das zonas salientes de uma imagem.

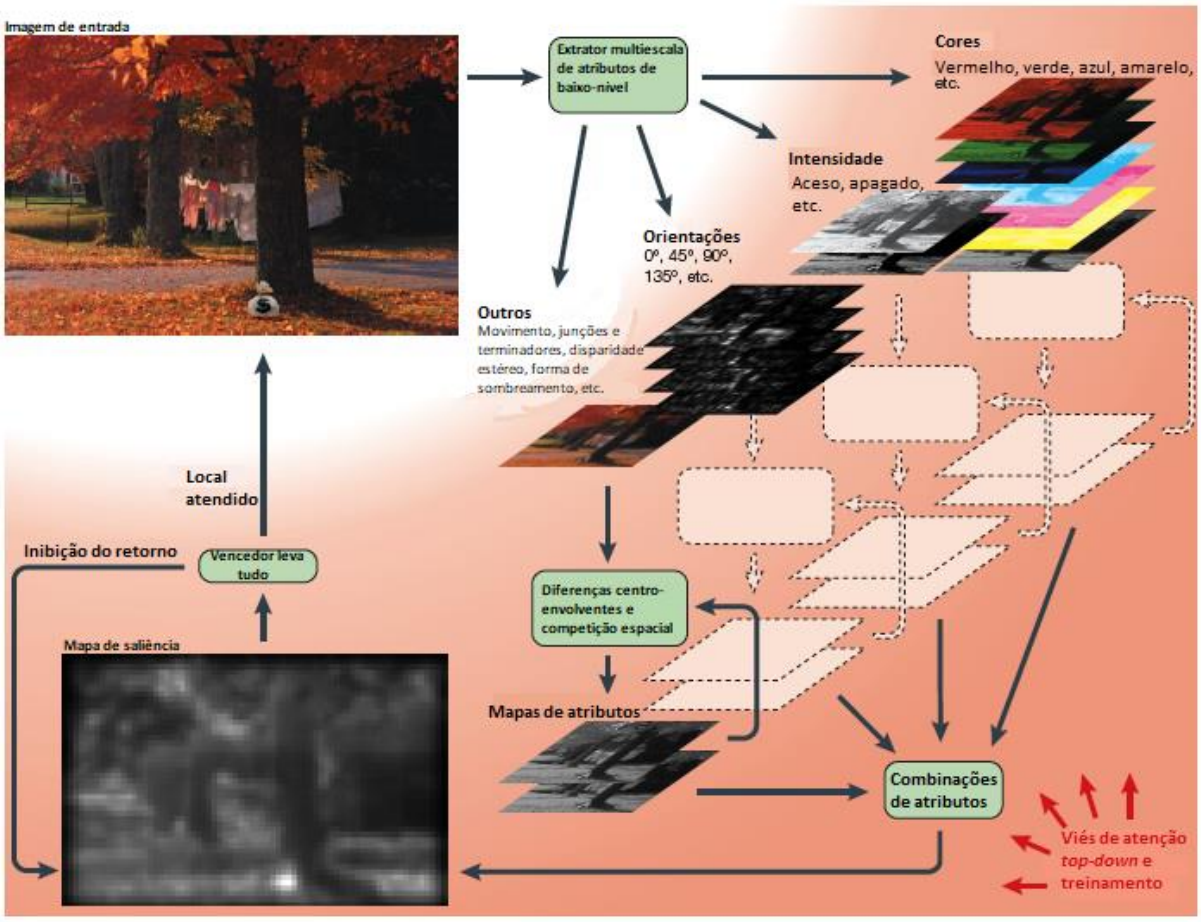

Figura 4. Estrutura clássica do algoritmo de Itti, Koch e Niebur (1998).

Como afirmou também Judd no mesmo trabalho, o processo de obtenção de um mapa de saliência é compreendido inicialmente pela formação de representações piramidais da imagem de entrada de forma a facilitar a extração de diversas *features* em diferentes escalas. *Features* como intensidade, cor e orientação costumam ser mais comumente utilizadas. As características de cor são formadas por canais vermelho-verde e azul-amarelo. Já as características de intensidade são calculadas como a média dos canais de cor. Informações de contraste são obtidas com a aplicação de diferenças gaussianas, ou diferenças centro-envolventes (em inglês, *center surround differences*). Essa operação busca encontrar o valor médio de uma região e compará-lo com o de regiões adjacentes. Os resultados são então somados, dando origem a mapas cujos valores são dependentes das *features*, chamados de mapas de conspicuidade. Os mapas de conspicuidade são normalizados e, após terem um peso atribuído a cada um, são somados, resultando no mapa final de saliência. Este mapa é representado em escala de cinza onde pontos salientes são representados pela cor branca.

Para entender essa estrutura, uma possível abordagem é relembrar do trabalho proposto por Itti, Koch e Niebur (1998) cujo modelo está ilustrado na Figura 4. Em seu artigo original, os autores propuseram uma abordagem baseada na forma de visão dos primatas. A partir do mapa de cores RGB, um mapa é criado para cada atributo após a imagem ser normalizada por um filtro gaussiano para separar a matiz de tonalidade da matiz de intensidade. Esse processo é a forma clássica com a qual um algoritmo de atenção opera até mesmo nos dias de hoje.

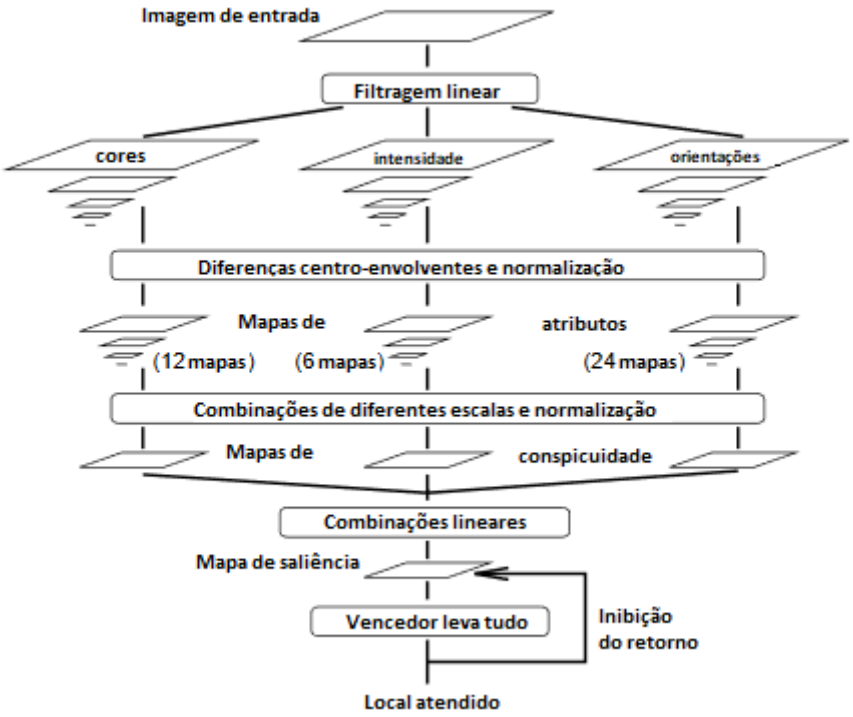

Figura 5. Esquemático do algoritmo de Itti et al. (1998).

Com os mapas de intensidade obtidos para cada canal de cor, o mapa de orientação é obtido por meio da aplicação de filtros de Gabor. Os mapas contendo as *features* são combinados em 3 "mapas de conspicuidade", divididos por intensidade, cor e orientação. A abordagem de separar em três mapas de conspicuidade surge a partir do princípio de que *features* semelhantes competem entre si pela saliência. Os pontos mais salientes são selecionados com uma abordagem na qual pode ser feita um paralelo com o aprendizado não supervisionado, uma vez que nesse também existe uma espécie de camada competitiva nos quais somente os vetores vencedores podem fazer parte de uma ou de outra classe. O nome desse tipo de abordagem de seleção é comumente chamado de "*winner-takes-all*" (em português, "o vencedor leva tudo"). A Figura 5 ilustra o algoritmo de Itti *et al.* (1998).

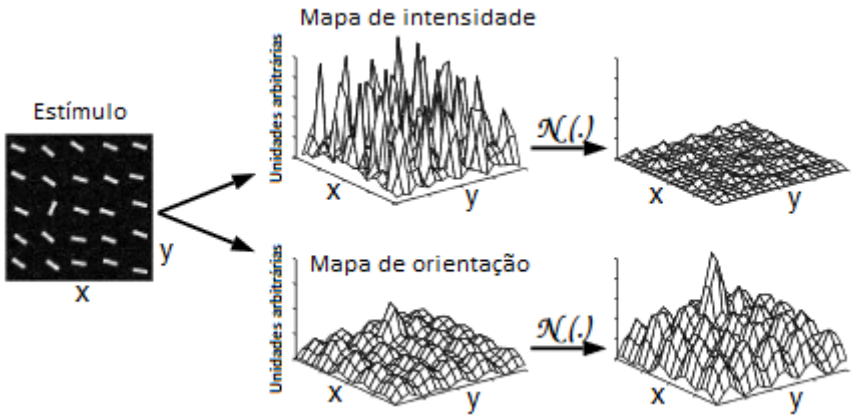

Figura 6. Operador de normalização N(.). (Itti et al., 1998)

Ao final, os três mapas são normalizados e somados, resultando no mapa de saliência final. De modo a diminuir o mascaramento de objetos salientes em alguns mapas, que pode ser causado por ruído ou pela presença de objetos menos salientes, após a combinação de todos os mapas, é feita uma normalização com um filtro N(.), como ilustrado a Figura 6.

A cada instante de tempo, o valor máximo do mapa de saliência define a localização da região mais saliente da imagem, ou seja, a região com o foco de atenção mais provável. Após o desenvolvimento deste algoritmo, surgiram diferentes implementações e abordagens para a construção de modelos de atenção visual. Na Figura 7, podemos ver uma comparação dos mapas de saliência obtidos por alguns modelos de atenção visual disponíveis na literatura.

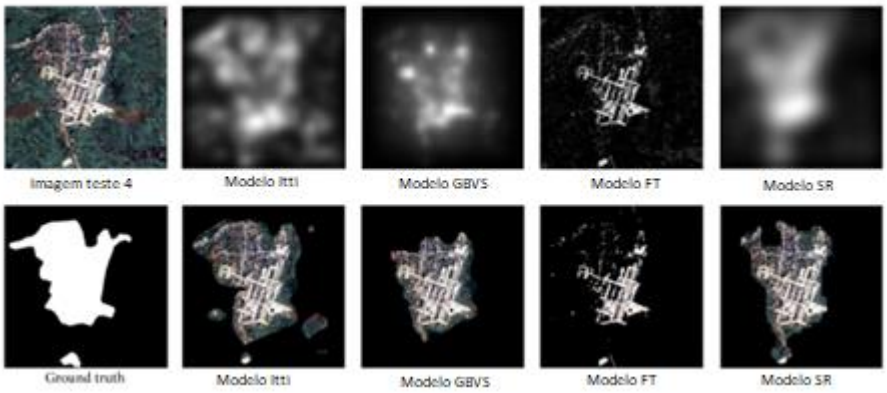

Figura 7. Comparação entre diferentes modelos de atenção visual (Judd, 2011).

Itti e Koch (2001) afirmaram que os fatores *top-down* do mecanismo de atenção visual do sistema visual humano são lentos e orientados a tarefas (Henderson e Hollingworth, 1999). Esses fatores têm um peso determinante na saliência da cena e na probabilidade de uma área da cena ser visualizada. Fatores top-down podem ser implementados em algoritmos de três formas diferentes: 1) quando as tarefas são de busca, são atribuídas mais importância às características do objeto procurado; 2) outros modelos investigam a *gist* (em português, essência) da cena; 3) por último, também existem aqueles que fazem um modelamento específico quando a tarefa é muito complicada ou interativa (como dirigir um carro), (Pena, 2014).

No entanto, estudos mostram que a unidade de atenção de um ser humano não é exclusivamente *top down* ou *botton up*. Isso implica que ambas as abordagens não são mutuamente exclusivas. O uso de abordagens *top-down* surgiu do argumento de que humanos pensam através de objetos (Pena, 2014). Algumas técnicas foram desenvolvidas ao longo dos anos, como o classificador bayesiano de Borji *et al.* (2014) ou o modelo de Judd com classificador de objetos por *Machine Learning*, cujo treinamento é feito com dados reais de fixação ocular, de forma que a rede aprenda os pontos salientes baseando-se nas regiões fixadas pelos usuários.

O modelo de Judd, por meio de técnicas de *Machine Learning* e agregando aspectos de modelos consolidados na literatura, faz uma combinação de 33 canais, como cor, orientação, detecção de objetos, dentre outros. Cada um desses canais pode ser dividido em um de três grupos de acordo com o nível de suas características: baixo, médio e alto-nível. As características de baixo-nível são características *bottom-up* e compõem a maioria dos canais, ou seja 28 no total. Os 13 primeiros correspondem à energia de filtros de pirâmides direcionáveis (*steerable pyramids*) (Judd *et al.*, 2009). Do trabalho descrito por Itti e Koch, surgem inspiração para os canais 15 ao 17, que representam informações de orientação, intensidade e contraste. O canal 14 é uma implementação do modelo de Torralba *et al.* (2006), baseado em *subband pyramids*. Os canais 18 ao 23 representam os três canais de cores RGB, junto com funções de probabilidade de cada um. Os últimos canais (24 ao 28) correspondem às probabilidades de cada cor, com seus cálculos feitos por meio de um histograma da imagem após passar por um filtro com mediana de 5 escalas diferentes.

Dentro do estudo de atenção visual, define-se *ground-truth* como tudo aquilo que é comprovado empiricamente, ao contrário do que é atestado por inferência. Para mensurar o desempenho de um algoritmo é comum comparar os seus resultados com os resultados empíricos, isto é, com dados obtidos por meio de experimentos com participantes. O *ground-truth* são valores de referência na área de atenção visual.

