Pontifícia Universidade Católica
do Rio de Janeiro

Susana de Souza Bouchardet

Um estudo de uso de segmentação de objetos para a aplicação de técnicas de Video Inpainting

Dissertação de Mestrado

Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre pelo Programa de Pós–graduação em Informática da PUC-Rio.

Orientador: Prof. Hélio Cortês Vieira Lopes

Rio de Janeiro
Maio de 2021

Pontifícia Universidade Católica
do Rio de Janeiro

Susana de Souza Bouchardet

Um estudo de uso de segmentação de objetos para a aplicação de técnicas de Video Inpainting

Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre pelo Programa de Pós–graduação em Informática da PUC-Rio. Aprovada pela Comissão Examinadora abaixo:

Prof. Hélio Cortês Vieira Lopes
Orientador
Departamento de Informática – PUC-Rio

Dr. Cassio Freitas Pereira de Almeida
ENCE

Prof. Bruno Feijó
Departamento de Informática – PUC-Rio

Dr. Guilherme Gonçalves Schardong
Departamento de Informática – PUC-Rio

Rio de Janeiro, 11 de Maio de 2021

Susana de Souza Bouchardet

Graduada em engenharia da Computação pela Universidade Estatual do Rio de Janeiro (Rio de Janeiro - Brasil) em 2017

Ficha Catalográfica

Bouchardet, Susana de Souza

Um estudo de uso de segmentação de objetos para a aplicação de técnicas de Video Inpainting / Susana de Souza Bouchardet; orientador: Hélio Cortês Vieira Lopes. – 2021.

72 f: il. color. ; 30 cm

Dissertação (mestrado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 2021.

Inclui bibliografia

1. Informática – Teses. 2. Deep Learning – Teses. 3. Visão Computacional. 4. Machine Learning. 5. Video Inpainting. 6. Video Object Segmentation. I. Hélio Cortês Vieira Lopes. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

Aos meus professores, que sempre me incentivaram a continuar estudando.

Agradecimentos

Primeiramente gostaria de agradecer a meu orientador Hélio Lopes, por me guiar e ensinar durante esse período.

Gostaria também de agradecer meu marido Pedro Otavio que me apoiou e ajudou durante meu período de mestrado. Ainda gostaria de agradecer a meus pais, Sandra e Douglas, que junto com meu irmão Daniel, me apoiaram durante esses anos de estudo.

Aos meus amigos Arnour Sabino, Gabriel Cardoso e Diogo Munaro que acreditaram na minha capacidade e me apoiaram nos desafios do mestrado junto com o trabalho.
O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

Resumo

Bouchardet, Susana de Souza; Hélio Cortês Vieira Lopes. **Um estudo de uso de segmentação de objetos para a aplicação de técnicas de Video Inpainting**. Rio de Janeiro, 2021. 72p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Nos últimos anos tem ocorrido um notável desenvolvimento de técnicas de *Image Inpainting*, entretanto transpor esse conhecimento para aplicações em vídeo tem se mostrado um desafio. Além dos desafios inerentes a tarefa de *Video Inpainting* (*VI*), utilizar essa técnica requer um trabalho prévio de anotação da área que será reconstruída. Se a aplicação do método for para remover um objeto ao longo de um vídeo, então a anotação prévia deve ser uma máscara da área deste objeto *frame* a *frame*. A tarefa de propagar a anotação de um objeto ao longo de um vídeo é conhecida como *Video Object Segmentation* (*VOS*) e já existem técnicas bem desenvolvidas para solucionar este problemas. Assim, a proposta desse trabalho é aplicar técnicas de *VOS* para gerar insumo para um algoritmo de *VI*. Neste trabalho iremos analisar o impacto de utilizar anotações preditas no resultado final de um modelo de *VI*.

Palavras-chave

Visão Computacional; Machine Learning; Video Inpainting; Video Object Segmentation.

Abstract

Bouchardet, Susana de Souza; Hélio Cortês Vieira Lopes (Advisor). **A study of the use of object segmentation for the application of Video Inpainting techniques**. Rio de Janeiro, 2021. 72p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

In recent years there has been a remarkable development of Image Inpainting techniques, but using this knowledge in video application is still a challenge. Besides the inherent challenges of the Video Inpainting (VI) task, applying this technique requires a previous job of labeling the area that should be reconstructed. If this method is used to remove an object from the video, then the annotation should be a mask of this object's area frame by frame. The task of propagating an object mask in a video is known as Video Object Segmentation (VOS) and there are already well developed techniques to solve this kind of task. Therefore, this work aims to apply VOS techniques to create the inputs for an VI algorithm. In this work we shall analyse the impact in the result of a VI algorithm when we use a predicted annotation as the input.

Keywords

Computer Vision; Machine Learning; Video Inpainting; Video Object Segmentation.

Sumário

Lista de figuras

Lista de Abreviaturas

VOS – *Video Object Segmentation*

VI – *Video Inpaiting*

PReMVOS – *Proposal Refinment Merging Video Object Segmentation*

OSVOS – *One-Shot Video Object Segmentation*

FCN – *Fully Convolutional Network*

CNN – *Convolutional Neural Network*

O temor do Senhor é o princípio do conhecimento, mas os insensatos desprezam a sabedoria e a disciplina.

Salomão, *Bíblia Sagrada.*

1
Introdução

A área de Visão Computacional tem crescido ao longo dos últimos anos, e junto a esse crescimento novos desafios tem surgido para serem solucionados. Em [1], Ebdelli define *Video Inpainting* (VI) como um campo de estudo de Visão Computacional que tem como objetivo preencher regiões em uma sequência de *frames* em um vídeo usando informações espaço-temporais dos pixels ao redor da região. Essa região pode ser definida de maneira a corrigir imperfeições em vídeos, ou remover um objeto do vídeo (logotipos, textos, pessoas, etc.). O principal desafio dessa tarefa é preencher a região desejada de maneira a gerar um resultado convincente ao ser analisado por pessoas.

Vamos nos concentrar nesse trabalho na tarefa de remover um objeto de um vídeo. Um desafio de usabilidade dessa técnica é que a região do objeto que deseja-se remover deve estar marcada em todos os *frames*. Isso pode tornar a técnica pouco prática. Esse panorama nos impulsiona a trabalhar em possíveis soluções para melhorar a usabilidade de técnicas de VI.

1.1
Objetivo e Proposta

O objetivo desse trabalho é avaliar técnicas para melhorar a usabilidade de modelos de *Video Inpainting* facilitando a etapa de marcação da região de um objeto. Para isso, propomos a utilização de técnicas de *Video Object Segmentation* (VOS). As técnicas de VOS fazem parte de uma área de estudo de Visão Computacional com objetivo de segmentar objetos em vídeos. Existem técnicas VOS que se dedicam a resolver problemas de diferentes especificações, como Yal *et al.* descreve em [2]. Entretanto, o comum a essas técnicas é que o resultado é uma máscara binária por *frame* que classifica se cada pixel da imagem faz ou não parte do objeto a ser segmentado, como é mostrado na Figura 1.1.

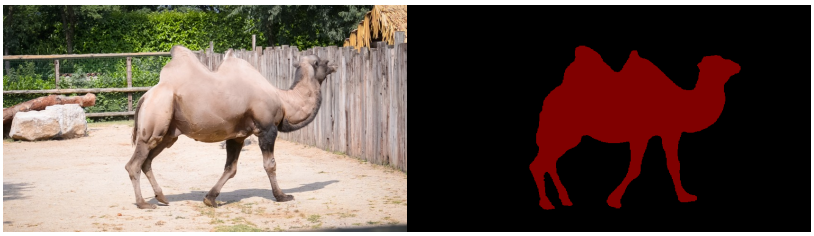


Figura 1.1: A esquerda um exemplo de um *frame* de entrada em um método de VOS; a direita o resultado esperado do método de VOS.

Dentre as técnicas de VOS, vamos nos concentrar nas técnicas de VOS Semi-Supervisionado. Essas técnicas buscam encontrar a região de um certo objeto ao longo de um vídeo usando a anotação da região desse objeto em *frames* de referência.

As técnicas de VOS Semi-Supervisionado tem avançado nos últimos anos e recebido atenção. Atualmente existe um desafio anual promovido pelo YouTube com objetivo de desenvolver técnicas de VOS. Uma das categorias é dedicada a VOS Semi-Supervisionado. Como fruto desse desafio, Luiten *et al.* mostra em [4] que em 2018 submeteu um modelo chamado PReMVOS (*Proposal Refinement Merge Video Object Segmentation*). Esse modelo foi o vencedor dessa edição e se demostrou uma alternativa interessante para solucionar o problema proposto nesse trabalho.

Além do modelo PReMVOS, decidimos incluir em nosso trabalho um modelo tradicional conhecido como OSVOS (*One-Shot Video Object Segmentation*). Em seu trabalho [5], Caelles *et al.* desenvolvem um modelo para solucionar problemas de VOS Semi-Supervisionado que se tornou referência por ter superado o estado-da-arte na época de sua publicação. Por isso, achamos interessante incluir esse modelo em nossa comparação.

Para solucionar o problema de VI decidimos utilizar o modelo proposto por Kim *et al.* em [6] por ser um dos trabalhos mais recentes da técnica e ter produzido resultados convincentes.

1.2 Organização do Trabalho

O restante do texto é dividido da seguinte maneira. No Capítulo 2 nos dedicamos a aprofundar conceitos básicos relacionados tarefa de VOS e a tarefa de VI. Além das tarefas nos aprofundamos também nas técnicas mais comuns na geração de modelos que resolvem essas tarefas. No Capítulo 3 nos aprofundamos nos trabalhos que serviram de base pra o desenvolvimento do modelo proposto. No Capítulo 4 iremos mostrar o modelo proposto e falare-

mos brevemente de como avaliar o modelo. Em seguida, no Capítulo 5, vamos nos aprofundar nas métricas de avaliação e exibir os resultados. Por fim, encerramos fazendo considerações finais e apontando possíveis desenvolvimentos futuros no Capítulo 6

2
Conceitos Básicos

Nesse Capítulo iremos nos aprofundar na taxonomia de modelos de VOS. Além disso iremos nos aprofundar em alguns conceitos básicos utilizados na construção dos modelos escolhidos para o desenvolvimento deste trabalho.

2.1
Taxonomia dos modelos VOS

Para compreendermos melhor o problema que VOS se propões a resolver, devemos entender duas outras tarefas da área de Visão Computacional:*Image Segmentation* e *Object Tracking*

A tarefa de *Image Segmentation* tem como objetivo segmentar a imagem de acordo com seu conteúdo. O resultado esperado para essa tarefa é uma classificação por pixel para identificar os pixels que compõem a região de interesse. Esse resultado também é chamado de máscara, já que ele descreve a imagem por segmentos. Essa tarefa ainda pode segmentar um ou mais objetos diferentes. Nesse caso a máscara gerada deve classificar cada pixel identificando a qual objeto ele pertence. A Figura 2.1 mostra um exemplo da tarefa de *Image Segmentation* em dois casos: *single-class* (com apenas um objeto a ser segmentado) e *multi-class* (com mais de um objeto a ser segmentado).

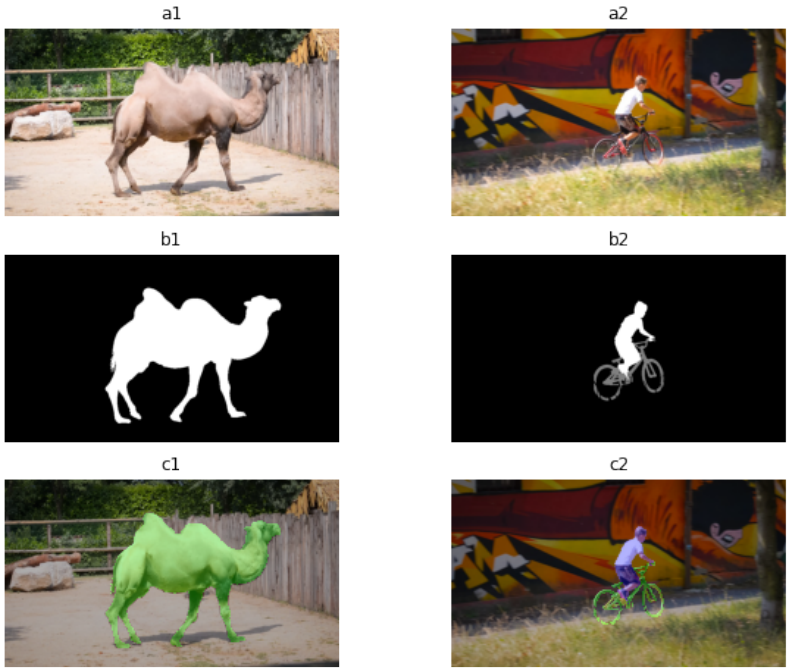


Figura 2.1: Exemplo do resultado esperado da tarefa de *Image Segmentation*. As imagens $a1$ e $a2$ mostram a imagem de referência para a tarefa. As imagens $b1$ e $b2$ representam as máscaras de cada imagem, sendo que a máscara $b1$ é a segmentação de apenas um objeto, enquanto a máscara $b2$ é a segmentação de 2 objetos. As imagens $c1$ e $c2$ mostram a sobreposição das máscaras em suas respectivas imagens de referência.