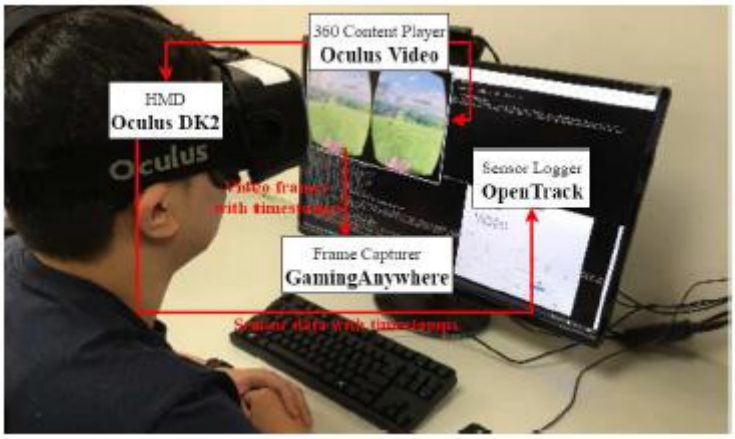

Figura 8. Usuário durante procedimento de obtenção do mapa de fixações (Judd, 2011).

O *ground-truth* do mapa de saliência de um vídeo pode ser obtido por meio de experimentos com um conjunto estatisticamente relevante de vídeos e pessoas. No caso, cada teste retorna um vetor de posições da cabeça do usuário de HMD a cada instante. Com essas posições, pode-se determinar qual região da imagem está sendo visualizada. De posse das posições, são estimadas as fixações do olhar e a partir delas os mapas de saliência. O mapa final se chama mapa de saliência subjetiva ou mapa de fixações ("*fixations maps*"). Geralmente, os mapas de fixações são construídos a partir do ponto central do *viewport* de cada usuário e estimando por quanto tempo cada ponto é visualizado.

A partir disso, determina-se o desempenho (precisão) do algoritmo com base no quanto a saliência predita se aproxima da saliência medida através da realização de experimentos. Para se medir esse desempenho, podemos utilizar diversas métricas que medem se há similaridade entre os mapas de saliência obtido e o medido em relação ao *ground-truth*. Neste trabalho utilizaremos a *area under the curve of the receiver operating characteristic* (AUC ROC), ou área abaixo da curva característica de operação do receptor, em português, para estimar o desempenho dos modelos de saliências. A curva é uma representação do número de positivos verdadeiros versus o número de falsos positivos. Neste trabalho é utilizada uma versão diferente do método conhecido com AUC-JUDD. Esse método é um classificador binário que separa positivos verdadeiros dos falsos positivos. Dado um limiar adquirido amostrando todos os valores de saliência dos mapas de fixação, o número de positivos verdadeiros é o número de todos os pontos do mapa de saliência no local da fixação acima do limiar nos pontos que coincidem nos dois mapas e os falsos positivos são todos aqueles acima do limiar nos pontos não coincidentes

Foi utilizada uma implementação em MATLAB que se encontra no sítio http://saliency.mit.edu/results_mit300.html. O algoritmo recebe como entradas um mapa de saliência, uma matriz binária de saliência subjetiva e dois outros parâmetros, que podem ser 1 ou 0. O primeiro valor indica a informação de "*jitter"*. Com *jitter* = 1, será adicionada uma constante diferente de zero a todas as localidades dos mapas para garantir que a ROC possa ser calculada de forma robusta. No entanto, se foi utilizado um filtro Gaussiano na obtenção da saliência, não é necessário alterar o valor de "*jitter"*. O segundo valor é para a variável "*toPlot"* que controla se a curva ROC será plotada ou não, sendo que com o '1' a curva ROC é desenhada. Antes de realizar a avaliação, o script tenta garantir que as duas matrizes tenham o mesmo tamanho.

2.4 O MODELO GBVS

O modelo de atenção visual utilizado nesse trabalho utiliza uma abordagem diferente da clássica. Diferentemente dos modelos clássicos, o GBVS não calcula em sua terceira parte o mapa de saliência por meio de normalização, convolução de filtros Gaussianos ou por meio de interações não lineares (Harel *et al.*, 2006).

Segundo Harel, Koch e Petrona (2006), dada uma imagem qualquer, o algoritmo inicialmente extrai os atributos da imagem da mesma forma que o algoritmo descrito no capítulo anterior, gerando ao final um mapa $M : [n]^2 \rightarrow R$. A partir deste, o próximo passo é gerar um mapa de ativação $H : [n]^2 \rightarrow R$. Nesse caso, $n \in N^+$; *i* e *j* são os índices das matrizes *H* e *M*. Portanto, a tupla (*i, j*) representa um ponto pertencente a esses mapas. Os autores garantem que podem considerar as matrizes *M* e *H* como matrizes quadradas de dimensão *n* sem perda de generalidade ou precisão matemática.

Para gerar *H*, inicialmente um outro ponto qualquer (*p, q*) pertencente à matriz *M* é definido, tal que $i \neq p, j \neq q$. Em seguida, o índice de dissimilaridade entre os valores *M*(*i, j*) e *M*(*p, q*)é calculada utilizando a seguinte equação (Harel *et al.*, 2006):

$$d\left((i,j),(p,q)\right) \triangleq \left| log \frac{M(i,j)}{M(p,q)} \right|. \quad (1)$$

Em outras palavras, os autores definiram a medida de dissimilaridade como sendo a distância entre os pontos (*i, j*) e (*p, q*), calculada por meio do valor absoluto da diferença (em escala logarítmica) entre *M*(*i, j*) e *M*(*p, q*).

O próximo passo é gerar um grafo a partir de *M*. Grafos são normalmente representados pela notação *G*(*V, A*), em que *V* representa um conjunto não vazio de pontos, chamados vértices, que são interligados uns aos outros por um subconjunto de *V* de pares não ordenados, conhecidos como arestas (Biggs *et al.*, 1986). A cada vértice, é associado um número que representa um peso ou um custo para cada aresta. Denota-se o número de arestas por |*A*| e o número de vértices por |*V*|. Grafos cujas arestas possuem orientação, isto é possuem um vértice de entrada e outro de saída, são chamados de dígrafos e suas arestas são chamadas de arcos. A Figura 9 representa um exemplo de um modelo de grafo.

Grafos podem ser esparsos ou densos. Grafos esparsos são aqueles em que |*A*| é muito menor que $|V|^2$ Por sua vez, grafos densos são aqueles nos quais |*A*| é próximo a $|V|^2$ (Alchieri, 2019). A implementação de grafos em algoritmos pode ser feita por meio de uma matriz de adjacência ou de uma lista de adjacência.

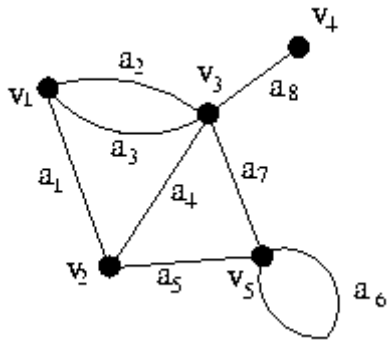

Figura 9. Modelo de grafo (Lin *et al.*, 2014).

A implementação por lista de adjacência é compacta e é a mais indicada para a maioria das aplicações, apesar de que ela é melhor quando deseja-se representar um grafo esparso. Nesta, cada vértice é armazenado de forma arbitrária. Cada linha da lista contém um vértice, os vértices adjacentes e os pesos das arestas que os conectam (Alchieri, 2019).

A implementação por matriz de adjacência é mais utilizada nos casos de grafos densos. Sejam os elementos de uma matriz de adjacência *A*(*G*) de um grafo *G* representados por a_{ij}, na qual *i* é o índice das linhas da matriz *A*(*G*) e *j* é o índice das colunas de *A*(*G*). Definem-se também o *i*-ésimo e o *j*-ésimo vértices de *G*(*V, A*) como v_i e v_j. No caso de um grafo não direcionado (grafos que não são dígrafos), matrizes de adjacência são matrizes quadradas de dimensões iguais ao número de vértices do grafo e

cada elemento da matriz representa uma aresta. Se dois vértices forem adjacentes, o elemento a_{ij} irá conter o peso da aresta que liga os dois vértices. Caso os vértices não forem adjacentes, o elemento irá conter o valor *0*. Por exemplo, se v_1 e v_2 forem adjacentes e ligados por uma aresta de peso *3*, o elemento a_{12} terá valor igual a *3*. Se o grafo for um dígrafo, a matriz simplesmente será um caso especial da matriz do grafo com arestas não direcionados, no qual convenciona-se que suas linhas sejam os vértices de saída de um arco e suas colunas os vértices de entrada (Alchieri, 2019). Para o GBVS, essa é a melhor implementação porque dificilmente os grafos gerados pelo algoritmo serão esparsos. Assim, cada ponto (*i*, *j*) de *M* é conectado aos outros *n-1* pontos. A partir desse processo, é gerado um dígrafo inteiramente conectado *G*(*V*, *A*), cujos arcos possuem pesos determinados pela seguinte equação:

$$w_1\big((i,j),(p,q)\big) \triangleq d\big((i,j),(p,q)\big)\cdot F\left(i-p,j-q\right), \tag{2}$$

na qual;

$$F\left(a,b\right) \triangleq exp\left(-\frac{a^2+b^2}{2\sigma}\right). \tag{3}$$

Nesta expressão $a = i - p$ e $b = j - q$. Portanto, o peso do arco que liga o ponto (*i*, *j*) ao ponto (*p*, *q*) é igual ao índice de dissimilaridade ponderado por uma função gaussiana 2-D de média zero e variância 2σ. Apesar de σ ser um parâmetro livre, geralmente utilizam-se valores entre um décimo e um quinto da largura dos mapas de atributos.

Uma cadeia de Markov é gerada a partir de *G*(*V*, *A*) por meio da normalização dos pesos dos arcos de saída de cada nó de modo que eles sejam iguais a *1*. Em seguida, é estabelecida uma equivalência entre vértices e estados, e pesos dos arcos e probabilidade de transição.

A última etapa é gerar um novo dígrafo a partir do mapa de ativação gerado no passo anterior. O processo se repete até que seja obtida uma nova cadeia de Markov que é o mapa de saliência. Na prática, o procedimento descrito acima é realizado para cada um dos mapas de *features M* gerados.

O uso de GBVS se justifica pelo fato de que, segundo Harel *et al*. (2006), esse modelo possui um desempenho melhor que algoritmos mais antigos como o de Itti e Koch. Em experimento realizado com 749 variações de 108 imagens de tamanho 600 X 400 pixels, GBVS conseguiu melhores resultados conforme Harel *et al.* (2006).

Foi utilizada uma implementação do GVBS baseada em matrizes de adjacência, em MATLAB, disponível no mesmo sítio onde está disponível o algoritmo "AUC_JUDD"[1]. O algoritmo utilizado chama-se 'flicker_motion.m' que se encontra dentro da pasta 'gbvs'. A razão para o GBVS ter sido escolhido foi sua rapidez em relação a outros algoritmos de atenção visual, o que é um fator bastante importante quando estamos considerando sinais de vídeo 360º. Além disso, o GBVS produz resultados bem robustos.

2.5 O ALGORITMO YOLOv3

Na última década, houve um processo de redescoberta de algoritmos de inteligência computacional e com ele surgiram diversos avanços na aplicação de tais algoritmos nas mais diferentes áreas do conhecimento.

Para entender o funcionamento do algoritmo YOLO os seguintes conceitos devem ser compreendidos:

- *Bounding box*: Como o nome diz, *bounding box* é uma caixa delimitadora. No caso, ela delimita a área do objeto detectado, de forma que todo o objeto se encontra dentro da área delimitada pela caixa. Algoritmos de detecção de objetos tentam prever as coordenadas dessas caixas. O

[1] https://github.com/sergeyk/vislab/tree/master/matlab/gbvs

YOLOv3 prevê as coordenadas x e y do centro da predição da posição do objeto e a medida das alturas e larguras da caixa delimitadora do objeto predito.

- IOU: *Intersection over union*, ou em português interseção sobre a união. Cada objeto em uma imagem possui uma caixa delimitadora que é considerado como *ground-truth* para esse objeto. Por cima dessa caixa, a rede desenha sua própria predição. IOU é uma métrica de precisão que é calculada pela área da interseção entre o *ground-truth* e a predição, dividida pela área da união entre as áreas das duas caixas.

- *Recall* e precisão: *Recall* (revocação) é o resultado da equação $R = {}^{T_p}\!/_{(T_p+F_p)}$, onde T_p são os positivos verdadeiros (*true positives*) e F_p sãos os falsos positivos. Já precisão é calculada substituindo F_p na equação anterior por F_n, os falsos negativos.

- mAP (*mean average precision*): *Average precision* (AP) é a área abaixo da curva *precision-recall* de uma classe, a qual representa os valores de precisão no eixo *y* e os valores de sensibilidade no eixo *x*. mAP é a média dos valores AP de todas as classes.

Todas as versões do algoritmo YOLO funcionam sob o mesmo princípio básico, apenas fazendo melhorias incrementais a cada nova versão. A rede do YOLOv3 precisa ser inicializada com coordenadas iniciais para as *bounding boxes*. Essas *bounding boxes* que inicializam a rede são chamadas pelos autores (Redmon e Farhadi, 2017) de *bounding boxes priors* ou caixas delimitadoras iniciais. Esses valores não são definidos aleatoriamente, mas pelo algoritmo k means, com k = 5. A rede então divide a imagem, ou o quadro do vídeo, N^2 células (supondo uma imagem quadrada), com N células na horizontal e N na vertical. Se um objeto estiver dentro de determinada célula, essa será a responsável por detectar o objeto.

O algoritmo YOLOv3 identifica então quatro coordenadas t_x e t_y são as coordenadas do centro da *bounding box* (em relação ao canto superior esquerdo de cada célula) e t_w e t_h são os valores de largura e altura de cada *bounding box*, respectivamente. Se a célula onde está a *bounding box* estiver deslocada em (c_x, c_y) em relação ao canto superior esquerdo da imagem, como mostra a Figura 10, e as *bounding boxes* iniciais tiverem largura p_w e altura p_h, as coordenadas de cada *bounding box* são dadas pelas seguintes equações:

$$\begin{aligned} b_x &= \sigma(t_x) + c_x \,, \\ b_y &= \sigma(t_y) + c_y \,, \\ b_w &= p_w e^{t_w} \,, \\ b_h &= p_h e^{t_h} \,, \end{aligned} \tag{4}$$

nas quais que $\sigma(.)$ é a função softmax (Bishop, 2006).

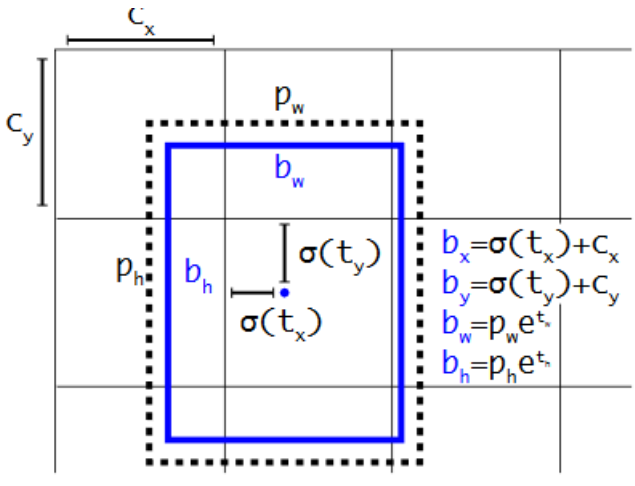

Figura 10. *Bounding boxes* com predição de localização. Figura retirada de Redmon e Farhadi, 2018.

Durante o treinamento, é utilizada uma função soma dos erros quadráticos (Redmon *et al.*, 2016) para ajustar os pesos. Supondo que o *ground truth* de alguma das quatro coordenadas seja $\hat{t}_*$, o gradiente será o valor computado para o *ground truth*, menos a predição, isto é, $\hat{t}_* - t_*$. O algoritmo também usa um modelo de regressão logística para determinar se a predição deve ser utilizada ou descartada (Lo *et al.*, 2017). Esse sistema só designa uma *bounding box* para cada objeto de *ground truth* portanto, mesmo se uma *bounding box* inicial não foi designada a nenhum objeto *ground-truth*, não há erro para essa coordenada ou predição de classe (Lo *et al.*, 2017).

A extração de features dessas camadas é feita utilizando um conceito próximo de redes piramidais (em inglês, *pyramid networks*) (Redmon e Farhadi, 2017). A esse extrator são adicionadas camadas convolucionais de dimensões *3* x *3* ou *1* x *1*, com suas saídas começando com dimensões de *256* x *256*, com diminuições progressivas a cada camada subsequente. A última delas em cada escala prediz a *bounding box* por meio de um tensor 3-D. O artigo original do YOLOv3 cita que em seus experimentos com o *dataset* COCO (Lin *et al.*, 2014) o tensor tem dimensões $N \cdot N \cdot \left[3 \cdot (4 + 1 + 80)\right]$ para os 4 *offsets* das *bounding boxes* com 1 predição de objetividade e 80 predições de classes. Em seguida, é feita a predição do objeto. O mapa de *features* é amostrado a uma taxa 2 vezes a taxa original. Esse mapa é combinado com um mapa anterior da rede.

O YOLOv3 utiliza classificadores logísticos independentes e, durante o treinamento, um critério baseado em entropia binária cruzada é aplicado (Lo *et al.*, 2017). O treinamento é feito por meio do framework Darknet (Redmon, 2013-2016). Esse tipo de implementação já supõe que diferentes classes podem se sobrepor. Implementações mais tradicionais, como o *softmax*, consideram as classes sobrepostas como sendo uma só, quando normalmente esse não é o caso. Por exemplo, se um *dataset* possui as classes “homem” e “pessoa”, uma camada *softmax* teria dificuldades para diferenciar uma da outra.

	Tipo	Filtros	Tamanho	Saídas
	Convolucional	32	3 × 3	256 × 256
	Convolucional	64	3 × 3 / 2	128 × 128
1×	Convolucional	32	1 × 1	
	Convolucional	64	3 × 3	
	Residual			128 × 128
	Convolucional	128	3 × 3 / 2	64 × 64
2×	Convolucional	64	1 × 1	
	Convolucional	128	3 × 3	
	Residual			64 × 64
	Convolucional	256	3 × 3 / 2	32 × 32
8×	Convolucional	128	1 × 1	
	Convolucional	256	3 × 3	
	Residual			32 × 32
	Convolucional	512	3 × 3 / 2	16 × 16
8×	Convolucional	256	1 × 1	
	Convolucional	512	3 × 3	
	Residual			16 × 16
	Convolucional	1024	3 × 3 / 2	8 × 8
4×	Convolucional	512	1 × 1	
	Convolucional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Conectadas		1000	
	Softmax			

Figura 11. Estrutura da Darknet-53 (Redmon e Farhadi, 2018).

A Figura 11 mostra a estrutura das camadas do YOLO. Nota-se que a saída é feita com uma função *softmax* e são 53 camadas convolucionais. O YOLOv3 utiliza arquitetura de classificadores estado da arte com menos operações em ponto flutuante e mais velocidade. Desse modo, os testes comparativos apresentados na Figura 12 mostram que o YOLOv3 possui uma maior rapidez, ao mesmo tempo mantendo a precisão, em comparação com o classificador RetinaNet.

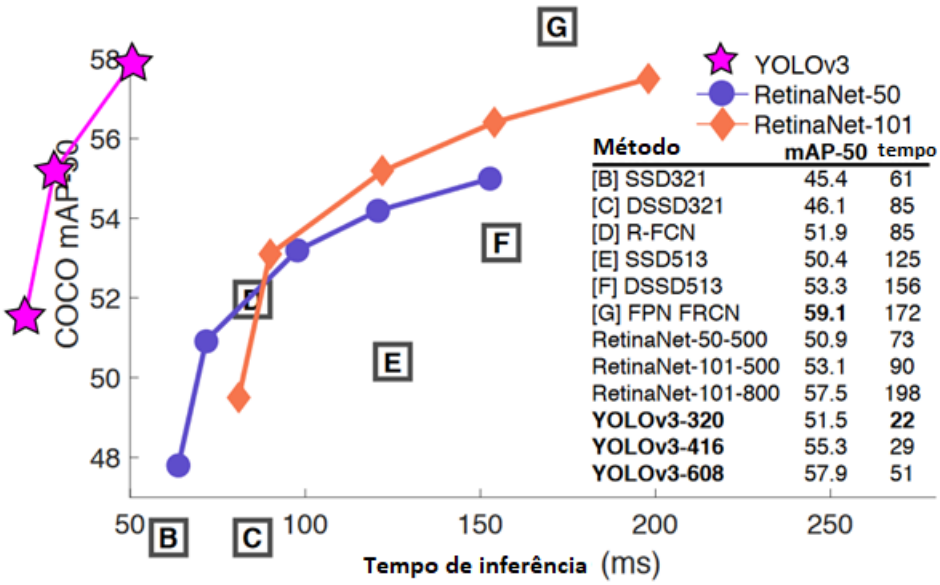

Figura 12.Tradeoff velocidade/precisão mostrado pela mAP (do inglês, *mean average precison*), utilizando 0,5 IOU (do inglês, *Intersection Over Union*) 53 (Redmon e Farhadi, 2018).

3 DESCRIÇÃO EXPERIMENTAL

Este capítulo descreve os procedimentos utilizados para a obtenção dos resultados analisados no capítulo 4.

Inspirado no trabalho de Judd (2011), os experimentos consistiram em calcular os mapas de saliência de uma série de vídeos 360°, utilizando um modelo computacional de atenção visual. Em seguida, os mapas de saliência gerados foram comparados com os mapas de fixação gerados a partir de dados experimentais, que foram obtidos a partir das posições do centro das *viewports* dos diferentes participantes. Estas informações são capturadas para se determinar o *ground-truth*, ou seja, os dados experimentais que serão referência para os algoritmos. Por fim, os vídeos selecionados tiveram seus frames processados pelo YOLOv3 para identificar e classificar os elementos em cena, sejam eles pessoas, animais, objetos etc. Espera-se que com esse procedimento seja possível determinar se há alguma correlação dos contextos nos quais os vídeos estão inseridos com as regiões dos vídeos para onde a atenção humana se volta.

Para o conjunto de treinamento, foi escolhido o banco de dados da *University of Texas at Dallas* (Taghavi *et al.*, 2019). Dos vídeos disponíveis, foram escolhidos 14 para a realização dos experimentos. Cada vídeo do banco de dados pode ser encontrado no endereço fornecido. Somente o vídeo 4 não pode ser visto no YouTube, já que foi produzido pela *University of Texas at Dallas* e não foi colocado no YouTube, mas pode ser encontrado no sítio do projeto da UTD. A nomenclatura foi respeitada, por isso não existem os vídeos 15 e 16, que foram eliminados no banco de dados original. A Figura 13 mostra alguns frames ilustrativos de alguns dos vídeos utilizados. Cada vídeo foi categorizado em uma de seis categorias descritas abaixo:

- Categoria 1: Baixa atividade temporal (pouco movimento) e baixa atividade espacial (texturas lisas, paredes, céu sem nuvens etc.);
- Categoria 2: Baixa atividade temporal (pouco movimento) e alta atividade espacial (textura rugosa, grama, cabelo etc.);
- Categoria 3: Alta atividade temporal e baixa atividade espacial;
- Categoria 4: Alta atividade temporal e alta atividade espacial;
- Categoria 5: Faces e interações (pessoas conversando com o espectador, vídeos com pessoas que chamem a atenção do usuário para um ponto específico);
- Categoria 6: Paisagens (vídeos estáticos sem foco de atenção, onde o usuário explora livremente o cenário e não se prende a nenhum ponto específico).

A Tabela 1 apresenta um resumo de todos os vídeos utilizados, com uma descrição breve de suas características principais.

Procurou-se utilizar pelo menos um vídeo de cada categoria, mas o vídeo 13 não foi utilizado em decorrência de problemas relacionados à leitura de seus vetores de fixação pelos algoritmos utilizados para gerar os mapas de fixação.

Cada vídeo vem acompanhado das saliências subjetivas encontradas por meio de um experimento, no qual 30 voluntários assistiram aos vídeos enquanto sensores presentes no HMD rastreavam seus movimentos de cabeça. Foi utilizado o HMD (sem fio) Oculus. Com estas posições é possível determinar o *viewport* de cada usuário para cada um dos vídeos. Cada participante assistiu a 14 vídeos de sessenta segundos em ordem aleatória (para compensar efeitos temporais). A posições foram salvas em diferentes arquivos (formato .csv) contendo vetores de posição das cabeças dos participantes.

Tabela 1. Relação dos vídeos utilizados e suas descrições.

ID	Nome	Endereço	Taxa	Desc.	Cat
1	360° Sunset Timelapse ft evolv | 360 Video | Virtual Reality Relaxation Experience	youtu.be/ESRz3-btZIA (0:40)	25	Exterior, Natureza	6
2	Nature 360° in Montana - Virtual 5K Nature Meditation for Gear VR, Oculus Go and Daydream	youtu.be/30cSb3wTc7U (0:00)	30	Exterior, Natureza	6
3	Closet Set Tour In 360 VR SCREAM QUEENS	youtu.be/ULixPLH-WA4 (0:07)	30	Interior, Urbano, Pessoas	5
4	----------	https://bit.ly/2QVCO76	30	Interior, Urbano, Pessoas	2
5	Caring for Rhinos Discovery VR (360 Video)	youtu.be/7IWp875pCxQ (0:18)	30	Exterior, Natureza	1
6	Amazing 360° DUBSTEP Dance Video!! @MattSteffanina Choreography	youtu.be/ze_w7Lh97Co (0:05)	30	Interior, Urbano, Pessoas	3
7	VR 360° 8K Drone Footage - VR360 Gimbal - Aerial 8K video	youtu.be/9XR2CZi3V5k (0:01)	25	Exterior, Natureza	2
8	Grand & Petite Anse Aerial in 360°VR with professional ambisonic audio	youtu.be/6TlW1ClEBLY (0:45)	30	Exterior, Natureza	6
9	Heartland Motorsports Park 18-04-29 Chasing Karl	youtu.be/tVsw0DvAWdE (0:15)	30	Exterior, Urbano	3
10	360 V19Cable4K	youtu.be/cNlQrTkXkOQ (0:15)	30	Exterior, Natureza, Pessoas	2
11	BEN-HUR (2016) - Chariot Race 360° Video - Paramount Pictures	youtu.be/jMyDqZe0z7M (0:00)	24	Exterior, Urbano, Pessoas, Animais	4
12	New York City 8K - VR 360 Drive	youtu.be/2Lq86MKesG4 (0:12)	30	Exterior, Urbano, Pessoas	4
14	VR 360° Video | 4K! | DHL Post Tower Glass Elevator in 360!	youtu.be/elhdcvKhgbA (0:14)	30	Interior, Urbano	4
17	Жар-птица VR 360 | Drop tower video 360	youtu.be/jau-Ric7kls (1:11)	30	Exterior, Urbano, Pessoas	5

Foi utilizado uma versão adaptada do script “Overlay.m” da UTD para extrair as matrizes de pontos dos vetores. O primeiro passo desse algoritmo é reorientar alguns vídeos que foram alterados para a reprodução dos experimentos. Se o valor da variável “rotation” for diferente de zero, significa que houve um movimento do usuário e a sua orientação deve ser reordenada de modo a corresponder ao *viewport* que foi visualizado. Os movimentos são representados por vetores que são convertidos para coordenadas esféricas para que seja aplicado um filtro. Os mapas de saliência para um frame de cada vídeo foram adquiridos tomando o ponto central das *viewports* a partir de um filtro Gaussiano. Um mapa de calor é criado para cada frame aplicando o filtro *passa-baixa* Gaussiano para todos os 30 participantes. A partir das coordenadas esféricas, é possível saber para qual região cada usuário estava olhando. Ao aplicar o filtro gaussiano, a região do *viewport* se torna um ponto bem pequeno na imagem que por sua vez representa o ponto central do *viewport*. A informação da posição do ponto central do *viewport* de um usuário é somada à informação de cada usuário que também assistiu aquele vídeo. O mapa final da

posição do quadro é a soma dos mapas de todos os usuários, dividido pela quantidade de usuários (Taghavi *et al.*, 2019). É realizada ao final, para ilustração, uma sobreposição dos mapas de fixação ao vídeo escalonado, como apresentado na Figura 14.