Por outro lado, a tarefa de *Object Tracking* se concentra em encontra a posição aproximada de um certo objeto ao longo de um vídeo. Essa marcação aproximada pode ser feita de algumas maneiras, mas a mais comum é por meio de *bounding boxes*, que representa a posição de um objeto posicionando um retângulo de maneira que o objeto esteja contido nele. A Figura 2.2 mostra a identificação de um objeto em imagens por meio de *bounding boxes* e máscaras.

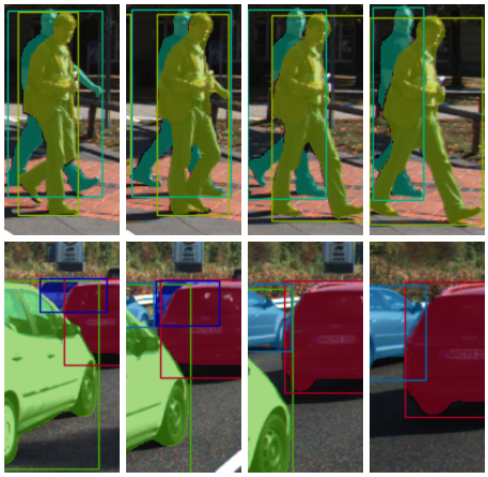


Figura 2.2: Exemplo, extraído da pesquisa [8] de Voigtlaender *et al.*, onde é possível notar a diferença entre *bounding boxes* e máscaras.

A partir dos conceitos de *Image Segmentation* e *Object Tracking*, podemos então definir que o objetivo da tarefa de VOS é fazer a segmentação de um ou mais objetos ao longo de um vídeo. Para isso, esperamos como resultado uma máscara por *frame*, que identifique os pixels dos objetos de interesse.

Em sua pesquisa [2], Yao *et al.* se aprofunda na taxonomia dos modelos que resolvem tarefas de VOS. A Figura 2.3 mostra as classes de modelos de VOS.

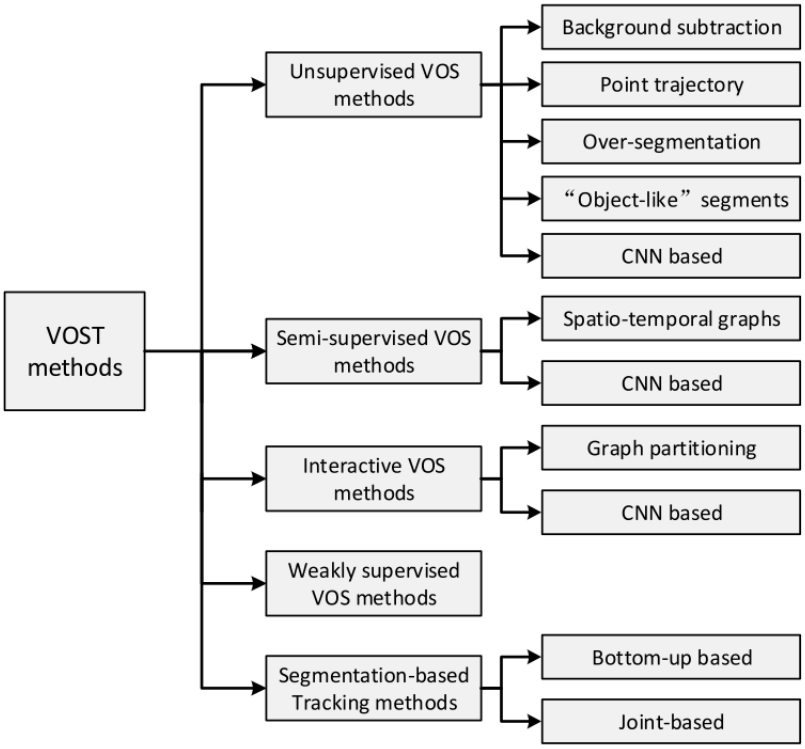


Figura 2.3: Taxonomia dos modelos de VOS. Extraído de [2]

Iremos descrever abaixo brevemente os tipos de modelos VOS e iremos nos aprofundar nos modelos VOS Semi-supervisionados.

2.1.1 VOS não-supervisionado

Os modelos VOS não-supervisionados recebem como entrada apenas o vídeo a ser processado. Com base na aparência e na movimentação, esses modelos tentam segmentar os objetos ao longo do vídeo.

Uma das premissas desses métodos é que os objetos a serem segmentados tem características de movimentação diferente do segundo plano, então um tipo de abordagem é a subtração do segundo plano. Um exemplo de trabalho é [31], de Han *et al.*, que propõe um algoritmo de subtração de segundo plano de vídeos.

2.1.2 VOS fracamente supervisionados

Os modelos VOS fracamente supervisionados tentam aprender a identificar uma certo objeto a partir de um grupo de vídeos que contem esse objeto

e que não tem nenhuma anotação. Esses modelos se tornam capazes de segmentar o objeto para o qual foram treinados.

Uma variação muito comum desse tipo de modelo, é explicitando a classe do objeto que deve ser segmentado no vídeo. Assim, o modelo é treinado para identificar várias classes de objetos, e para isso basta receber, além do vídeo, o identificador da classe à ser segmentada como entrada (*e.g.* se o modelo receber em treinamento vídeos que contêm aviões como a classe objetivo, então, esse modelo está preparado para identificar vídeos da classe "avião").

Glenn *et al.* em [33] propõe um modelo deste tipo criado a partir de uma grande base de dados fracamente classificados (*i.e.* a base de treinamento não conta com as máscaras dos objetos, apenas a classe do objeto que cada vídeo contém).

2.1.3 VOS interativos

Este tipo de modelo, ao contrário dos que falamos até agora, permite uma interação direta do usuário durante a tarefa de segmentação. A interação do usuário é permitida para identificar regiões que fazem parte do objeto ou que não fazem parte. Assim o resultado do modelo pode se corrigir utilizando essa informação.

Em seu trabalho Maninis *et al.* [32] apresentam uma técnica de segmentação a partir da marcação dos pontos extremos do objeto (*i.e.* ponto que delimitam o mais a esquerda, direita, acima e abaixo do objeto). A premissa deste modelo é a iteração do usuário, para demarcar esses pontos.

2.1.4 *Tracking* baseado em segmentação

Outros modelos de VOS trabalham utilizando informações de movimento e aparência para encontrar uma máscara binária que marque os pixels que pertencem a representação de um certo objeto. Isso costuma ser computacionalmente caro.

Por outro lado, a tarefa de *object tracking*, como já citamos no início da seção 2.1, utiliza como anotação normalmente *bounding boxes*, que são estruturas mais simples do que as máscaras binárias. Entretanto, esses métodos podem ter dificuldade para identificar objetos que sofrem deformações ao longo do vídeo, ou até mesmo identificar objetos sobrepostos. Para superar esses desafios surgiram modelos que integram conceitos de segmentação a modelos de *object tracking*.

O modelo proposto por Wang *et al.* em [34] é um exemplo de como os conceitos de *object tracking* e segmentação podem ser incorporados para criação de modelos mais rápidos.

2.1.5 VOS Semi-supervisionado

Ao contrário das ultimas seções, iremos nos aprofundar mais na taxonomia dos métodos de **VOS Semi-supervisionado**, pois os métodos escolhidos para compor esse trabalho fazem parte deste grupo.

Os modelos VOS Semi-supervisionado utilizam como insumo um ou mais *frames* anotados como referência, como é mostrado no diagrama da Figura 2.4. Esta figura mostra, como exemplo, um modelo que utiliza apenas o primeiro *frame* como referência.

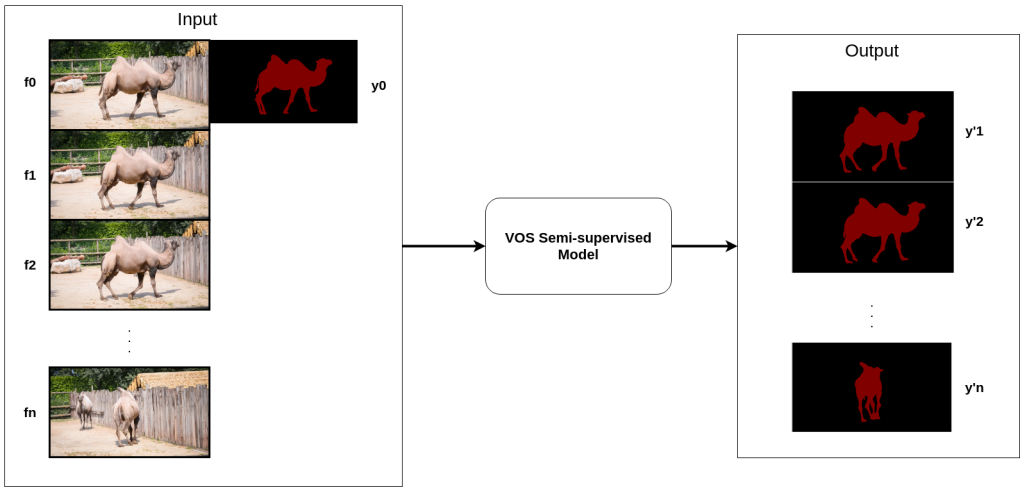


Figura 2.4: Diagrama representando a entrada e saída esperadas de um modelo VOS Semi-supervisionado

Os modelos da família dos VOS Semi-supervisionados ainda se dividem em dois grupos: Os baseados em grafos espaço-temporais e os baseados em *Convolutional Neural Network* (CNN). Nas próxima seções iremos explorar as características desses grupos.

2.1.5.1 Grafos espaço-temporais

Esse conjunto de métodos propões modelar o problema como grafos espaço-temporais que podem ser resolvidos a partir de atributos clássicos, como por exemplo: fluxo óptico, bordas e aparência.

Esse modelo conta com duas características principais: A estrutura do grafo e as conexões temporais.

A estrutura do grafo é como cada modelo irá construir um grafo sobre o objeto. Esses modelos podem definir os nós desse grafo a partir de pixels, *super-pixels* ou *object-patches*. Assim cada *frame* tem um grafo construído sobre ele, além disso cada nó desse grafo possui um *label* que define se ele faz parte do objeto de interesse ou não.

As conexões espaço-temporais se referem à como o modelo irá estimar as conexões temporais do grafo, ou seja, como ele irá tratar a propagação de *labels*. Essa estimativa das conexões pode ser feita utilizando, por exemplo, similaridade de aparência, proximidade ou características do fluxo óptico.

Em seu trabalho Lazema *et al.* [17] apresenta um modelo para segmentação em vídeo aplicando grafos espaço-temporais. A Figura 2.5 mostra de maneira simplificada a estrutura e as conexões espaço temporais que foram tratadas nos parágrafos anteriores.

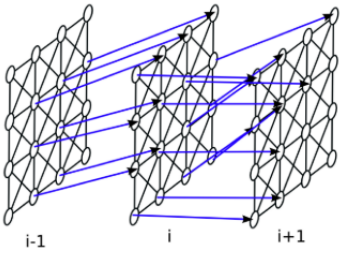


Figura 2.5: Exemplo de grafo espaço-temporal extraído de [17].

A Figura 2.5 representa a estrutura do grafo nos *frames* $i-1$, i e $i+1$. As setas azuis representam as estimativas de conexões espaço-temporais, apontando a movimentação prevista de uma certa região entre os *frames*.

2.1.5.2 CNN

Modelos utilizando CNN tem tido grande sucesso em resolver tarefas de segmentação de imagens estáticas. Para transpor essa técnica para vídeo, existem dois tipos de abordagens: uma baseada em detecção e outra baseada em movimento.