Figura 13. Frames de alguns dos vídeos vídeos 360° Sunset Timelapse ft evolv | 360 Video | Virtual Reality Relaxation Experience, Nature 360° in Montana - Virtual 5K Nature Meditation for Gear VR, Oculus Go and Daydream, Closet Set Tour In 360 VR SCREAM QUEENS, 4

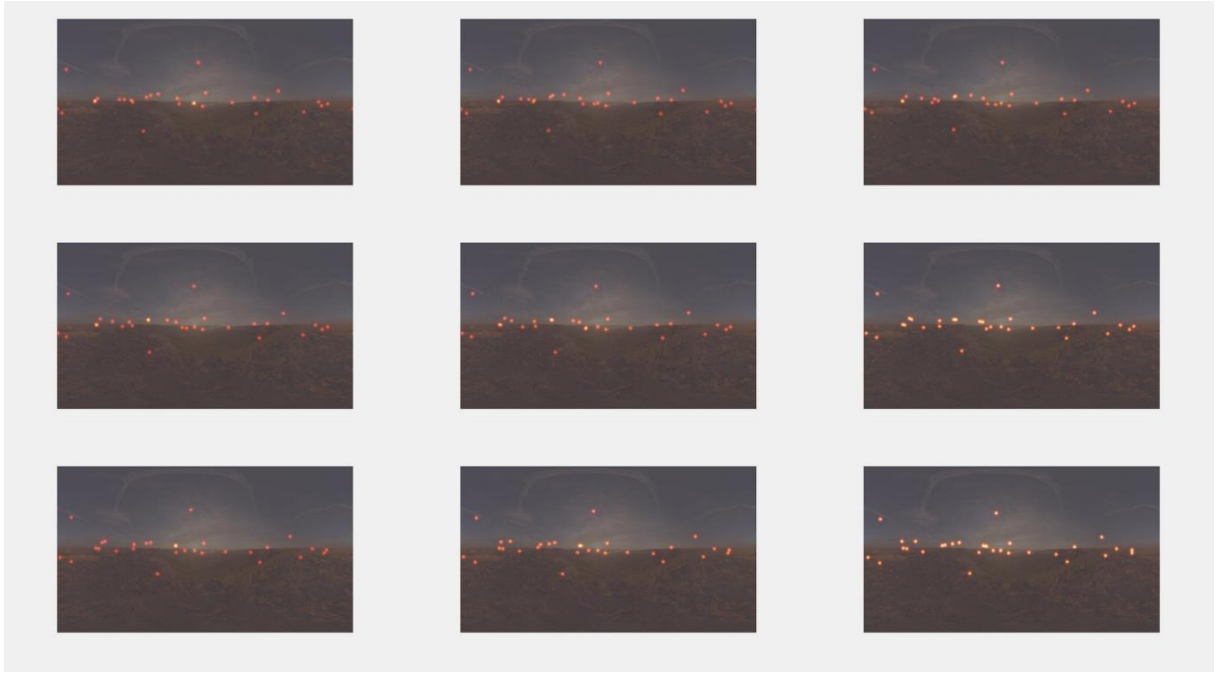

Figura 14. Frames do vídeo 360° Sunset Timelapse ft evolv | 360 Video | Virtual Reality Relaxation Experience com sobreposição dos centros das *viewports*.

Cada vídeo possui resolução 4K (3840 X 2048 pixels). Vídeos desse tamanho demandam muito tempo e memória RAM para serem processados. Por esse motivo sua resolução foi modificada para a full HD (1920 X 1080), gerando sinais quatro vezes menores do que o sinal original. Esse é um fator que poderia interferir na obtenção ou na precisão do mapa de saliência. Entretanto, no trabalho de Judd *et al.* (2009) foi mostrado que não há diferenças consideráveis nos mapas de saliência quando a resolução é diminuída

por um certo valor razoável que mantenha uma resolução espacial aceitável. Naturalmente, quando essa redução é muito grande a precisão do mapa começa a cair. Ainda assim, o processamento dos vídeos demandou muito esforço computacional e, por esse motivo, só foi feita a análise com 600 quadros de cada vídeo 360º. Por uma questão de padronização, foram utilizados somente 600 quadros para todos os vídeos. Neste trabalho, não estamos considerando os aspectos temporais dos vídeos. Desta forma é indiferente ser realizarmos uma redução temporal. Em seguida, os vídeos passam pelo processamento do YOLOv3. Os vídeos foram processados em 4K e todos os frames de cada vídeo foram utilizados.

Após a conclusão de todos os testes, foi feita uma análise e crítica dos dados encontrados por todos os algoritmos. Todos os experimentos com o GBVS foram feitos em um desktop com processador I7-4790 de 3,6 GHz, 32 GB de RAM e uma placa de vídeo GTX-970. Os experimentos com o YOLOv3 foram realizados no Google Collab, uma plataforma de computação na nuvem gratuita baseada em python. O Google Collab foi utilizado porque o processamento dos vídeos demorava horas no hardware acima, mesmo com a redução da resolução dos vídeos. O código utilizado para o YOLOv3 também está disponível no *github* do projeto. Os comandos de instalação do YOLOv3 no Collab foram retirados do código que pode ser encontrado no sítio https://github.com/ivangrov/YOLOv3-GoogleColab (Goncharov, 2019). O restante dos comandos de execução do YOLOv3 foi retirado do site de seu criador (Redmon, 2013-2016).

4 RESULTADOS EXPERIMENTAIS

Este capítulo apresenta os resultados dos experimentos descritos no capítulo anterior 3.

Devido à quantidade de material gerado por todos os algoritmos, só serão apresentados alguns exemplos para embasar a conclusão. O restante do material pode ser encontrado na página https://github.com/Bernmor/TCC2, juntamente com os *scripts* utilizados.

O algoritmo GBVS completo foi rodado em todos os vídeos, com resolução dos mapas de *features* igual a um quarto da resolução da imagem original. Ao final, um arquivo com todas as informações foi salvo, mas somente o mapa mestre de cada frame foi utilizado. No caso, mapa mestre (‘master_map’) é a designação que o algoritmo usa para o mapa de saliência GBVS.

Como já foi dito antes, o GBVS em imagens bidimensionais pode chegar a alcançar desempenhos de até 98% quando avaliado pela métrica AUC-JUDD. Apesar de não termos conseguido o mesmo desempenho, os resultados obtidos neste trabalho podem ser considerados satisfatórios, considerando que não foi utilizada nenhum tipo de compensação no ato de calcular os mapas de saliência ou durante a medição de desempenho. Uma curva ROC foi gerada para cada *frame* de cada vídeo utilizado, bem como suas respectivas pontuações AUC-JUDD ROC. Os resultados do escore ROC são apresentados na Figura 15, na forma de um gráfico de barras, com a média da pontuação AUC-JUDD dos *frames* de cada vídeo. A pontuação girou em torno de 80%, o que é aceitável considerando que o GBVS não foi projetado para vídeos e tampouco para vídeos em 360°.

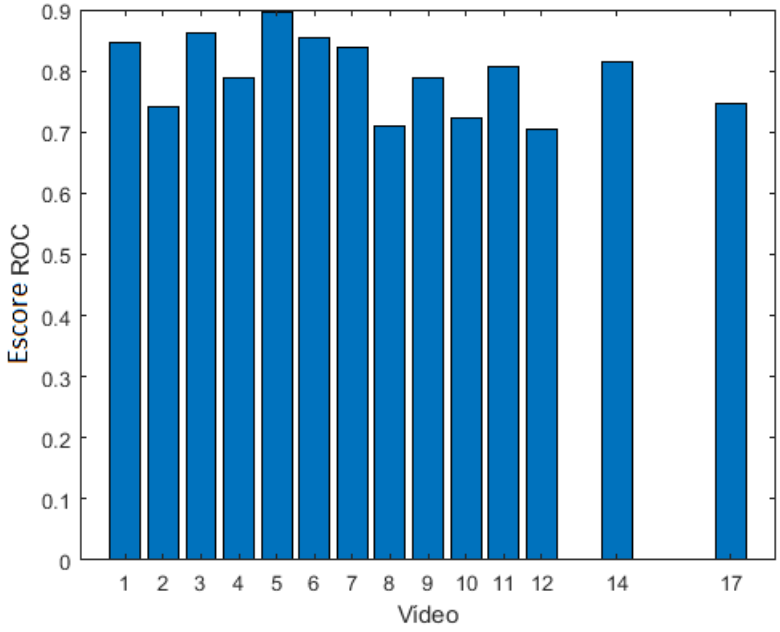

Figura 15. Escore ROC médio utilizando o GBVS.

A média do ROC de cada um dos quadros apresentados na Figura 15 considera que existe uma variação dos valores entre quadros que pode ser alta dependendo de onde o vetor do mapa de fixações estiver posicionado. Em alguns vídeos, o GBVS previu corretamente que os participantes do experimento olhariam mais para o centro do vídeo. Em outros casos, os participantes concentraram suas atenções no centro, enquanto o GBVS computou mapas com regiões salientes concentradas nos cantos do vídeo. A Figura 16 mostra exemplares de casos onde o algoritmo previu o foco no centro do vídeo e, nesse vídeo específico, o ponto de interesse coincidiu com o centro.

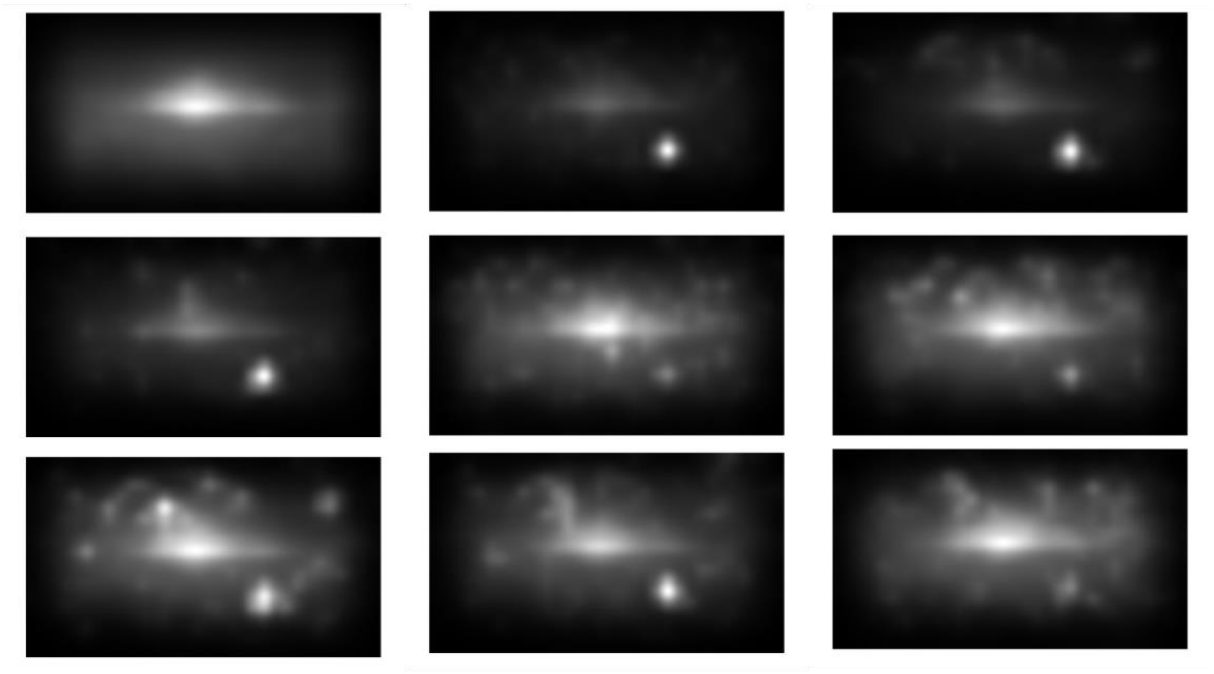

Figura 16. Mapas de saliência gerados pelo GBVS nos nove primeiros quadros do vídeo 1 (ver Figura 14).

Como era de se esperar, em muitos dos vídeos o foco dos usuários se voltou para o centro. Isso mostra como os seres humanos têm a tendência de focar no centro de imagens, como era esperado por Judd no seu trabalho. Desta forma, quando o GBVS gerou mapas com pontos salientes afastados do centro, seu desempenho foi pior. O vídeo 9, em particular, coloca o espectador na posição de piloto e, naturalmente, o foco do espectador é na direção do movimento do carro. Não só isso, como os mapas de fixações mostram, o usuário tende a seguir o objeto em movimento, no caso, o outro carro. O GBVS computou regiões salientes pequenas (seguindo o movimento do outro carro do vídeo) ou das laterais do vídeo. No entanto, a maior parte dos participantes focou no centro do vídeo. A Figura 17 compara o mapa gerado pelo GBVS, com as fixações dos participantes.