Os **modelos baseados em detecção** não utilizam informações temporais. Esse tipo de modelo se baseiam em aprender as características do objeto com base nos *frames* anotados para então encontra-lo em outros *frames*. Esse tipo de modelo utiliza a técnica de *fine-tuning* para adaptar um modelo base

aos *frames* anotados. Na secção 2.2 iremos explicar mais detalhadamente a técnica de *fine-tuning*.

Nesse trabalho utilizamos o modelo OSVOS, que é um modelos baseados em detecção. Iremos nos aprofundar em sua implementação na secção 3.1.

Os **modelos baseados em movimento** buscam manter a coerência temporal da movimentação do objeto. Para isso, muitos recorrem a utilização do fluxo óptico dos pixels. Na seção 2.3 explicamos o conceito do fluxo óptico e o motivo dele ser utilizado em modelos que buscam manter coerência temporal. Este tipo de modelo acaba sendo abordado como um problema de propagação de máscara, a partir dos *frames* anotados.

Neste trabalho utilizamos o modelo PReMVOS, que é classificado como um modelo baseados em movimento. Iremos nos aprofundar na implementação desse modelo na secção 3.2

Em sua pesquisa Yao *et al.* [2] pontua as vantagens e os possíveis problemas de ambas as abordagens. Em modelos baseados em detecção é possível perder a coerência temporal em partes do vídeo, uma vez que esses métodos não utilizam atributos temporais. Entretanto, caso haja erros em um certo ponto do vídeo, esse método não os propaga para *frames* subsequentes.

Já em modelos baseados em movimento, quando há a propagação errada da máscara em um certo ponto do vídeo esse erro é propagado para os *frames* subsequentes. A propagação errada da máscara neste tipo de método costuma ocorrer em vídeos com movimentos muito bruscos ou que contêm oclusão do objeto. Por outro lado, os resultados deste tipo de método costumam manter uma boa coerência temporal entre os *frames*, que é uma dos principais desafios na solução deste problema.

2.2 Fine-tuning

Como relatado em [5], o modelo OSVOS utiliza uma técnica de *transfer learning* chamada *fine-tuning*. A premissa de técnicas de *transfer learning* é reutilizar o aprendizado de uma tarefas para outra. Para modelos baseados em rede neural *transfer learning* pode ser feito a partir da técnica de *fine-tuning*. Em [9], Li *et al.* define *fine-tuning* como uma técnica que reutiliza camadas de uma rede que resolve um problema similar ao objetivo.

Supondo que há uma rede neural pré-treinada para resolver um problema similar ao problema objetivo (*e.g.* o problema de selecionar carros é similar ao problema de selecionar caminhões). Vamos ainda considerar que θ_o são os parâmetros desta rede neural.

Ao criar uma nova rede neural que utiliza a técnica de *fine-tuning* algumas das camadas e parâmetros θ_o são copiados da rede neural pré-treinada e algumas camadas novas com parâmetros θ_n são adicionadas. A Figura 2.6 mostra um exemplo de arquitetura de uma rede neural que utiliza *fine-tuning*.

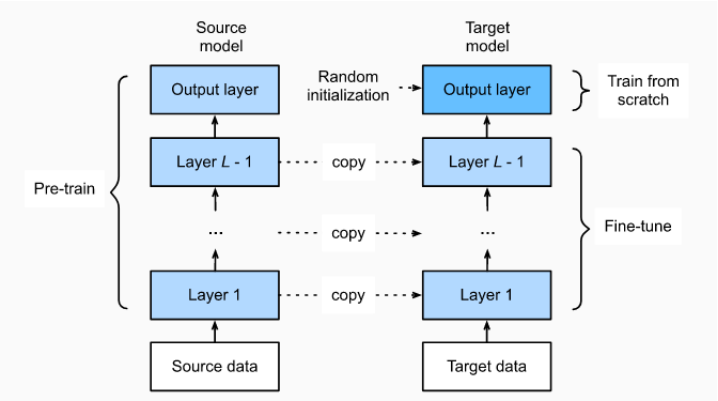


Figura 2.6: Exemplo de arquitetura de rede neural utilizando *fine-tuning*. Extraído do livro [10]

O treinamento desse tipo de rede é diferente de uma rede tradicional já que os parâmetros θ_o são congelados e apenas os parâmetros θ_n são ajustados para reduzir o erro.

Uma consideração importante sobre a técnica é que não precisamos utilizar todos os parâmetros da rede neural pré-treinada, o que nos da liberdade de remover camadas.

2.3 Fluxo óptico

Em [18], Beauchemin *et al.* definem o fluxo óptico como uma aproximação da projeção do vetor de velocidade de uma superfície 3D em uma imagem plana capturada a partir de um sensor. Podemos dizer então que o fluxo óptico é a projeção do vetor de velocidade de um certo objeto em um vídeo.

A hipótese inicial a qual se baseia a medida do fluxo óptico pode ser descrita como

$$I(x, y, t) \approx I(x + \delta x, y + \delta y, t + \delta t) \tag{2-1}$$

A equação 2-1 diz que a intensidade de um pixel na posição (x, y) no tempo t deve ser aproximadamente a mesma que um pixel na posição $(x + \delta x, y + \delta y)$ no tempo $t + \delta t$, onde δx e δy são, respectivamente, os deslocamentos de x e y após um tempo δt.

Essa premissa assume que os pixels de objeto ao longo de um vídeo não sofrem drásticas mudanças de intensidade.

Em [18] é mostrada a expansão da equação 2-1 em uma série de Taylor, que após ser trabalhada chega a um formato

$$\nabla I.\mathrm{v} + I_t = 0 \tag{2-2}$$

Na equação 2-2 é importante compreender que a intensidade é interpretada como uma função $I(x, y, t)$. Em outras palavras, a intensidade de um pixel em um video depende da sua posição (representando pelas variáveis x e y) e do tempo (representado por t).

Ainda na equação 2-2, simbolo ∇I representa o vetor gradiente da função intensidade levando em consideração apenas a posição do pixel. Podemos então dizer que $\nabla I = (I_x, I_y)$, onde $I_x = \frac{\partial I}{\partial x}$ e $I_y = \frac{\partial I}{\partial y}$. O vetor $\mathrm{v} = (u, v)$ representa a velocidade do pixel na posição (x, y). Por fim é definido $I_t = \frac{\partial I}{\partial t}$, ou seja, I_t é uma representação de como a intensidade do pixel da posição (x, y) varia de acordo com a variação do tempo.

A equação 2-2 é conhecida como a **equação que rege (limita) o fluxo óptico**. Entretanto a equação 2-2 não é um problema bem-posto, uma vez que o objetivo é descobrir o vetor $\mathrm{v} = (u, v)$, que possui duas variáveis, e só temos uma equação linear.

Expandindo ainda a equação 2-2 temos a seguinte equação:

$$uI_x + vI_y + I_t = 0 \tag{2-3}$$

A Figura 2.7 mostra a reta formada pela equação 2-3. Além disso, a Figura 2.7 mostra o posicionamento do vetor de velocidade normal, que é perpendicular a reta.

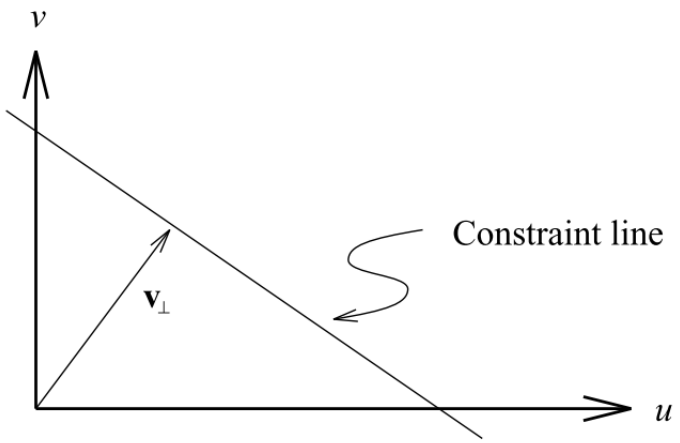


Figura 2.7: Reta que rege (limita) o fluxo óptico. Extraído de livro [18]

Além do vetor de velocidade normal, exibido da Figura 2.7, qualquer valor que atende a equação 2-3 é um potencial vetor de velocidade para o pixel (x, y) no tempo t. Existem algumas técnicas descritas por Beauchemin *et al.* em [18] para encontrar o vetor que melhor descreve a velocidade de pixels que compões um certo objeto.

Atualmente, com o avanço da utilização de redes neurais, existem modelos baseados em redes neurais profundas para encontrar o fluxo óptico de um *frame* em um vídeo. O modelo PReMVOS [4], que soluciona problemas de VOS, utiliza o fluxo óptico de um vídeo para estimar a propagação da máscara de um objeto ao longo do vídeo. Neste trabalho utilizamos o modelo PWC-Net [19] para gerar o fluxo óptico que o modelo PReMVOS utiliza.

A Figura 2.8 mostra o resultado do modelo PWC-Net aplicado para encontrar o fluxo óptico em um *frame*. A visualização em cores é feita com base no trabalho [20].

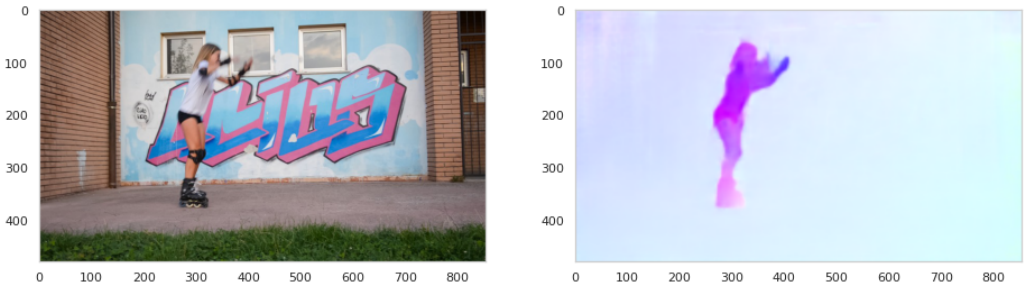


Figura 2.8: Exemplo de visualização de fluxo óptico gerado pelo modelo PWC-Net. Os vetores são representados por cores com base no trabalho [20].

Observando a Figura 2.8 é possível compreender como o fluxo óptico pode ser um atributo poderoso na tarefa de propagar uma máscara ao longo de um vídeo. Essa figura mostra como a região em movimento no vídeo tem vetores de fluxo óptico visivelmente diferente das de mais regiões da imagem.

Durante o desenvolvimento deste trabalho contribuímos para que esse tipo de visualização de fluxo óptico fosse mais acessível transformando a implementação atual em um pacote Python. Assim, foi possível importar o pacote `flow-vis` para gerar a visualização do fluxo óptico proposta em [20]. A implementação pode ser encontrada em `https://github.com/tomrunia/OpticalFlow_Visualization`, onde também está creditada nossa contribuição.

3
Trabalhos relacionados

Neste capítulo iremos falar dos trabalhos utilizados como referência no desenvolvimento deste projeto. Nas seções 3.1 e 3.2 iremos tratar os trabalhos que propões modelos que solucionam a tarefa de VOS. Em seguida, na seção 3.3 iremos discorrer sobre o trabalho que apresenta um modelo para solucionar a tarefa de VI. Por fim, na seção 3.4 iremos expor o trabalho que propõe solucionar o mesmo problema que estamos apresentando neste trabalho.

3.1
OSVOS

O trabalho [5], desenvolvido por Caelles *et al.*, apresenta o desenvolvimento do modelo OSVOS. Este modelo tem como objetivo resolver a tarefa de VOS Semi-supervisionado, e para isso se baseia em uma CNN para aprender características do objeto que se deseja segmentar.

Modelos que tem como objetivo aprender características do objeto para segmenta-lo costumam aplicar a técnica de *fine-tuning*. Para isso assumimos que iremos partir de uma rede pré-treinada que deve ser refinada para a tarefa em questão.

No modelo OSVOS, a rede pré-treinada utilizada é uma rede capaz de realizar a segmentação do primeiro plano do vídeo. Essa rede é do tipo *Fully Convolutional Network*(FCN). O treinamento dessa rede, chamado de *Offline training*, é composto por duas fases:

- *base network*: Inicialmente a rede neural é pré-treinada utilizando o *dataset* ImageNet para classificação [11]. A rede resultante deste treinamento é chamada *base network*. Este *dataset* é preparado para resolver outro tipo de problema, por isso a *base network* não é capaz de realizar bem a segmentação.
- *parent network*: Em seguida a rede é ajustada para identificar as máscaras binárias do *dataset* de treino DAVIS [7]. Nesta etapa o objetivo é que a rede aprenda noções genéricas de separação de primeiro e segundo plano da imagem.

Ao final do *Offline training* é esperado que se tenha a *parent network* que ainda não é capaz de resolver o problema de VOS Semi-supervisionado, mas separa objetos do primeiro do segundo plano.

Para chegar no objetivo existe uma etapa chamada *Online training* que é executada no momento do teste. Essa etapa aprimora a *parent network* para gerar uma *test network*. A *test network* é resultado da técnica de *fine-tuning* aplicada a *parent network* utilizando os *frames* anotados da sequencia. Assim, a *test network* é uma rede específica para aquela entrada já que aprende a identificar o objeto segmentado nos *frames* anotados.

Em seu trabalho [5] Caelles *et al.* mostram duas possibilidades de gerar a *test network*: geração totalmente online, em que o *fine-tuning* acontece no momento de teste; geração offline, onde o *fine-tuning* já prepara as *test networks* uma vez que tenha acesso aos *frames* anotados dos objetos antes da etapa de teste.

A Figura 3.1 mostra as etapas de aprimoramento da rede neural com seus respectivos resultados quando recebem um mesmo parâmetro de entrada.

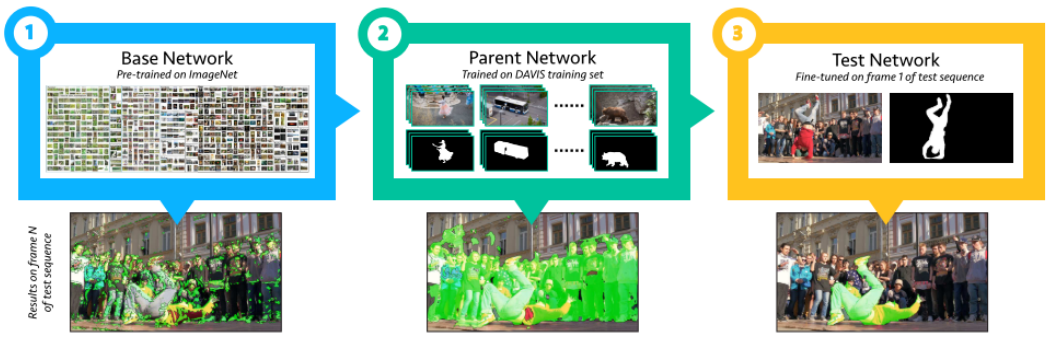


Figura 3.1: Etapas para geração do modelo OSVOS. Extraído de [5]

Ainda em [5] é falado como o resultado do modelo OSVOS pode gerar máscaras sem um encaixe perfeito no objeto. Para realizar esse ajuste fino é mencionado uma etapa para *boundary snapping* que é responsável por aperfeiçoar o encaixe da máscara resultante do modelo com o objeto. Toda essa etapa é baseada na extração das bordas da imagem. A Figura 3.2 mostra como essa etapa utiliza as bordas extraídas da imagem para criar uma versão mais refinada da máscara.

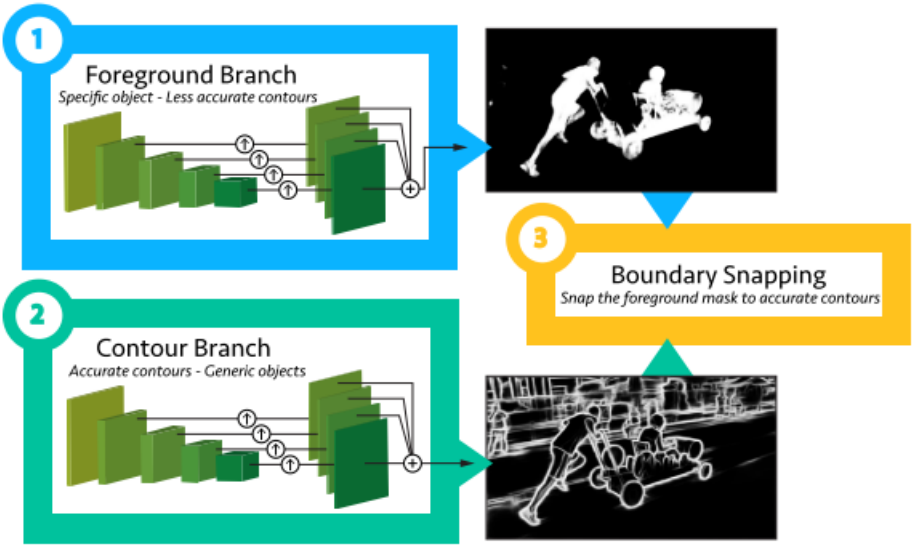


Figura 3.2: Etapas para o ajuste das bordas das máscaras geradas pelo modelo OSVOS. Extraído de [5]

Como mostrado na Figura 3.2, existem dois tipos de informação da imagem utilizadas para fazer o *boundary snapping*: (1) Os pixels do primeiro plano; (2) A borda de toda a imagem.

Para extrair a borda da imagem é utilizada uma CNN que, diferente de métodos baseados em gradiente, consegue encontrar as bordas de maneira mais acurada.

3.2 PReMVOS

Nesta seção iremos nos aprofundar no trabalho de Luiten *et al.* [4] no qual é apresentado o modelo PReMVOS. Esse modelo tem como objetivo resolver a tarefa de VOS Semi-supervisionado utilizando atributos de movimentação do vídeo. Na taxonomia definida apresentada na seção 2.1 esse modelo se encaixa como VOS Semi-supervisionado que utiliza CNN baseados em movimento.

Em seu trabalho [4], Luiten *et al.* enumera algumas diferenças de sua abordagem quando comparada com outros trabalhos. Sua abordagem se baseia em inicialmente encontrar uma proposta grosseira de máscara para então, com foco na região encontrada, o resultado ser mais refinado. Em seguida, esse resultado é utilizado, junto com o resultado de frames vizinhos, na aplicação de uma algoritmo de *merge* para manter a coerência temporal. É possível observar o resultado de cada etapa relatada na Figura 3.3.

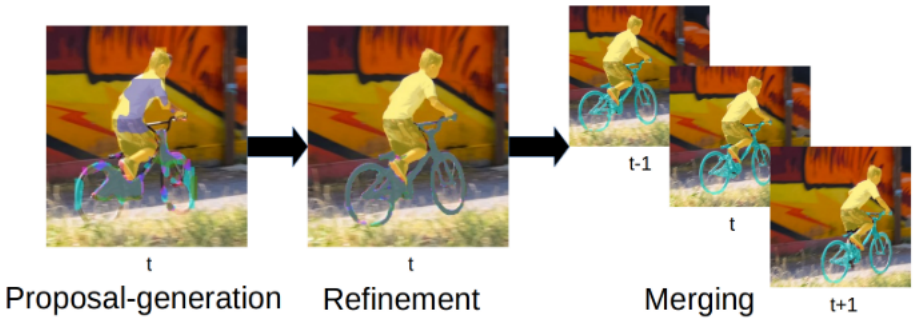


Figura 3.3: Etapas da aplicação do modelo PReMVOS. Extraído de [4]

Como já falamos superficialmente a maneira que o modelo PReMVOS funciona podemos nos dedicar nos próximos parágrafos a nos aprofundar em cada etapa desse modelo. A Figura 3.3 representa a arquitetura do modelo.

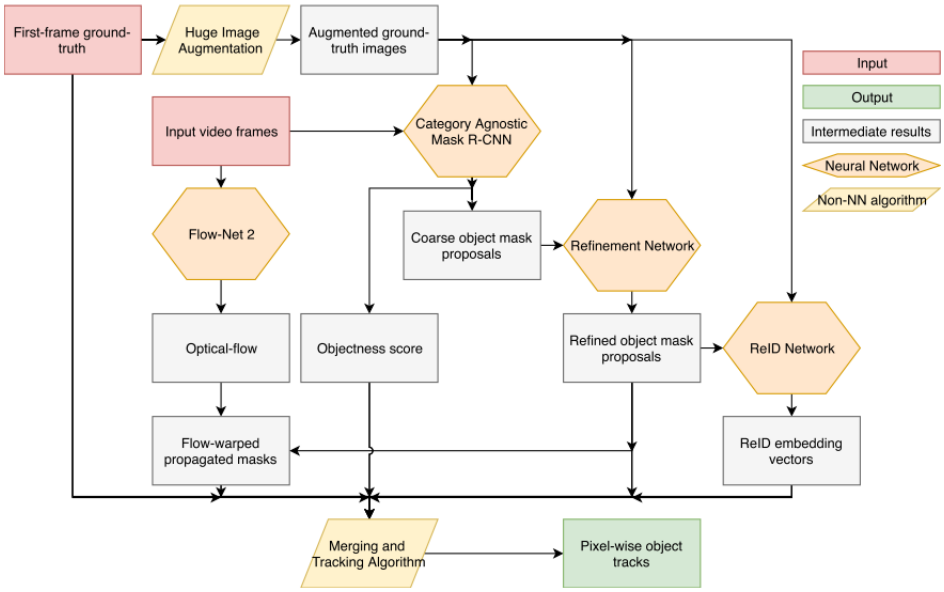


Figura 3.4: Arquitetura do modelo PReMVOS. Extraído de [4]

Na Figura 3.4, a etapa de *Huge image Augmentation* representa a utilização do método Lucid Data Dreaming, proposto em [13], para aumentar a base anotada de maneira artificial. Como esse método são geradas 2500 imagens com base no primeiro *frame*.

A geração de propostas fica a cargo de uma implementação de uma rede neural do tipo Mask R-CNN, mostrada em [22], que utiliza como *backbone* uma arquitetura do tipo ResNet101 [23]. Inicialmente essa rede neural é

treinada apenas para separar o primeiro plano da imagem, selecionando objetos genéricos. Por fim é criar uma rede neural, resultado do processo de *fine-tuning* para os dados de uma classe. O resultado desta etapa é uma máscara grosseira, uma *bounding-box* e o *Objectness score* que identifica qual objeto do *frame* de referência cada marcara se refere.

O modelo *Refinement Network* é baseado em uma FCN, com arquitetura baseada no modelo DeepLabV3+ [24]. Essa rede neural recebe como entrada imagens de 385×385 que são as região propostas cortadas e redimensionada. Ao cortar o *bounding-box* resultante da etapa de proposta, são adicionados 50 pixels em todas as direções para garantir que nenhuma região do objeto se perca. Além disso a *bounding-box* é adicionada como um quarto canal a imagem como se fosse uma máscara. Essa rede é pré-treinada para retornar uma segmentação acurada a partir de uma *bounding-box*. Para capturar as características do objeto, é criada uma versão dessa rede resultado do processo de *fine-tuning* utilizando as 2500 imagens geradas a partir do *frame* anotado. Por fim, essa rede pode ser utilizada para gerar a máscara refinada utilizando apenas a *bounding-box* gerada na etapa anterior.

Na seção , ao falar sobre modelos baseados em movimento, falamos como esse tipo de problema pode ser encarado como uma propagação de máscara ao longo dos *frames* de um vídeo. Nesse caso, a etapa de propagação de máscara utiliza o *Mask Propagation score*. O valor do *Mask Propagation score* é calculado a partir do fluxo-óptico entre *frames* consecutivos e suas máscaras refinadas.

Outra etapa que utiliza as máscara refinadas geradas pela *Refinement Network* é a extração de *ReID embedding vectors*. Esse modelo é baseado em uma variante da rede neural ResNet [25] pré-treinado utilizando o *dataset* ImageNet [11] e treinado no dataset COCO [26]. O objetivo desse modelo é transformar as imagens em vetores de um espaço vetorial cuja imagens de mesma classe fiquem próximas enquanto de classes diferentes são mais distantes. Em seguida esse modelo passa pelo processo de *fine-tuning* utilizando as 2500 imagens geradas na etapa *Huge image Augmentation* para cada um dos vídeos, para ser capaz de diferenciar os 242 objetos do dataset DAVIS [7]. Assim o resultado desse modelo pode ser utilizado para diferenciar os objetos segmentados entre si, e compara-los com o objeto do *frame* de referência. A Figura 3.5 mostra as transformações que são feitas na imagem antes de passar pelo modelo de *embedding*.