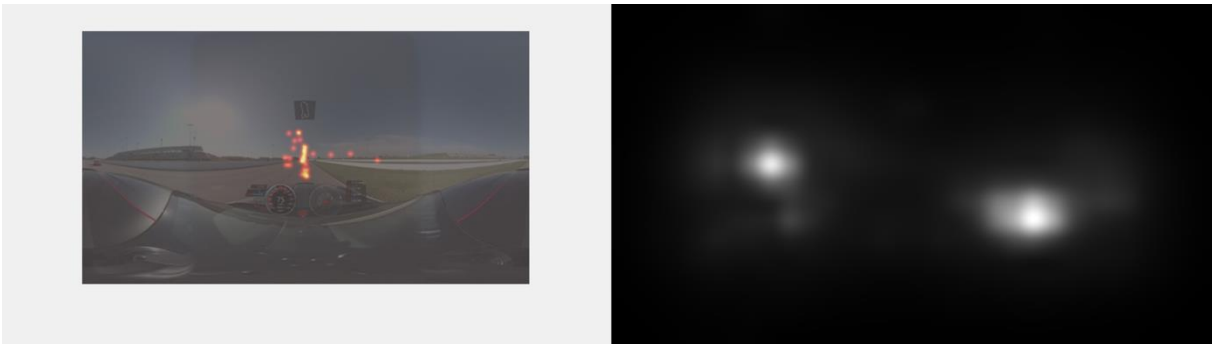

Figura 17. Comparação entre o mapa de saliência e o mapa de fixação dos participantes para um mesmo *frame*.

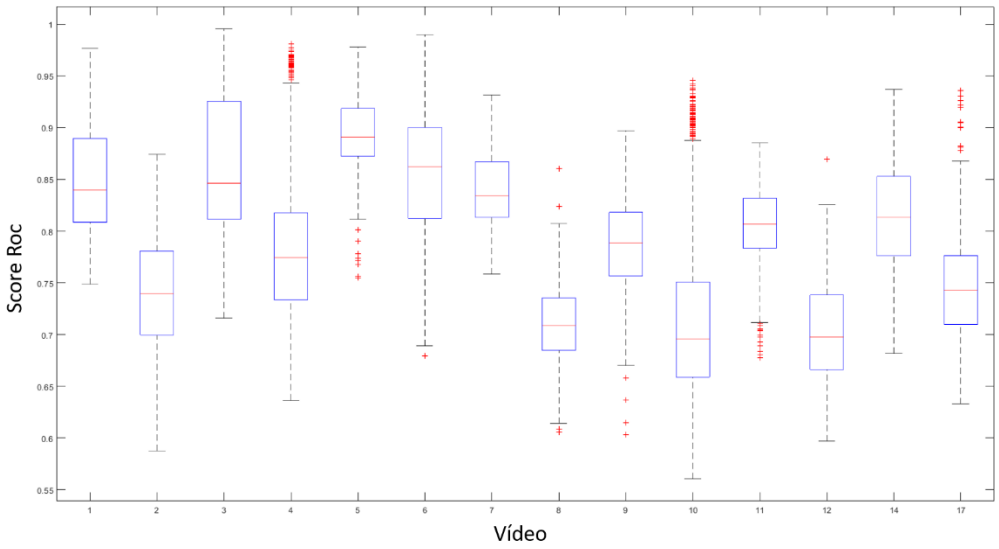

Figura 18. *Boxplots* dos escores ROC computados para os mapas de saliência obtidos com o GVBS para cada um dos vídeos.

Como praticamente um vídeo de cada categoria obteve um desempenho satisfatório, os resultados levam a crer que o desempenho está relacionado mais intrinsecamente com o conteúdo do vídeo. Os piores resultados ocorreram com os vídeos 2, 8, 10 e 12. No caso do vídeo 12, por exemplo, que possui um foco de atenção bem disperso, o expectador tem ao seu redor muitos pontos de interesse, incentivando-o a explorar o cenário. Como o GBVS não está preparado para computar mapas em vídeos 360°, esse pode ter sido um dos motivos da grande diferença entre os mapas empíricos e os computacionais.

Para uma análise mais aprofundada, a Figura 18 mostra o *boxplot* com as linhas vermelhas dentro das caixas representando a mediana dos escores ROC dos *frames* de cada vídeo. Acima das medianas estão os pontos no 75% quartil e abaixo dela estão os pontos no 25% quartil. Os pontos fora da curva estão representados por cruzes e a variância representada pelas linhas pontilhadas. Vemos então qual é a mediana para cada um dos vídeos. Chama atenção novamente os resultados do vídeo 10 que, não só apresentou a maior variância dentre todos os vídeos, como também teve a maior quantidade de pontos fora da curva (em inglês: outliers). O GBVS apresentou boa média nos testes com o vídeo 3 (que contém um homem caminhando por um closet), mas exibiu um intervalo interquartil grande. Isso se deve ao fato de o GBVS ter conseguido encontrar o homem, mas falhou ao identificar quaisquer outras zonas de saliência, como mostra a Figura 19. A Figura 18 também está disponibilizada no endereço do *github* do projeto.

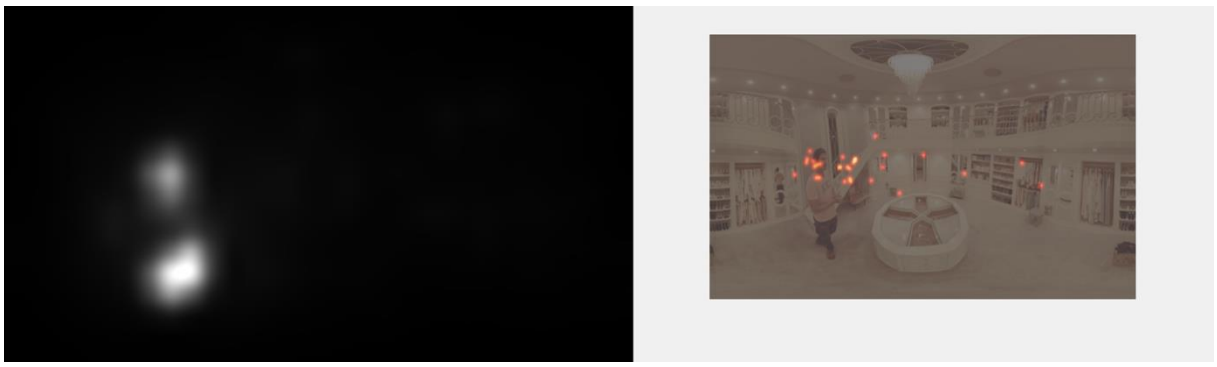

Figura 19. Comparação entre o mapa de saliência do GBVS com o mapa de fixações.

Após gerar os mapas de saliência por meio do algoritmo GBVS, os vídeos foram processados pelo YOLOv3 que poderia corrigir possíveis erros se pesos maiores fossem dados a regiões com objetos classificados, mas que não foram consideradas salientes pelo GBVS. O primeiro resultado que chama a atenção é o fato de que todos os vídeos ocuparam o dobro de espaço da memória quando o YOLOv3 foi aplicado. Seria necessária uma investigação mais aprofundada para descobrir por que isso ocorre. O próprio YOLOv3 pode inserir informações adicionais aos vídeos ao classificar os objetos em cena, no entanto, chama a atenção o fato do tamanho dos vídeos dobrar.

Observamos que o YOLOv3 não identificou nenhum objeto nos vídeos 1 e 2. Estes vídeos contêm imagens naturais, onde não estão presentes objetos facilmente identificados conforme definido pelo conjunto de classes do YOLOv3. Em dois vídeos (7 e 14) o algoritmo interpreta que há somente um único objeto grande na cena. A Figura 20 mostra o algoritmo prevendo que há um bolo no vídeo 7 e a Figura 21 um micro-ondas no vídeo 14. O algoritmo também identifica a classe "*bowl*" no início do vídeo 7. O YOLOv3 apresentou esse mesmo problema nos vídeos 8, 9, 10, 11. No entanto, ele conseguiu identificar objetos corretamente, ao contrário do que ocorreu com os vídeos 7 e 14. No vídeo 8, o YOLOv3 identificou erroneamente objetos próximos ao texto do canto inferior da tela, apesar de haver outros objetos em cena, como pessoas na praia e um barco no oceano, como mostram as Figuras 22 e 23. Curiosamente o barco é classificado como pessoa, como mostra a Figura 24. Conforme a câmera se aproxima, o algoritmo identifica corretamente o barco, como mostra a Figura 25.

Figura 20. Predições do YOLOv3 para o vídeo 7.

Figura 21. Predições do YOLOv3 para o vídeo 14.

Figura 22. Desempenho do YOLOv3 durante o vídeo 8 (cena-exemplo 1).

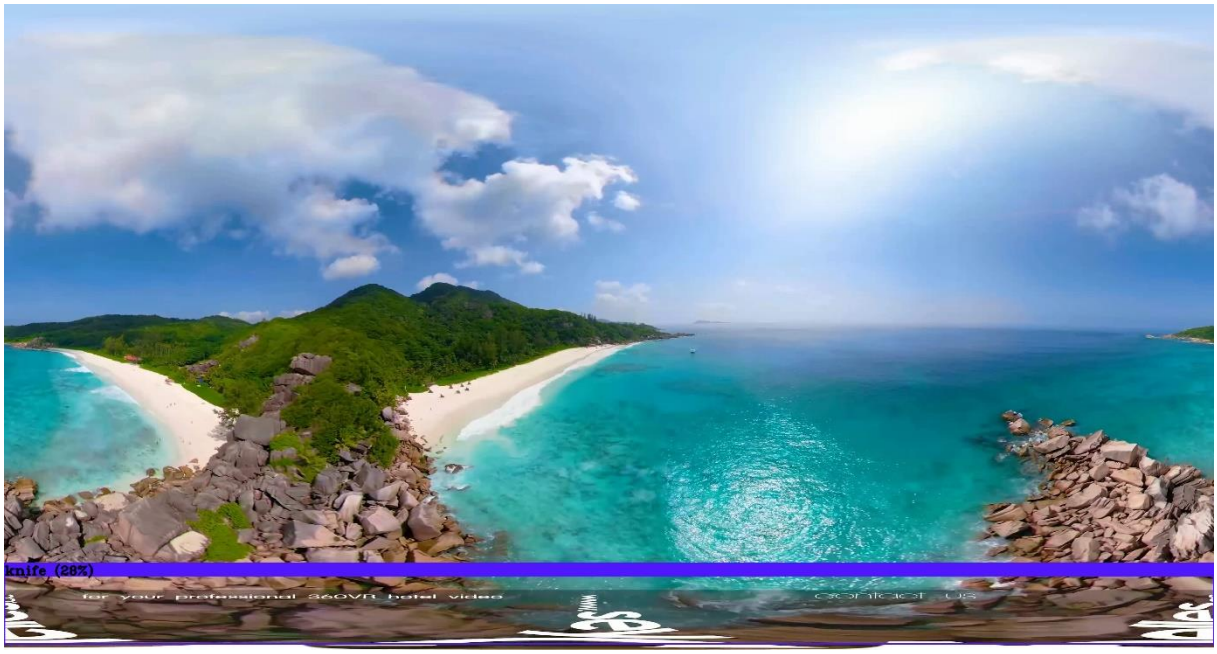

Figura 23. Desempenho do YOLOv3 durante o vídeo 8 (cena-exemplo 2).

Figura 24. Desempenho do YOLOv3 durante o vídeo 8 (cena-exemplo 3).

Figura 25. Desempenho do YOLOv3 durante o vídeo 8 (cena-exemplo 4).

No vídeo 9, o YOLOv3 identificou carros na pista, mas indicou uma motocicleta ocupando toda a tela. Além disso, o YOLOv3 identificou de forma errônea outros objetos menores. Por exemplo, o tacômetro e o velocímetro foram identificados como carros, e placas ao redor da pista foram caracterizadas como semáforos. O desempenho do YOLOv3 durante o vídeo 9 está ilustrado nas Figuras 26 a 29.

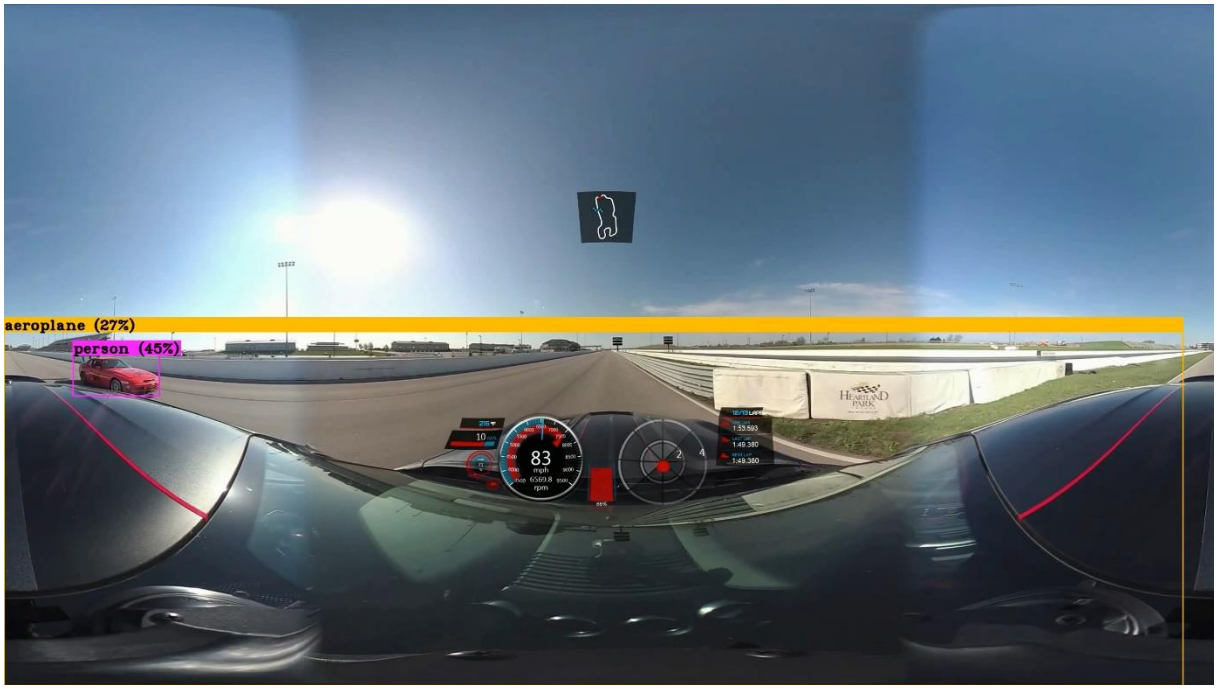

Figura 26. Desempenho do YOLOv3 durante o vídeo 9 (cena-exemplo 1).