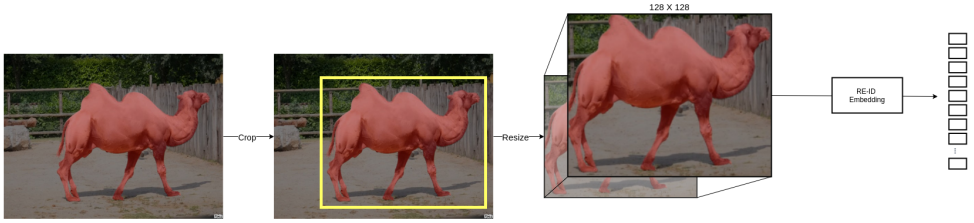


Figura 3.5: Pré-processamento para geração do vetor *ReID embedding*

Por fim, é aplicado o algoritmo de *merging* e *tracking*. Para estimar de qual objeto uma certa máscara refinada é, essa etapa utiliza o *Objectness score*, o *Mask Propagation score* e o *ReID score* que é gerado a partir da distância Euclideana entre os vetores gerados pelo *ReID embedding*.

Esse algoritmo aplica uma abordagem gulosa, onde calcula ente *frames* consecutivos a partir do primeiro *frame*, os *scores* mencionados no acima. Por fim esses *scores* são combinados e então cada máscara pode ser classificada como uma das classes presentes no vídeo.

O modelo PReMVOS é mais complexo do que o modelo OSVOS, mostrado na seção 3.1, já que ele se propões a realizar a segmentação multi-classe. Por isso existem passos dedicados a manter a coerência temporal entre diferentes classes do vídeo.

3.3 VINet

As seções anteriores deste capítulo se dedicaram a falar sobre métodos de VOS. Nessa seção iremos falar sobre o modelo escolhido para realizar a etapa de Video Inpainting, o modelo VINet descrito em [6].

Apesar da evolução na solução do problema de *Image Inpainting*, utilizando redes neurais profundas, ainda há muito espaço para evoluir quando entramos no domínio de vídeos. Uma maneira mais direta de solucionar o problemas de VI é aplicar métodos de *Image Inpainting* em cada *frame* do vídeo, entretanto não é garantido uma coerência temporal. Em seu trabalho Kim *et al.* discorrem sobre um modelo que tem duas funções principais: agregação de atributos temporais, ou seja utilizar informações de outros *frames* para reconstruir a região selecionada; e consistência temporal.

O problema de VI visa reconstruir regiões em *frames* de um vídeo $X_1^T = X_1, X_2, \ldots, X_T$. As regiões reconstruídas devem ser muito similares a região verdadeira, como nos *frames* completos $Y_1^T = Y_1, Y_2, \ldots, Y_T$. O problema de VI pode ser formalizado como a busca de uma função que mapeie X_1^T

para $\hat{Y}_1^T = \hat{Y}_1, \hat{Y}_2, \ldots, \hat{Y}_T$ de maneira que a distribuição $p(\hat{Y}_1^T | X_1^T)$ seja igual a $p(Y_1^T | X_1^T)$.

Para simplificar o problema, a probabilidade pode ser fatorizada, da seguinte forma:

$$p(Y_1^T | X_1^T) = \prod p(\hat{Y}_t | X_t) \tag{3-1}$$

A Equação 3-1 é uma interpretação ingênua do problema e pode ser interpretada como aplicação de *Image Inpainting frame* a *frame*. Entretanto para gerar um resultado mais convincente, o trabalho [6] propõe a inclusão das seguintes variáveis na geração de Y_t:

- Alguns *frames* na vizinhança do t-ésimo *frame*. Assim passamos a incluir o subconjunto de *frames* X_{t-N}^{t+N}, sendo N a variável de raio da vizinhança.
- O resultado da reconstrução do *frame* anterior, $\hat{Y}_{t-1}$.
- Um fator de memória recorrente M_t.

Sendo assim, a distribuição modelada em [6] passa a ser

$$p(Y_1^T | X_1^T) = \prod p(\hat{Y}_t | X_{t-N}^{t+N}, \hat{Y}_{t-1}, M_t) \tag{3-2}$$

O modelo VINet é baseado em uma rede neural profunda que aprende uma função $\hat{Y}_t = D(X_{t-N}^{t+N}, \hat{Y}_{t-1}, M_t)$, que busca respeitar a distribuição da Equação 3-2. A Figura 3.6 é, de uma maneira simplificada, uma representação de como o modelo está estruturado.

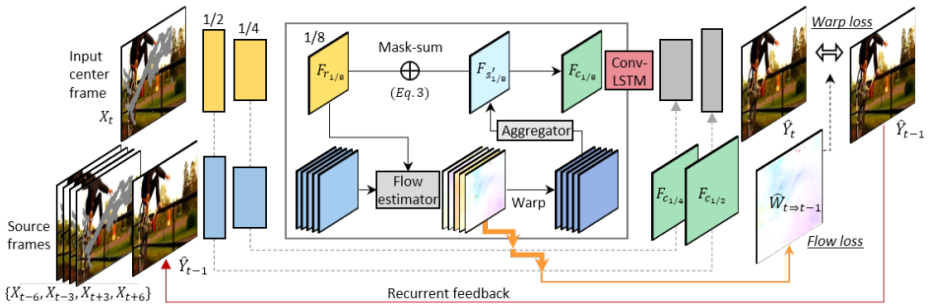


Figura 3.6: Arquitetura do modelo VINet. Extraido de [6]

A Figura 3.6 mostra que o modelo recebe múltiplos *frames* como entrada. O *frame* que deve ser reconstruído X_t é chamado de *Input center frame*. Outros *frames* que fornecem informação para reconstrução são chamados *Source frames*. Os *Source frames* são $\hat{Y}_{t-1}$ e $X_{t-N}^{t+N} = \{X_{t-6}, X_{t-3}, X_{t+3}, X_{t+6}\}$ (O modelo proposto tem raio $N = 2$ e um fator de deslocamento stride $= 3$).

Como dito em [6], a arquitetura é inspirada em um rede neural do tipo *encode-decode*. Em [10] o processo de *encode* é definido como a transformação de uma entrada de tamanho variado para uma saída que representa um estado de tamanho fixo, também chamado de *feature map*. Por sua vez, a etapa de *decode* deve a partir do *feature map* reconstruir uma saída que pode ser de tamanho variado. A partir disso, a arquitetura *encode-decode* busca aprender a representação do dado como um *feature map* e em seguida reconstruir o dado original a partir deste *feature map*.

Para alimentar o modelo as máscaras que representam as áreas que devem ser reconstruídas são concatenadas nas imagens na dimensão dos canais da imagem. Em seguida as 6 entradas (*i.e.* 5 entradas do *Source frames* mais o *Input center frame*) , passam por *encoders*. Os *Source frames* passam por um *encoder* que compartilha os mesmo pesos, enquanto o *Input center frame* passa por outro.

Os *encoders* nesse modelo retornam 4 escalas de *feature maps*: 1 (256×256), 1/2 (128×128), 1/4 (64×64) e 1/8 (32×32). Em seguida é estimado o fluxo óptico entre os *feature maps* do *Input center frame* e dos *Source frames* em cada uma dessas escalas.

Os fluxos estimados serve como guia para distorcer os *feature maps* . Essa distorção das posição dos valores do *feature maps* é também chamada de *Warp*.

O *Agregator* mostrado na Figura 3.6 representa a concatenação dos *Source frames* ao longo de um eixo dos resultados após o processo de *Warp* citado anteriormente. Esse resultado é passado por camadas convolucionais que produz finalmente $F_{s'}$, que é a representação dos atributos espaço-temporais.

Por outro lado F_r é apenas o resultado do *encoder* do *Input center frame*. Os valores de $F_{s'}$ e F_r são utilizados na geração de uma máscara de oclusão que deve ser usada para calcular um novo *feature maps* combinando $F_{s'}$ e F_r. Para obter essa máscara, uma rede neural recebe como entrada $|F_{s'} - F_r|$ e retorna a máscara m.

A combinação de $F_{s'}$ e F_r é feita utilizando a máscara m, obtida na etapa anterior. O resultado F_c é definido pela seguinte equação:

$$F_c = (1 - m) \odot F_r + m \odot F_{s'} \tag{3-3}$$

Como falamos anteriormente, o *encoder* geram *feature maps* em 4 escalas. Sendo assim, $F_{s'}$, F_r, m e F_c também são gerados para 4 escalas. Por fim, esses *feature maps* alimentam o *decoder* que devem gerar uma proposta de $\hat{Y}_t$, que ainda não está refinada e iremos chamar de $\hat{Y}'_t$.

O fluxo óptico $\hat{W}_{t \Rightarrow t-1}$ é a estimativa do fluxo entre os *feature maps* de

X_t e $\hat{Y}_{t-1}$ na escala mais refinada (*e.g.* 256×256). Seja então $\hat{W}_{t \Rightarrow t-1}(\hat{Y}_{t-1})$ a representação do processo de *Warp* aplicado à $\hat{Y}_{t-1}$ utilizando o fluxo $\hat{W}_{t \Rightarrow t-1}$. Sendo assim, o modelo propõe calcular $\hat{Y}_t$ como

$$\hat{Y}_t = (1 - m) \odot \hat{Y}'_t + m \odot \hat{W}_{t \Rightarrow t-1}(\hat{Y}_{t-1}) \tag{3-4}$$

Além de usar o ultimo resultado $\hat{Y}_{t-1}$ na estimativa de $\hat{Y}_t$, uma camada ConvLSTM é aplicada ao *feature map* de menor escala (*e.g.* 32×32). Essa rede neural utiliza um parâmetro de estado que é repassado entre execuções de *frames* e funciona como um parâmetro de memória dos *frames* passados. Essa parte da arquitetura do modelo representa o parâmetro M_t na Equação 3-2.

O modelo VINet foi treinado em duas etapas: primeiro é treinado sem considerar a utilização de $\hat{Y}_{t-1}$ e M_t, com objetivo de aprender a agregação dos atributos temporais que podem ser extraídos de X_{t-N}^{t+N}; A segunda etapa inclui a ConvLSTM e o $\hat{Y}_{t-1}$ para refinar os resultados com o fator de memória e o resultado do modelo do *frame* anterior.

3.4 Remoção de objetos em vídeos

Nas secções anteriores nos aprofundamos nas técnicas escolhidas para desenvolvimento deste trabalho. Agora iremos tratar de uma pesquisa com o mesmo objetivo, mas que utiliza técnicas diferentes para as etapas de VOS e VI.

Em seu trabalho [12], Le *et al.* desenvolvem uma técnica de remoção de objetos em vídeos. Neste trabalho a modelagem do problema é similar a que utilizamos, uma vez que o problema é quebrado em dois problemas menores: VOS e VI. Entretanto, este trabalho permite a intervenção do usuário para ajustar os possíveis erros da etapa de VOS. Além disso ele implementa diferentes abordagens de VI de acordo com a qualidade do resultado esperado pelo usuário.

3.4.1 Video Object Segmentation

Na etapa de VOS o trabalho [12] descreve uma sequência de modelos aplicados para realizar essa tarefa. A Figura 3.7 mostra com mais detalhes as etapas aplicadas para realizar a segmentação.

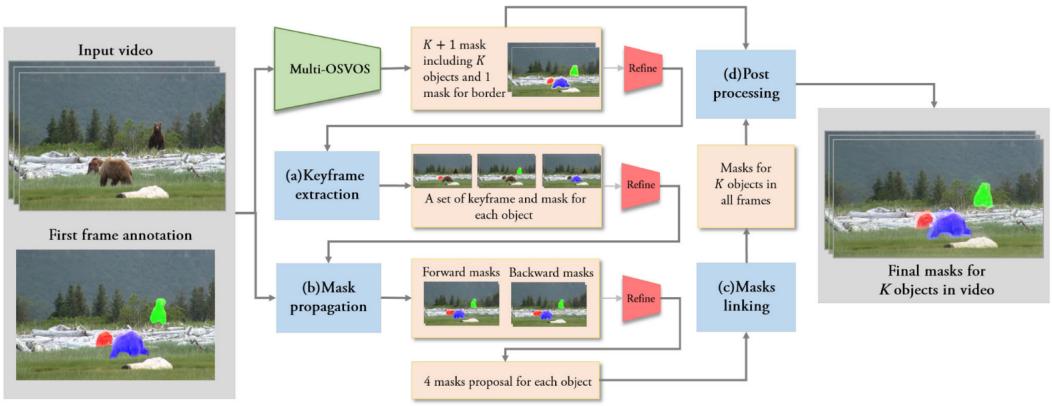


Figura 3.7: Etapas para a segmentação de objetos proposto por Le *et al.* em [12]. Imagem extraída de [12].

Neste são aplicados modelos baseados em detecção e modelos baseados em movimento para solucionar o problema de VOS. A Figura 3.7 mostra em verde o modelo baseado em detecção e em azul os modelos baseados em movimento. Além disso a imagem mostra em vermelho uma etapa de refinamento de marcaras que é aplicada após algumas etapas. Ao longo dos próximos parágrafos iremos nos dedicar a falar sobre cada uma dessas etapas e como elas contribuem para o resultado final desde *pipeline*.

O modelo **Multi-OSVOS**, em verde na Figura 3.7, é baseada no modelo OSVOS [5] mas com algumas diferenças importantes:

- O modelo Multi-OSVOS permite a detecção de diferentes objetos, não só a separação de primeiro e segundo plano como é feito pelo modelo OSVOS.
- Diferente do modelo OSVOS, o modelo Multi-OSVOS não se baseia em uma *Fully Convolutional Network*(FCN) mas sim na arquitetura do modelo Deeplab v2 [16].
- Na etapa de *Fine-tuning* o modelo Multi-OSVOS aplica o modelo Lucid Data Dreaming [13] para aumentar a base anotada de maneira artificial. Similar ao que é feito no modelo PReMVOS falado na seção 3.2.

O modelo Multi-OSVOS pode separar objetos do segundo plano de um vídeo, entretanto ele se baseia apenas na aparência do objeto, sem considerar atributos de movimento do vídeo. Por isso em vídeos com objetos semelhantes o modelo pode segmentar objetos errados. Para evitar isso Le *et al.* propõe incorporar outros passos na solução do problema de VOS.

Em busca de máscaras mais precisas é adicionado um modelo responsável pelo refinamento das máscaras, mostrado na Figura 3.7 como *Refine*. Esse modelo tem a mesma arquitetura que o modelo Multi-OSVOS, mas recebe além do *frame* uma proposta rustica de segmentação.

Os *Keyframes* de um certo objeto são *frames* cuja a máscara para esse objeto é considerada precisa. Além das máscaras dadas como referência, esse modelo propõe uma heurística de extração de *Keyframes*. O processo de propagação de máscara ocorrerá apenas entre *Keyframes* consecutivos.

O modelo Multi-OSVOS propõe algumas máscaras para um certo objeto. Se essas máscaras propostas tem uma taxa de intersecção maior que 80% com a maior das máscaras propostas e estão todas isoladas de outras máscaras de outros objetos, consideramos esse um *Keyframe* para o objeto i.

As máscaras são propagadas pra frente e para trás dos *Keyframes*. O algoritmo de propagação de máscara se baseia na similaridade de regiões de pixels entre *frames* e o quanto as regiões mais similares tem de intersecção com a máscara do *frame* anterior.

Como a propagação é feita em dois sentidos entre *Keyframes*, cada *frame* passa a ter duas máscaras: uma máscara M^{fw} resultante da propagação para frente, seguindo a sequência $t, t+1, \ldots, t+n$; outra máscara M^{bw} resultante da propagação para trás, seguindo a sequência $t+n, t+n-1, \ldots, t$. Além dessas máscaras, na proposta também são incluídas $M^{fw} \cap M^{bw}$ e $M^{fw} \cup M^{bw}$.

Como foi falado anteriormente, após a propagação das máscaras cada objeto tem 4 propostas de máscara. Para decidir a melhor máscara para o objeto é aplicada uma técnica baseada em grafo, similar a descrita na seção 2.1.5.1. Essa técnica também auxilia na correção de associações erradas entre objetos e máscaras, uma vez que considera a similaridade global das máscara ao longo de todo o vídeo.

A etapa anterior resulta em apenas uma máscara por objeto por *frame*. Em seguida uma etapa de pós-processamento é incluída para garantir que as máscaras cubram completamente os objetos. Nessa etapa um processo de dilatação das máscaras resultantes é aplicado, uma vez que o objetivo é a remoção do objeto a perda de precisão do método de VOS ao fazer essa dilatação não deve impactar o resultado final.

3.4.2 Video Inpainting

Na etapa responsável pela aplicação da técnica de VI o trabalho [12] mostra a aplicação de duas possíveis abordagens: Uma para vídeos com segundo planos estáticos; outra para vídeo com segundo plano dinâmicos.

3.4.2.1
Segundo plano estático

A técnica de VI desenvolvida nesse modelo assume que o fundo do vídeo é estático, ou apenas com pequenas instabilidades, e que o fundo é completamente visível ao longo dos *frames*.

Assumindo o que foi descrito acima, a abordagem proposta é copiar o fundo para a parte do vídeo que deve ser reconstruída. Esse processo de cópia é baseado no fluxo óptico do vídeo para encontrar o pixel que deve ser copiado para certa região.

Inicialmente, o vídeo é estabilizado, em seguida o fluxo óptico da parte que não deve ser reconstruída é calculado. Para calcular o fluxo óptico da parte que deve ser reconstruída, é aplicado um algoritmo de *Image Inpainting* ao fluxo óptico já calculado.

Assim, com o fluxo óptico de todos os *frames* é possível buscar o pixel que deve ser copiado apara preencher a região a ser reconstruída. Em seguida uma técnica de *Poisson blending*, como mostrada em [28], é aplicada para corrigir mudanças de luminosidade e intensidade do pixel copiado.

3.4.2.2
Segundo plano dinâmico

Quando o segundo plano é dinâmico, o método anterior mostrado na Seção 3.4.2.1 falha ao capturar as variações no segundo plano. Por exemplo, caso haja outros objetos se movendo no segundo plano, o método pode falhar ao tentar reconstruir uma região que cause uma oclusão nesses objetos.

O método aplicado nesse caso é desenvolvido por Le *et al.* em [29]. Esse trabalho descreve um algoritmo de VI que trata as imagens a partir de regiões de pixels, conhecidas como *patches*.

Este trabalho aborda o problema de VI como a busca por mapas de *patches* que mapeie entre *frames* os *patches* de regiões que não devem ser reconstruídas para *patches* de regiões que devem ser reconstruídas. Para garantir a coerência temporal esse mapeamento utiliza o fluxo óptico.

4
Metodologia Proposta

Neste Capítulo iremos expor o modelo proposto e como foi feita sua implementação e avaliação.

4.1
Modelos Propostos

Como mostrado na Seção 3.4 uma abordagem para solucionar o problema de remoção de objetos em vídeo utilizando poucas anotações pode ser dividido em duas etapas: VOS e VI.

Nossa abordagem será a de dividir o problema em VOS e VI. O diferencial são os modelos utilizados para solucionar cada etapa. Para o problema de VOS utilizamos o modelo PReMVOS e OSVOS, enquanto para o problema de VI utilizamos o modelo VINet. A Figura 4.1 mostra os *pipelines* implementados de maneira simplificada.

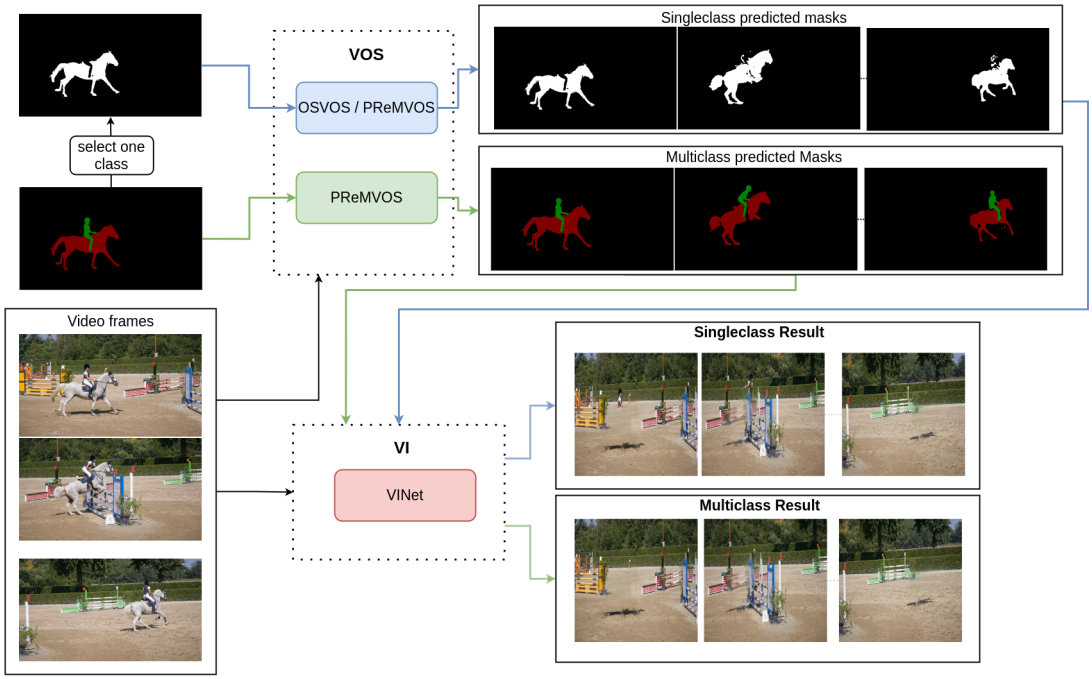


Figura 4.1: Arquitetura simplificada de modelo proposto

Como mostrado na Figura 4.1 foram implementados diferentes *pipelines*, um mostrado em verde e outro em azul. O *pipeline* em azul mostra o processo de remoção de apenas um objeto do vídeo, enquanto o mostrado em verde mostra a remoção de vários objetos.

Uma vez que o *dataset* escolhido possui vídeos com mais de uma classe, um processo de preparação do dado de entrada foi criado para alimentar o *pipeline* de remoção de apenas um objeto. Esse processo extrai a máscara do objeto que ocupa a maior área dos vídeos.

Para a remoção de apenas um objeto do vídeo podem ser utilizados um dos dois modelo implementados para a solução da tarefa de VOS: OSVOS ou PReMVOS. A Figura 4.1 mostra o resultado do modelo OSVOS como *Singleclass predicted masks*.

Enquanto isso, Para a remoção de mais de um objeto, apenas o modelo PReMVOS foi aplicado. O resultado desse modelo é mostrado como *Multiclass predicted masks*.

Ao contrário do proposto em [12], optamos por ter apenas um modelo para solucionar a etapa de VI. O trabalho de Le *et al.* [12] implementa dois modelos para lidar com tipos de vídeos diferentes: um para lidar com vídeos de fundo estático, e outro para vídeos com fundo dinâmico. Para diminuir a complexidade do processo final então escolhemos a aplicação de apenas um modelo mais robusto e que lide com ambos os tipos de vídeos.

Para solucionar a tarefa de VI escolhemos o modelo VINet por ter apresentado bons resultados mediante percepção humana. Nessa avaliação, mostrada na Figura 4.2, o modelo VINet na maioria dos vídeos do *dataset* escolhido gerou resultados mais convincentes quando avaliado por pessoas do que outros modelos.

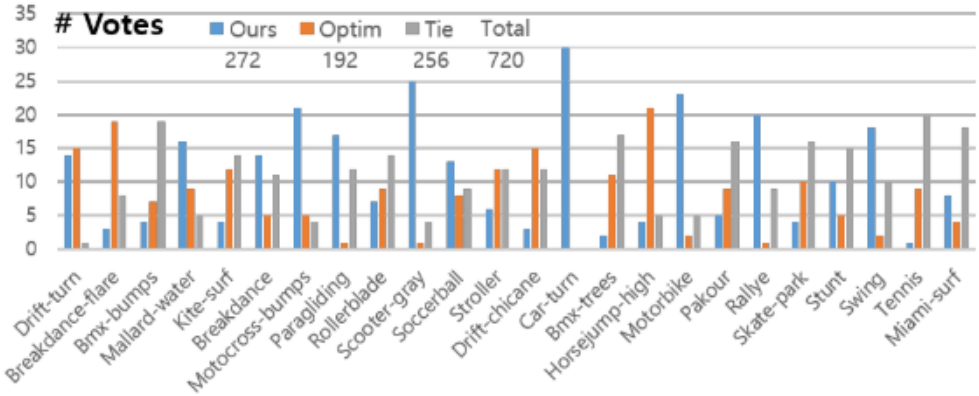


Figura 4.2: Avaliação de modelos de VI por usuários. Extraído de [6]

A Figura 4.2 mostra a comparação do modelo VINet com outro forte candidato [35]. Segundo o gráfico representado na figura, quando expostos aos resultados dos modelos, na maioria dos vídeos a maioria dos usuários julga o resultado do modelo VINet como melhor.