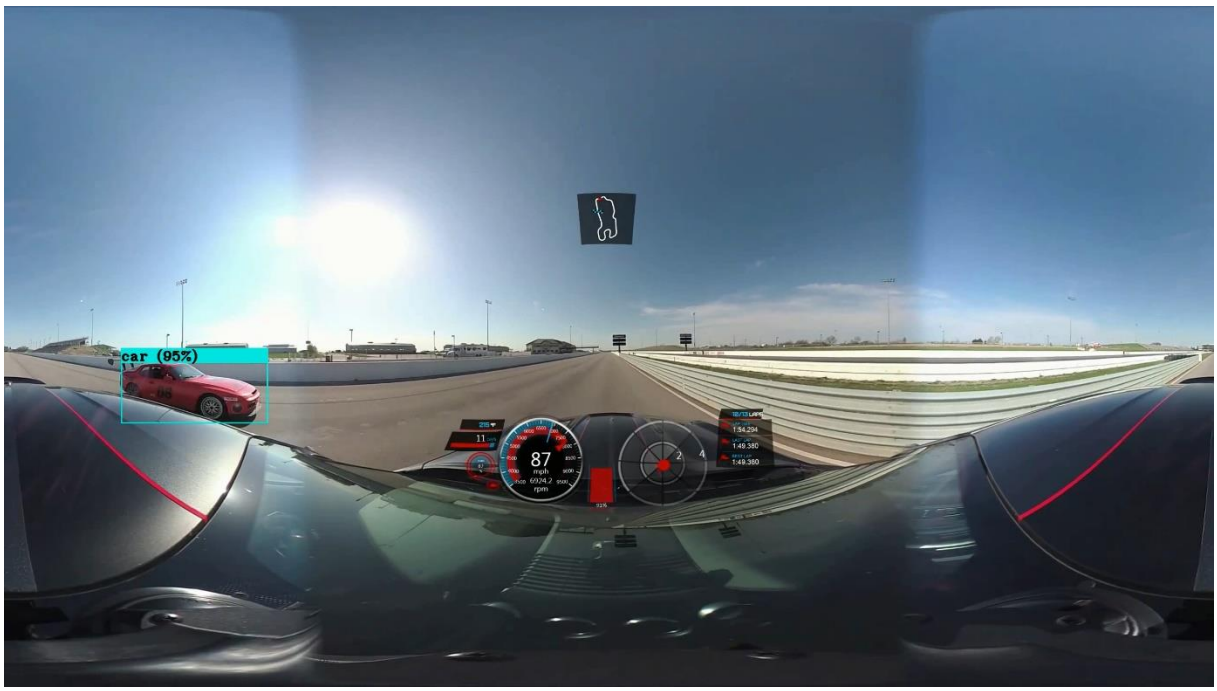

Figura 27. Desempenho do YOLOv3 durante o vídeo 9 (cena-exemplo 2).

Figura 28. Desempenho do YOLOv3 durante o vídeo 9 (cena-exemplo 3).

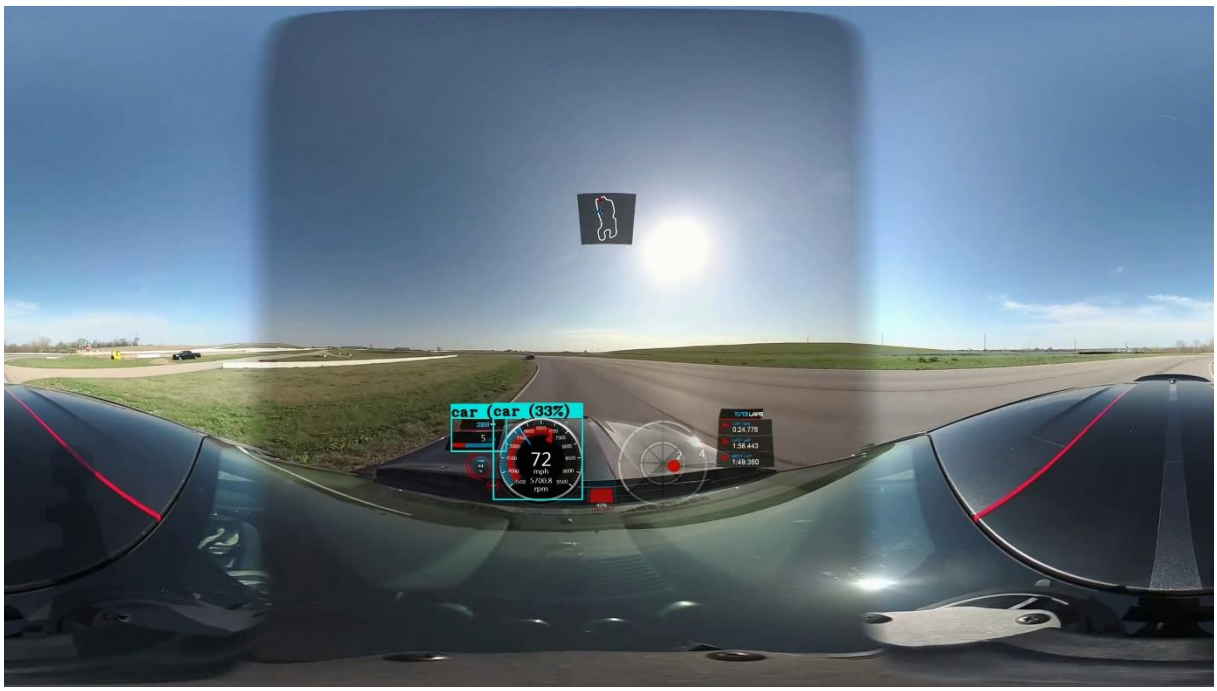

Figura 29. Desempenho do YOLOv3 durante o vídeo 9 (cena-exemplo 4).

No vídeo 10, o YOLOv3 identificou um objeto no canto superior e atribuiu a ele a classe "*aeroplane*", como visto na Figura 30. O algoritmo conseguiu identificar a mulher caminhando. Ele também identificou árvores como sendo pessoas, como é visto na Figura 31. No vídeo 11, o YOLOv3 detectou os cavalos que puxam as bigas que foram classificados como pessoas com frequência e em alguns momentos o YOLOv3 identificou objetos próximos às distorções das bordas. No entanto, o YOLOv3 identifica os objetos nas cenas com muita precisão, apesar de não oferecer a classe correta o tempo todo. As Figuras 32 a 35 ilustram o desempenho do algoritmo nesse vídeo.

Figura 30. Desempenho do YOLOv3 durante o vídeo 10.

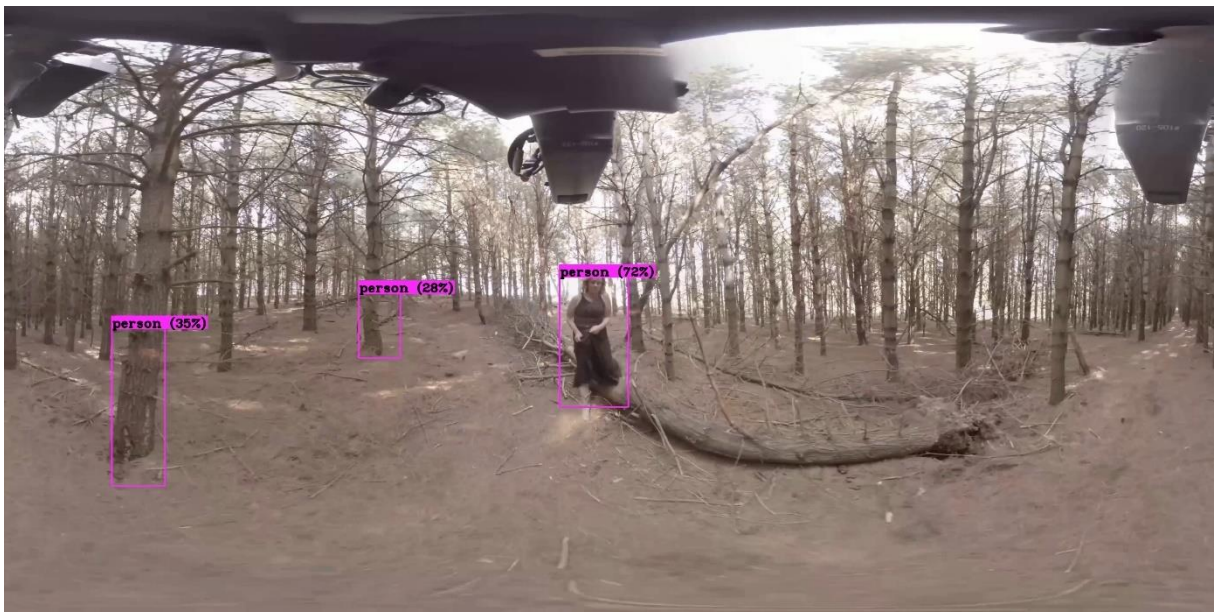

Figura 31. Desempenho do YOLOv3 durante o vídeo 10.

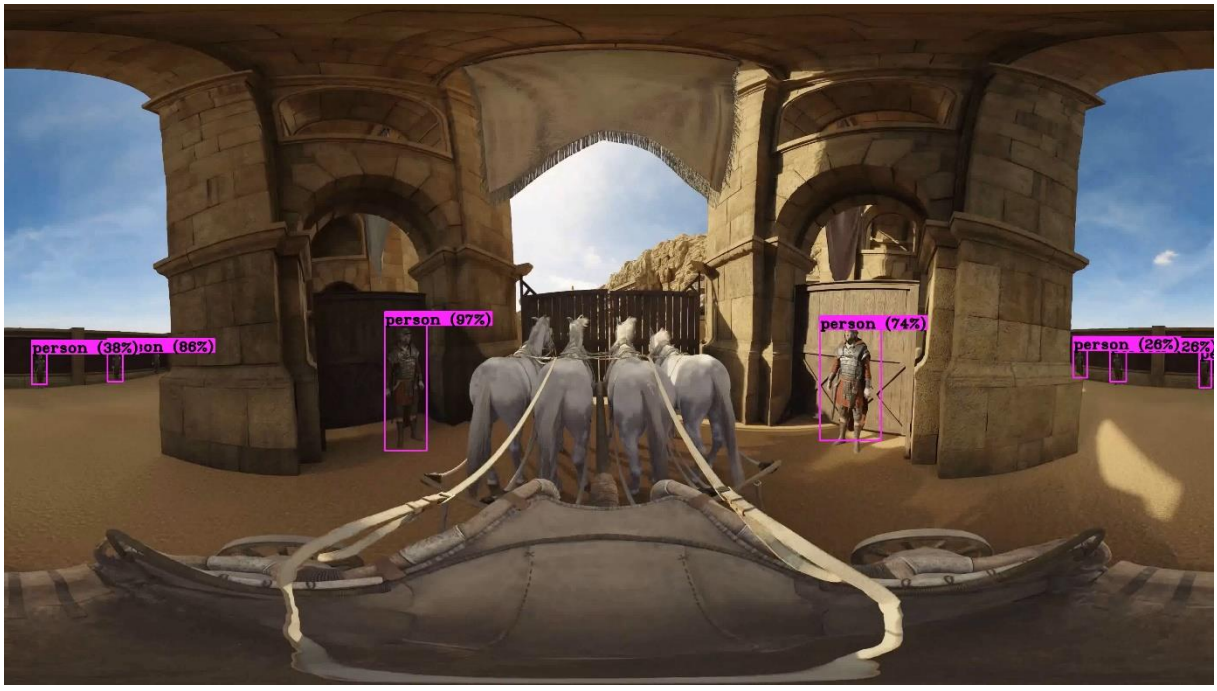

Figura 32. Desempenho do YOLOv3 durante o vídeo 11.

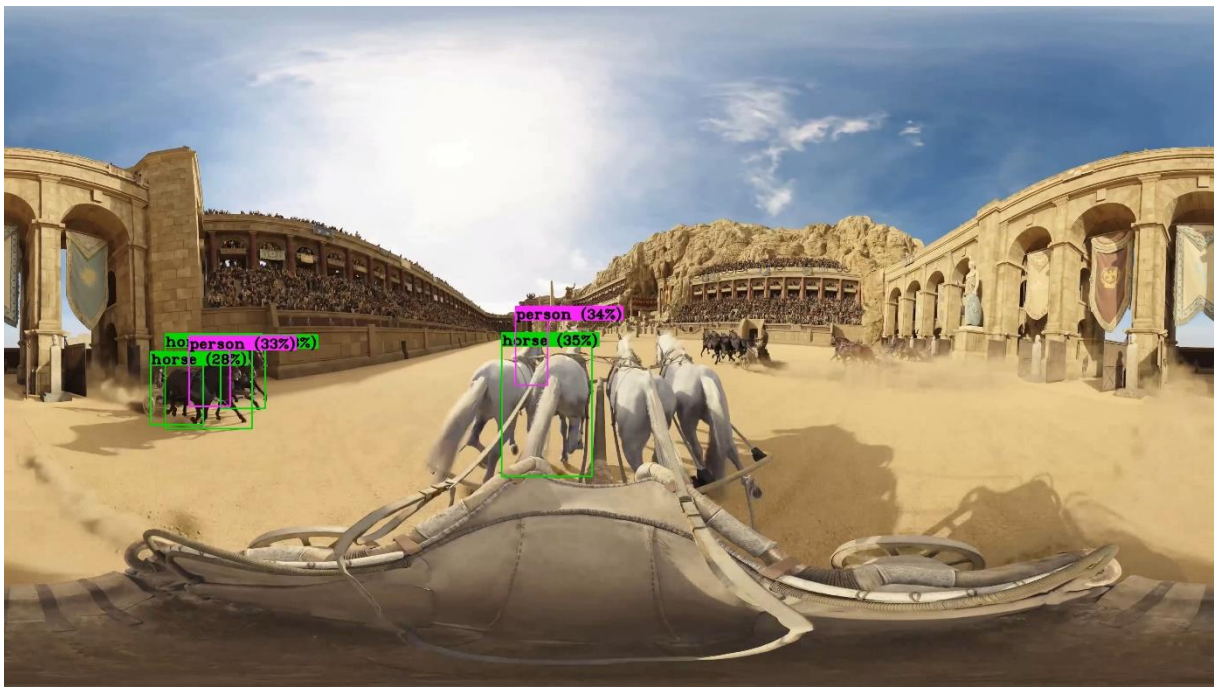

Figura 33. Desempenho do YOLOv3 durante o vídeo 11.