4.2 Método de avaliação

Para avaliação do modelo proposto, extraímos métricas tanto dos resultados gerados pela etapa de VOS quanto pelos resultados gerados pela etapa de VI.

Os resultados da etapa de VOS foram avaliados com as três métricas principais para avaliações desses métodos, como mostrado em [2]: *precision*, *recall* e IoU(*Intersection Over Union*).

Para avaliação dos modelos de VI, como mostrado em [6], as métricas são mais subjetivas. Como desejamos avaliar o impacto que o modelo de VOS proposto tem sobre o resultado final, propomos uma avaliação com a métrica de SSIM (*Structural SIMilarity*).

O objetivo da métrica SSIM, como descrito por Wang *et al.* em [30], é medir a similaridade entre imagens de maneira estrutural, similar a percepção humana. Para isso a métrica não é muito influenciada por por ruídos, mudanças no contraste ou diminuição de resolução da imagem, ao contrário da medida de MSE (*Mean Square Error*) por exemplo.

A métrica SSIM é aplicada por áreas nas imagens. isso faz com que ele capture informações estruturais. Essa métrica costuma ser sumarizada normalmente como MSSIM (*Mean* SSIM), ou seja, a média dos valores de SSIM por região.

Por ser uma métrica que se assemelha a avaliação humana de similaridade de imagens, achamos que essa é uma métrica razoável para ser aplicada nessa avaliação. Utilizamos a métrica de SSIM para medir a similaridade entre as reconstruções pseudo-verdadeiras (*i.e.* reconstruções com base nas máscaras das anotações verdadeiras) e as reconstruções preditas(*i.e.* reconstruções com base nas máscaras preditas pelos modelos de VOS).

Além do valor de MSSIM, usamos o mapa dos valores de SSIM como avaliação das regiões com maior diferença. Isso nos possibilita avaliar o quanto as falhas dos modelos VOS impactam o resultado da imagem reconstruída.

5
Resultados

Neste Capítulo iremos nos dedicar a exibir os resultados. Iremos nos aprofundar nos resultados dos métodos de VOS para segmentações *Singleclass* e *Multiclass*. Além disso vamos comparar os resultados do modelo aplicado na etapa de VI quando alimentado com dados preditos e dados reais.

Para performar os testes utilizamos o *dataset* DAVIS [7]. Nesse *dataset* encontramos vídeos e anotações de máscaras que segmentam um ou mais objetos em 90 vídeos. Esse *dataset* tem sido utilizado como referência para validação de tarefas de VOS e VI, como é visto nos trabalho [6], [12], [5] e [4], que utilizamos como referência nesse trabalho.

5.1
Resultados da etapa de VOS

Nessa seção vamos explorar os resultados obtidos da etapa de VOS. Como estamos performando essa tarefa para aplicação de um modelo de VI logo em seguida, vamos focar na avaliação da métrica de *recall*.

O foco na avaliação do *recall*, assim como falado em [12], é para garantir que nenhuma parte do objeto a ser removido deixe de ser incluída. Entretanto é importante não deixar de observar outras métricas, já que incluir regiões para serem reconstruídas além do objeto pode penalizar o resultado final.

5.1.1
Singleclass VOS

A Figura 5.1 mostra as métricas de *recall*, *precision* e *IoU* para os 15 vídeos do *dataset* com maior *recall* médio do resultado do modelo OSVOS. A linha vertical no topo das barras mostra toda a variação de que a métrica teve nos *frames* do vídeo.

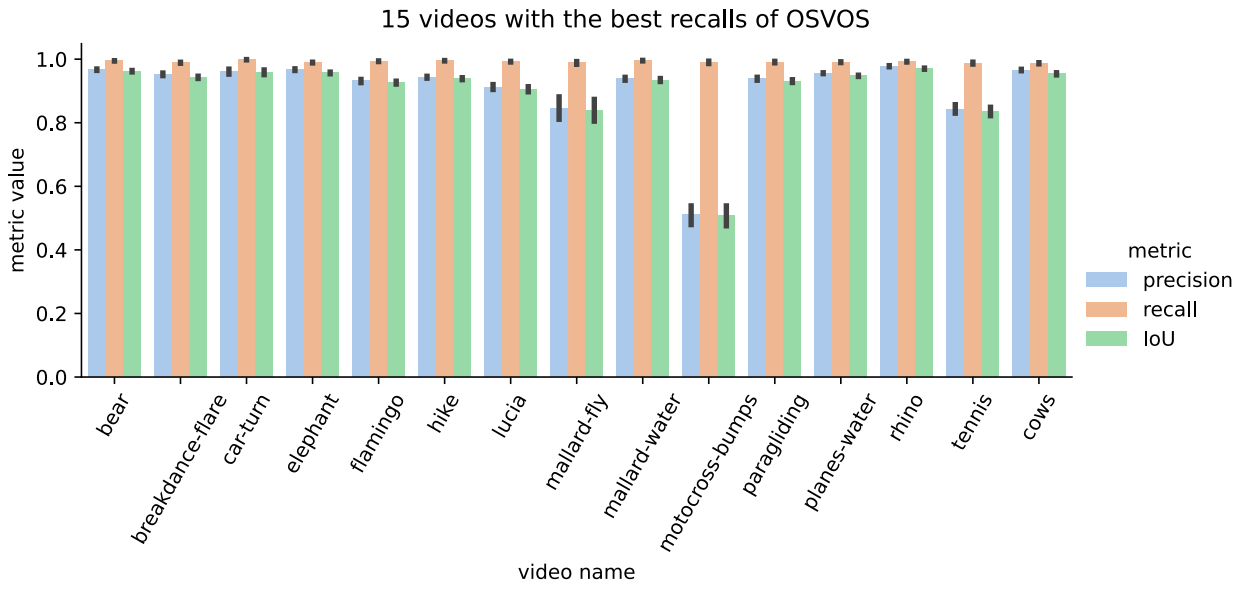


Figura 5.1: Métricas de *precision*, *recall* e IoU dos 15 vídeos com maiores *recall* médio do modelo OSVOS.

Observando os resultados apresentados na Figura 5.1 vemos que apenas um vídeo, apesar de manter um alto valor de *recall*, tem valores de *precision* e IoU mais baixos. Isso acontece no vídeo `motocross-bumps`.

Na Figura 5.2 mostramos o resultado da aplicação do modelo OSVOS no vídeo `motocross-bumps` e podemos observar que o modelo falha ao aprender as características da moto e acaba por englobar o motoqueiro como o objeto a ser segmentado. Isso acontece uma vez que foi dado para o modelo apenas o primeiro *frame* como referência, onde o objeto que deve ser segmentado esta em sua maior parte ocluso.

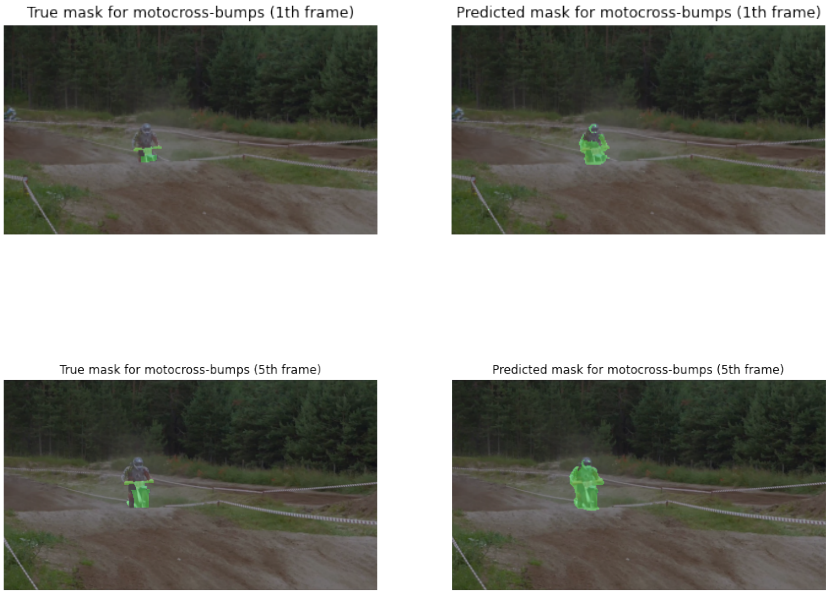


Figura 5.2: Comparação entre máscaras preditas e verdadeiras do modelo OSVOS no vídeo `motocross-bumps`.

Abaixo a Figura 5.3 mostra as mesmas métricas para os 15 vídeos do *dataset* que tiveram menor *recall* médio do resultado do modelo OSVOS.

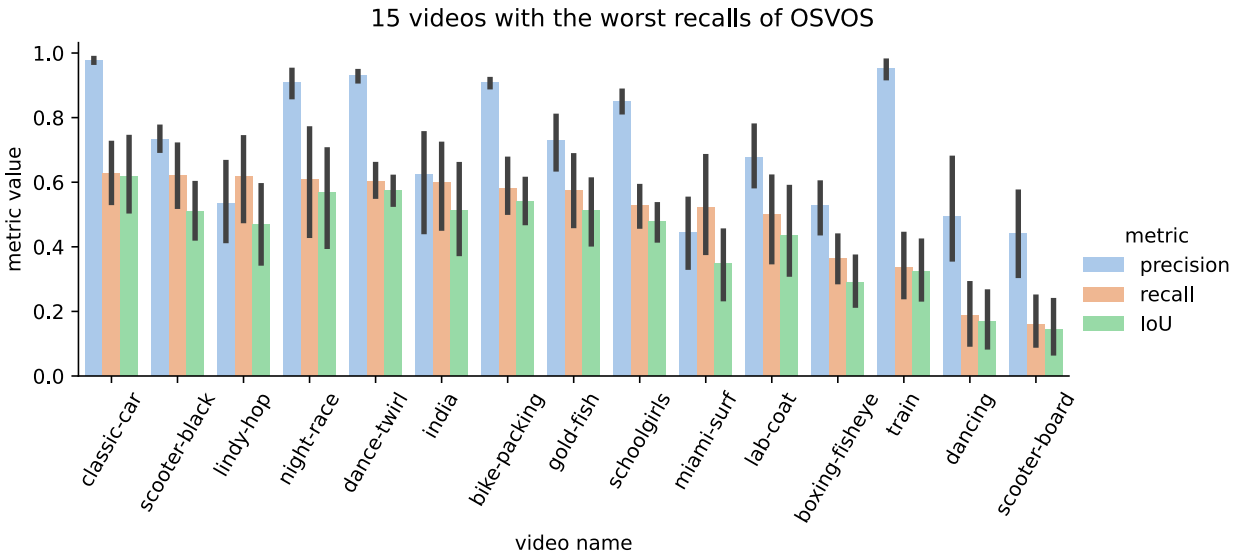


Figura 5.3: Métricas de *precision*, *recall* e *IoU* dos 15 vídeos com menores *recall* médio do modelo OSVOS.

Observando o resultado mostrado no gráfico da Figura 5.3 encontramos casos em que, apesar do baixo *recall*, o valor de *precision* é bem alto. Um desses vídeos é o `train`. Observando a Figura 5.4 podemos ver como o modelo não consegue segmentar o ultimo vagão por completo, segmentando apenas uma região dele, o que explica o alto valor de *precision* mas o baixo valor de *recall* e IoU.

Figura 5.4: Comparação entre máscaras preditas e verdadeiras do modelo OSVOS no vídeo `train`.

A Figura 5.5 mostra as métricas de *recall*, *precision* e IoU para os 15 vídeos do *dataset* com maior *recall* médio do resultado do modelo PReMVOS.