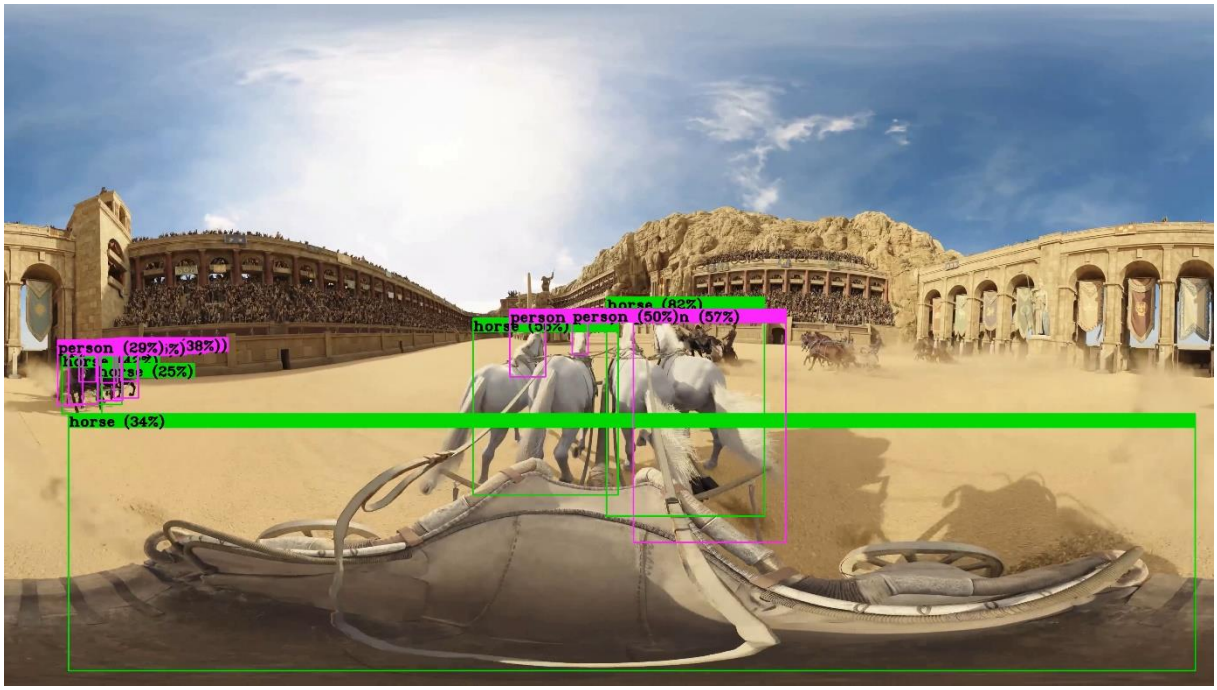

Figura 34. Desempenho do YOLOv3 durante o vídeo 11.

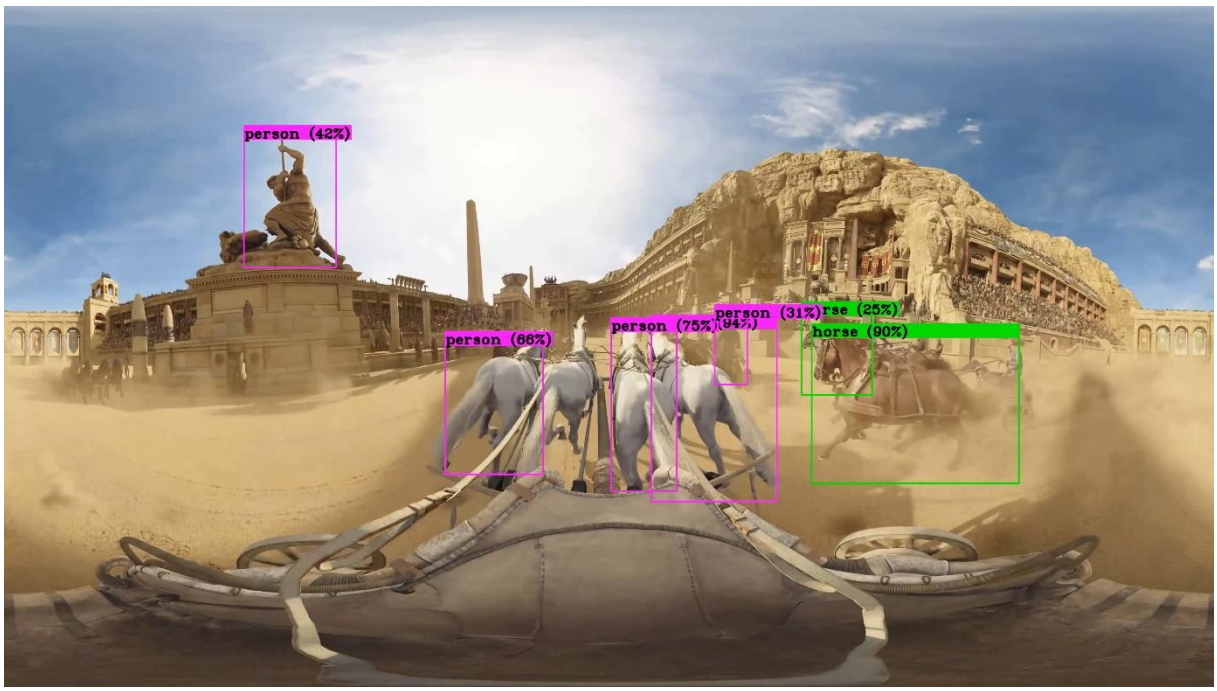

Figura 35. Desempenho do YOLOv3 durante o vídeo 11.

Mesmo que não tenha sido perfeito, o YOLOv3 cometeu poucos erros nos vídeos 3, 5, 6 e 17. Em geral, o algoritmo identificou todos os objetos, mas classificou alguns deles de forma errada. Isso pode acontecer até mesmo quando o YOLOv3 processa vídeos bidimensionais, geralmente porque alguma classificação específica não estava nas amostras de treinamento do YOLOv3. Nesse experimento, no entanto, muitos erros provavelmente ocorreram porque os objetos tiveram a sua forma distorcida quando o vídeo foi projetado para o plano bidimensional. Mais precisamente, os erros foram:

- No vídeo 3 existe uma única classe importante, que é a pessoa andando pelo closet. O YOLOv3 conseguiu identificar corretamente que era uma pessoa e seguiu-a por todo o recinto. O algoritmo identificou o reflexo do homem nos espelhos.

- No vídeo 5, todos os rinocerontes foram reconhecidos, mas foram classificados como elefantes.

- Todas as pessoas foram identificadas corretamente durante todo o vídeo 6, mas, holofotes posicionados mais próximos da borda inferior do vídeo foram classificados como mochilas e maletas (em inglês: *backpacks* e *suitcases*).

No vídeo 17, com exceção de alguns momentos em que o YOLOv3 classifica um objeto como pessoa quando era uma montanha russa, e de algumas pessoas que foram classificadas como malas, o desempenho foi satisfatório. Esses erros estão ilustrados na Figura 39 abaixo.

O YOLOv3 conseguiu desempenho excelente em dois vídeos: 4 e 12. Os dois vídeos continham uma variedade muito grande de classes, como móveis, pessoas, carros, caminhões, mochilas e semáforos. Foram estes os vídeos em que o YOLOv3 mais se aproximou de reconhecer corretamente todos os objetos presentes. As Figuras 40 e 41 mostram a classificação feita pelo YOLOv3 nos dois vídeos.

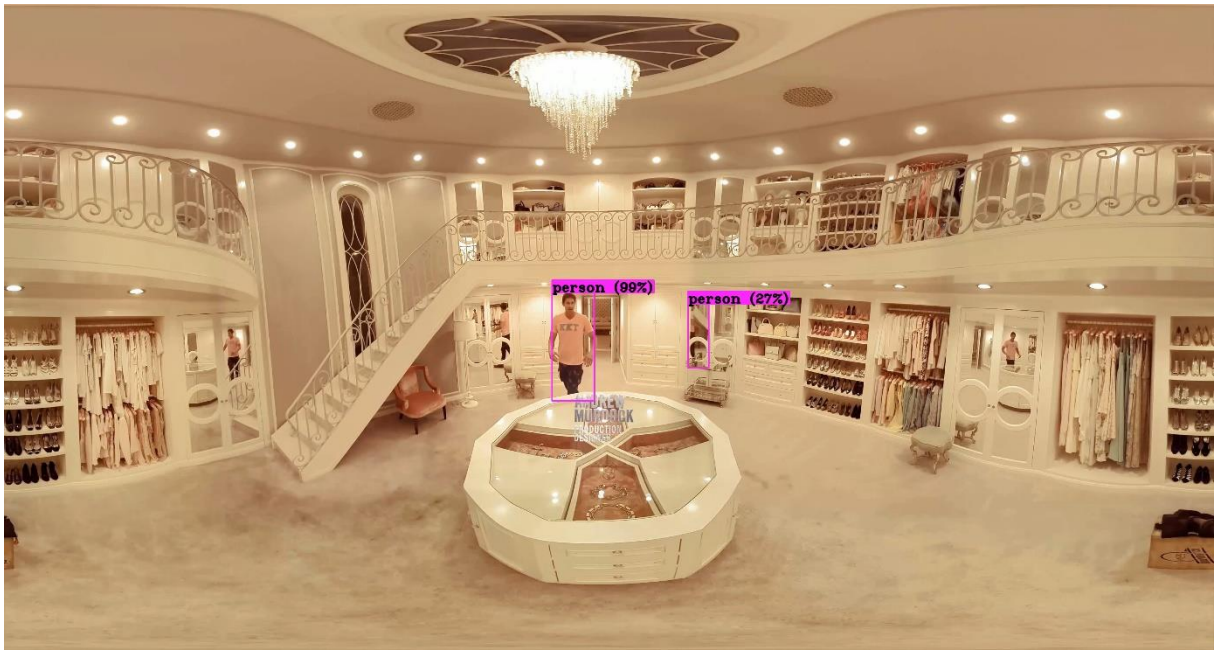

Figura 36. Desempenho do YOLOv3 durante o vídeo 3.

Figura 37. Desempenho do YOLOv3 durante o vídeo 5.

Figura 38. Desempenho do YOLOv3 durante o vídeo 6.

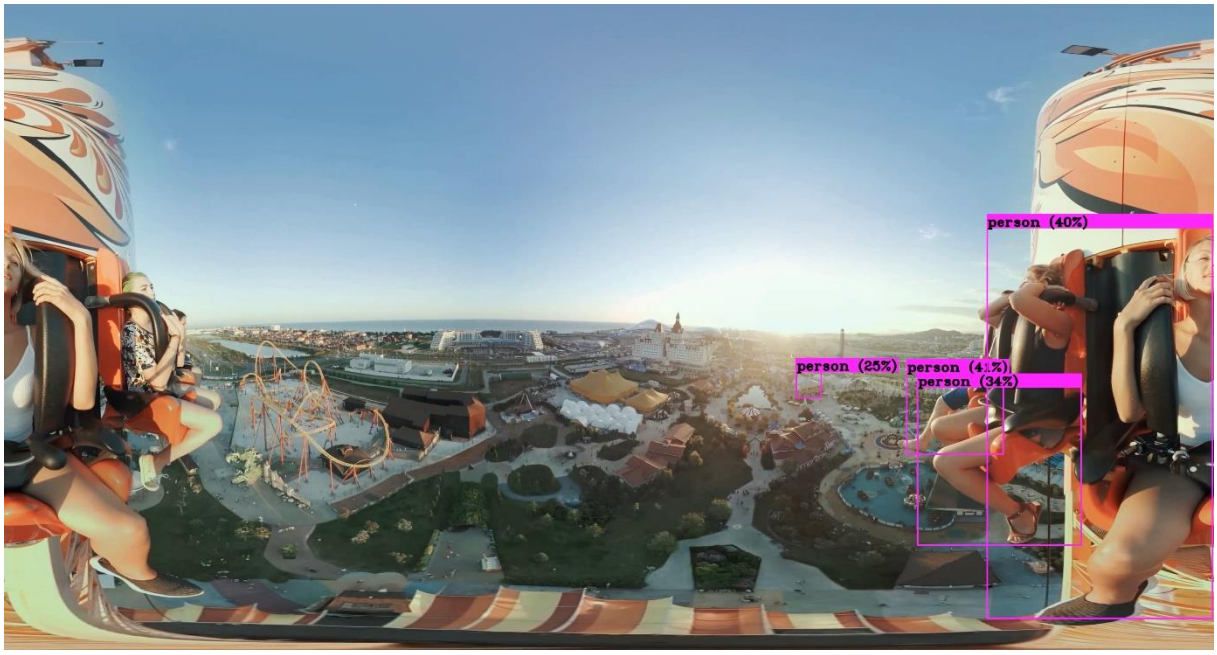

Figura 39. Desempenho do YOLOv3 durante o vídeo 17.

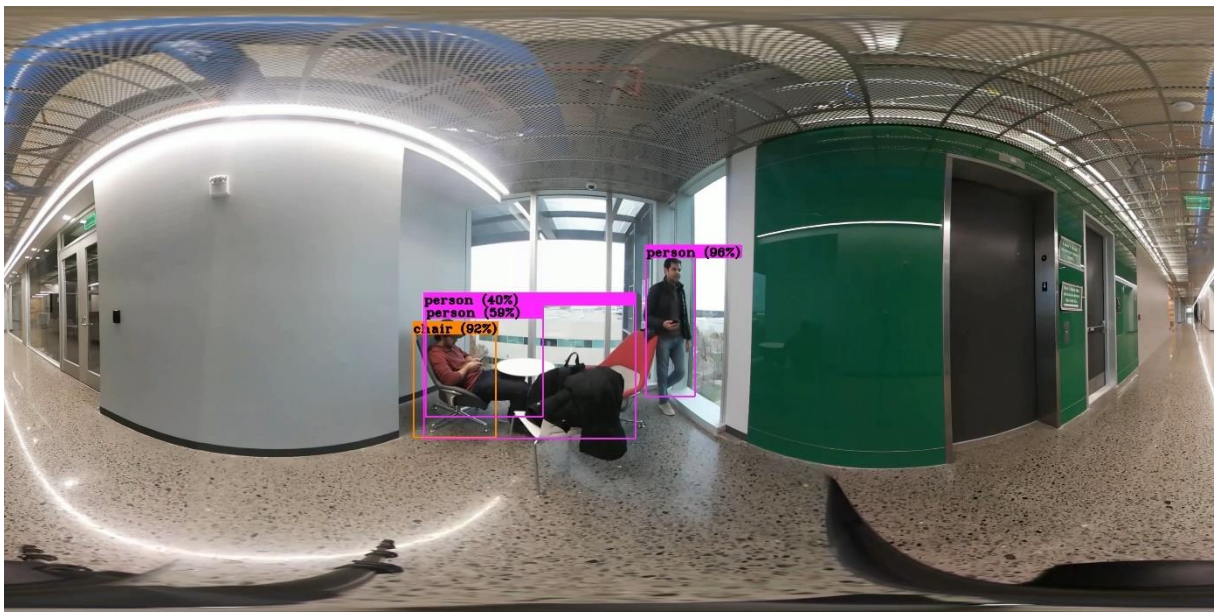

Figura 40. Detecção de objetos no vídeo 4.

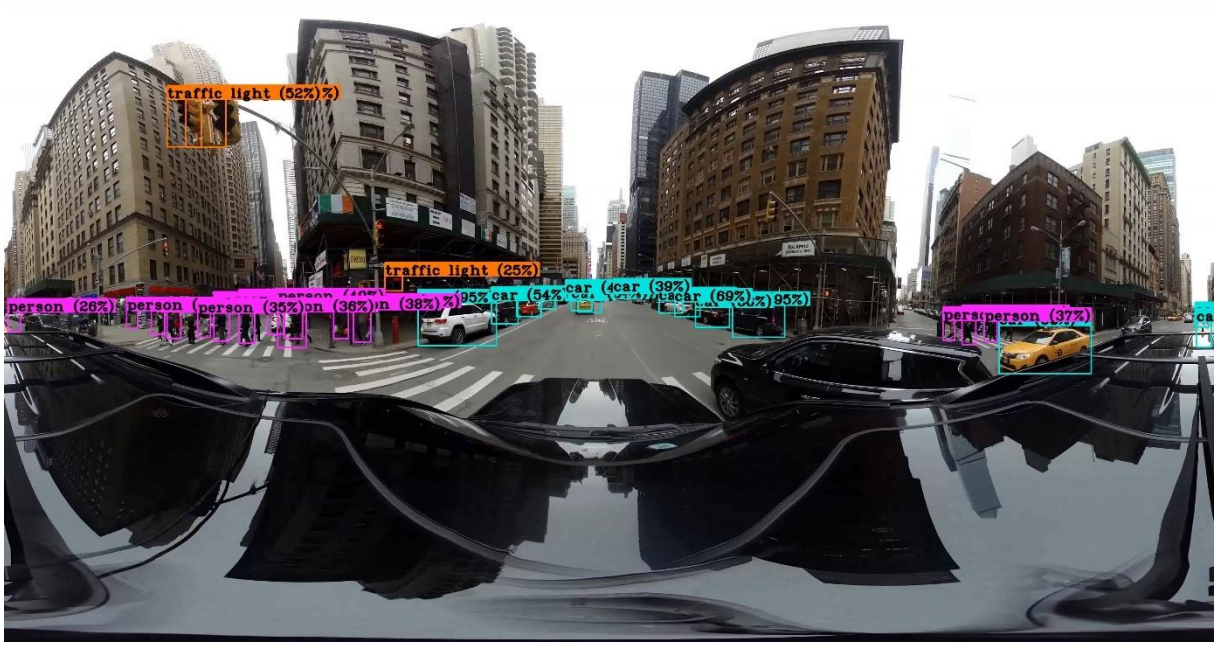

Figura 41. Detecção de objetos no vídeo 12.

5. CONCLUSÃO

Os experimentos mostraram que os algoritmos citados podem funcionar minimamente mesmo quando testados com vídeos 360°, apesar de apresentarem resultados aquém do que poderiam mostrar. Para o GBVS, de modo geral, confirmaram-se as suspeitas iniciais de que haveria problemas com as distorções inerentes à projeção equirretangular. O conteúdo de cada vídeo e os seus contextos também foram determinantes para o desempenho do algoritmo, uma vez que os participantes apresentaram uma tendência forte a concentrar suas atenções na direção do centro do vídeo.

O YOLOv3, por sua vez, só não conseguiu identificar nenhum objeto em quatro vídeos: 1, 2, 7 e 14. Todos esses vídeos mostravam ou paisagens naturais, ou um elevador como no caso do 14. Além do fato do YOLOv3 não estar preparado para analisar vídeos 360°, esses vídeos em particular não apresentaram classes de objetos características do *Imagenet*, que é o banco de imagens no qual o YOLOv3 foi treinado. Os únicos vídeos em que o YOLOv3 apresentou um desempenho muito bom foram os vídeos 12 e 4, onde praticamente todos os objetos foram identificados corretamente, desde transeuntes nas ruas até semáforos. Em todos os outros vídeos, o YOLOv3 identificou erroneamente algum objeto em cena, apesar de ter encontrado algo. Não é um problema incomum, uma vez que o YOLOv3 pode errar até mesmo quando colocado para analisar imagens estáticas. No entanto, muitos erros se deveram às distorções causadas pela projeção equirretangular dos vídeos, já que muitos desses erros foram localizados nas regiões de bordas dos vídeos (polos) da tela. Em vários vídeos as distorções fizeram o YOLOv3 encontrar apenas um único objeto grande no vídeo, como a previsão do bolo no vídeo 7 ou a previsão da motocicleta no vídeo 9.

Alguns aspectos precisam ser repensados antes que se possa aproveitar o potencial do YOLOv3 e do GBVS para estimar a saliência visual de vídeos 360°. Em alguns vídeos, como os 12 e 1, os dois algoritmos pareceram se portar de forma complementar. Porém, é necessário descobrir se os objetos que o YOLOv3 identifica podem ser considerados pontos salientes dentro de um determinado contexto, mesmo quando o algoritmo de saliência conclui que não são.

Deixa-se como sugestão para trabalhos futuros:

- Testar outros algoritmos de atenção visual;
- Compensar as deformações causadas pela projeção Equirretangular;
- Testar o desempenho dos algoritmos utilizando outras métricas, além da AUC_JUDD;
- Quantificar o desempenho do YOLOv3 com métricas de desempenho;
- Realizar testes com as áreas identificadas pelo YOLOv3 e os mapas de saliência, para verificar se existe uma correlação entre o objeto identificado e a atenção humana;
- Realizar testes com bibliotecas completas de vídeos 360º.

REFERÊNCIAS BIBLIOGRÁFICAS

Alchieri, E. **Grafos**. 2019. Disponível em <https://cic.unb.br/~alchieri/disciplinas/graduacao/ed/ed.html>. Acesso em 14 fev 2020

Biggs, N.; Lloyd, E.; WILSON, R. **Graph Theory**, 1736-1936, Oxford University Press, (1986).

Bishop, C. **Pattern Recognition and Machine Learning**. 1. ed. Springer (2006).

Borji, A.; Itti, L. **State-of-the-art in visual attention modeling**. IEEE Transactions on Pattern Analysis and Machine Intelligence, IEEE, v.35, p.185-207, 2013.

Borji, A.; Sihite, D. N.; Itti, L. **What/where to look next? Modeling top-down visual attention in complex interactive environments.** IEEE Transactions on Systems, Man, and Cybernetics: Systems, IEEE, v.44, p.523-538, 2014.

Chun, Marvin M. **Contextual Guidance of Visual Attention**. In: ITTI, Laurent; REES, Geraint; TSOTSOS, John K. (ed.). Neurobiology of Attention, Academic Press, 2005, p. 246-250. cap. 40.

Corbetta, M. **Frontoparietal cortical networks for directing attention and the eye to visual locations: identical, independent, or overlapping neural systems?** Proceedings National Academy of Sciences, v. 95, n. 3, p. 831–838, 1998.

El-Ganainy, T.; Hefeeda, M. **Streaming virtual reality content.** 2016. Disponível em: <http://arxiv.org/abs/1612.08350>. Acesso em 12 jul. 2019.

Goncharov, I. **QUICKEST_YOLOv3_On_Colab**. Disponível em <https://github.com/ivangrov/YOLOv3-GoogleColab>. Acesso em 31 jul. 2020.

Harel, J.; Koch, C.; Perona, P. **Graph-Based Visual Saliency**. In: PROCEEDINGS OF NEURAL INFORMATION PROCESSING SYSTEMS (NIPS' 06), 2006. Anais… Cambridge, MA, USA: MIT Press, 2006. p. 545-552.

Henderson, J; Hollingworth, M. **A High-level scene perception**. Annual Review of Psychology, v.50, p.243-271, 1999.

IMARC Group. Virtual Reality (VR) Gaming Market 2020-25 | Global Size, Analysis, Prices and Industry Trends. Disponível em < https://www.marketwatch.com/press-release/virtual-reality-vr-gaming-market-2020-25-global-size-analysis-prices-and-industry-trends-2020-05-25?tesla=y >. Acesso em: 31 jul. 2020.

Itti, L.; Koch, C. **Computational modelling of visual attention.** Nature Reviews Neuroscience, v. 2, p. 193–203, 2001.

Itti, L.; Koch, C.; Niebur, E. **A model of saliency-based visual attention for rapid scene analysis.** IEEE Transactions on Pattern Analysis and Machine Intelligence, IEEE, v. 20, n. 11, p. 1254-1259, 1998.

James, W. **The principles of psychology**. New York, NY USA: Henry Holt and Company, 1890. v. 2.

Judd, T. *et al*. **Learning to predict where humans loo**k. In: IEEE. INTERNATIONAL CONFERENCE ON COMPUTER VISION (ICCV). Kyoto, 2009. Anais... p.2106-2113.

Judd, T. **Understanding and Predicting Where People Look in Images.** Tese (Doutorado)—Massachusetts Institute of Technology, 2011.

Lin, Tsung-Yi *et al.* **Feature pyramid networks for object detection.** In: PROCEEDINGS OF THE IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION. Anais… p. 2117–2125, 2017

Lin, Tsung-Yi *et al.* **Microsoft Coco: Common objects in context.** In: EUROPEAN CONFERENCE ON COMPUTER VISION. Anais… p. 740–755. Springer, 2014.

Lo, Wen-Chih *et al.* 2017. 360° **Video Viewing Dataset in Head-Mounted Virtual Reality.** In: PROCEEDINGS OF THE 8TH ACM ON MULTIMEDIA SYSTEMS CONFERENCE (MMSYS'17). Anais eletrônicos… ACM, New York, NY, USA. p. 211-216. Disponível em: <https://doi.org/10.1145/3083187.3083219>. Acesso em 12 jul. 2019.

Marcato, André Luís Marques. **Introdução à Robótica, aula 3.** 2010. Disponível em <http://www.ufjf.br/andre_marcato/files/2010/08/Rob%C3%B3tica_Aula_3.ppt>. Acesso em: 10 fev. 2020.

Pena, Rodrigo Cerqueira Gonzalez. **Métricas de qualidade de vídeo com *características top-down* de atenção visual.** 2014. vi, 61 f., il. Monografia (Bacharelado em Engenharia Elétrica)—Universidade de Brasília, Brasília, 2014.

Redmon, J *et al.* **You only look once: Unified, real-time object detection.** 2016. Disponível em: <https://pjreddie.com/publications/>. Acesso em 12 jul. 2019.

Redmon, J. **Darknet: Open Source Neural Networks in C.** 2013-2016. Disponível em: <https://pjreddie.com/darknet/yolo/>. Acesso em 11 jul. 2019.

Redmon, J.; Farhadi, A. **Yolo9000: Better, faster, stronger.** In: COMPUTER VISION AND PATTERN RECOGNITION, (CVPR), 2017. Anais… IEEE CONFERENCE ON. p. 6517–6525. IEEE, 2017. Disponível em: <https://pjreddie.com/publications/>. Acesso em 12 jul. 2019.

Redmon, J.; Farhadi, A. **YOLOv3: An Incremental Improvement.** 2018. Disponível em: < https://pjreddie.com/publications/>. Acesso em 12 jul. 2019.

Taghavi, A. *et al.* **A taxonomy and dataset for 360° videos.** 2019. Disponível em: <https://arxiv.org/abs/1905.03823>. Acesso em: 13 fev. 2020.

Torralba, A. *et al.* **Contextual guidance of eye movements and attention in real-world scenes: The role of global features on object search.** Psychological Review, American Psychological Association, 2006.

Yarbus, A.L. **Eye Movements and Vision**, New York, NY USA: Plenum Press, 1967.