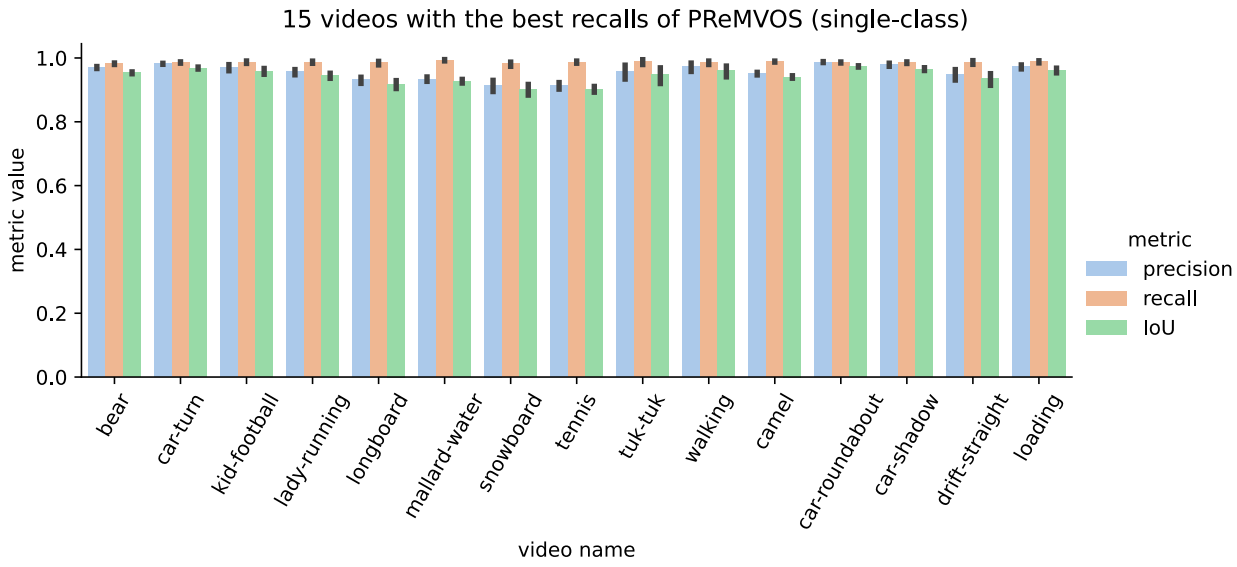


Figura 5.5: Métricas de *precision*, *recall* e *IoU* dos 15 vídeos com maiores *recall* médio do modelo PReMVOS.

Em seguida temos também a Figura 5.6 que mostra as mesmas métricas para os 15 vídeos do *dataset* que tiveram menor *recall* médio do resultado do modelo PReMVOS.

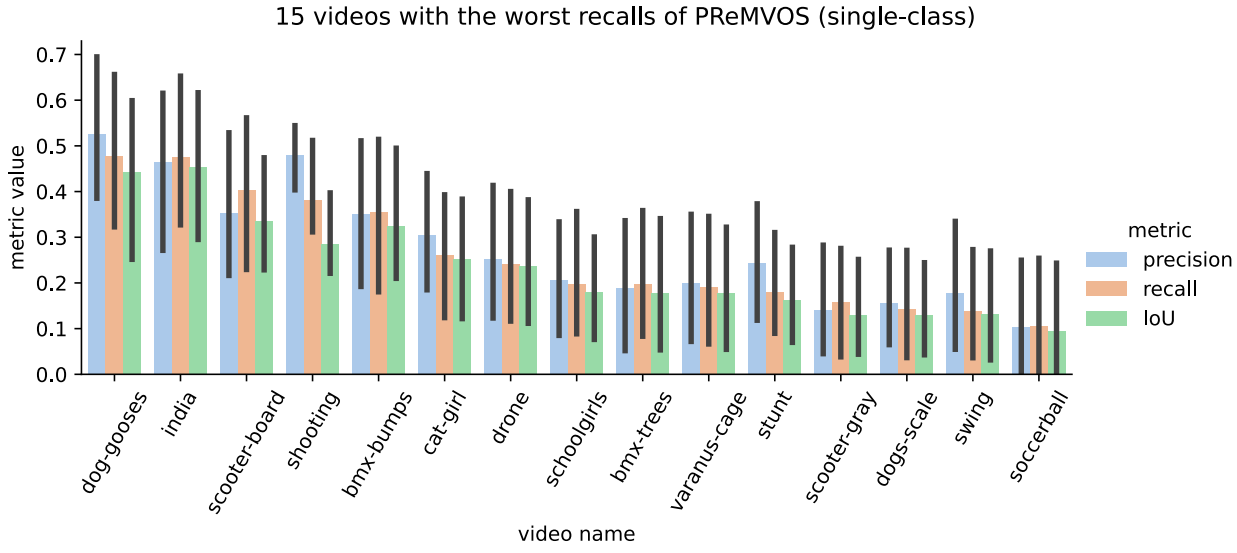


Figura 5.6: Métricas de *precision*, *recall* e *IoU* dos 15 vídeos com menores *recall* médio do modelo PReMVOS

Diferente dos resultados do modelo OSVOS, não temos casos de grande diferença entre os valores de *precision* e *recall*. Entretanto, percebemos a piora

dos resultados deste modelo em vídeos onde o objeto passa por alguma oclusão. A Figura 5.7 mostra um exemplo em que o objeto passa por oclusão no vídeo `bmx-trees`.

Figura 5.7: Comparação entre máscaras preditas e verdadeiras do modelo PReMVOS no vídeo `bmx-trees`.

A Figura 5.8 representa a comparação dos valores de *recall* do conjunto de vídeos que aparecem como maiores valores em algum dos modelos.

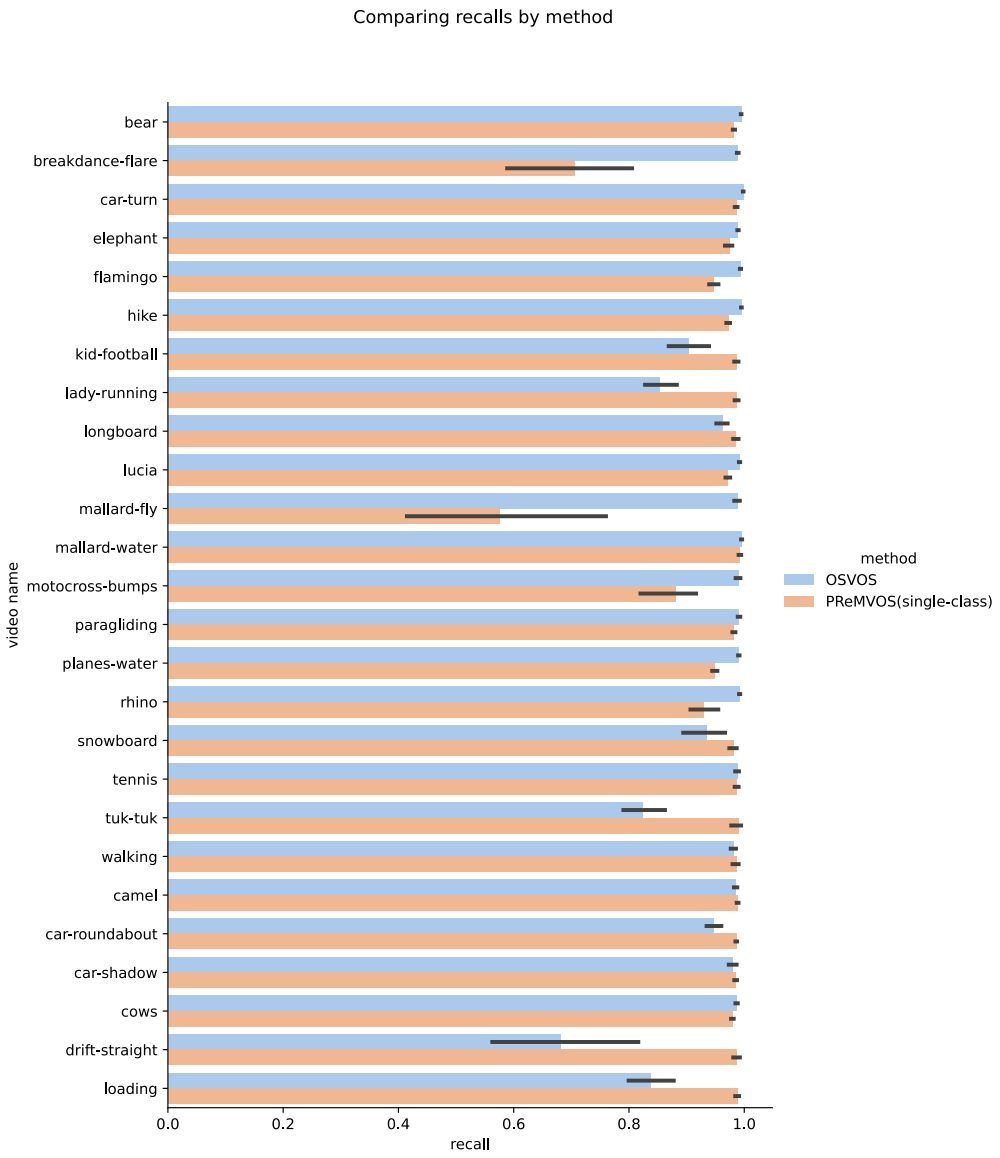


Figura 5.8: Maiores valores de *recall* por modelo

Por outro lado, a Figura 5.9 representa a comparação dos valores de *recall* do conjunto de vídeos que aparecem como menores valores em algum dos modelos.

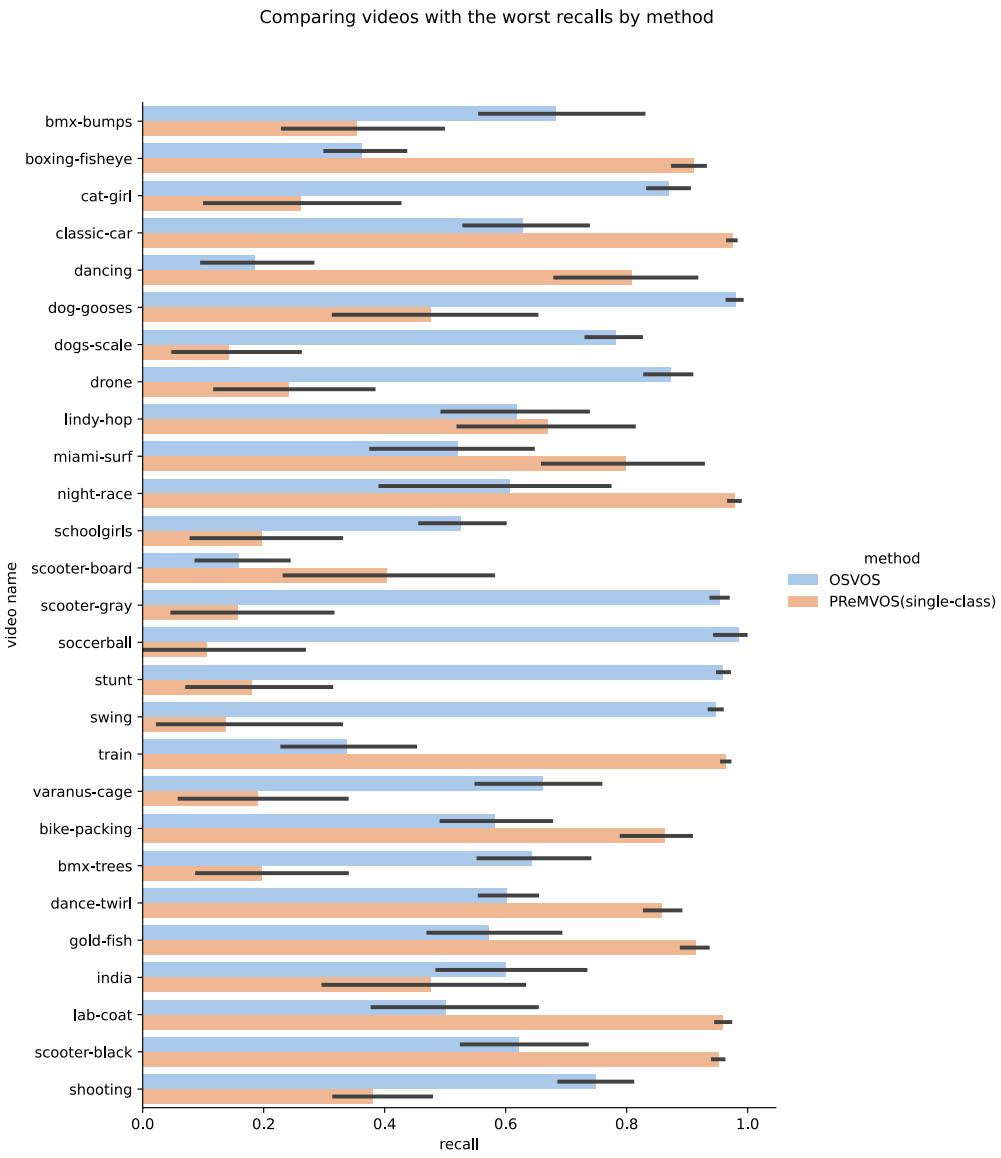


Figura 5.9: Menores valores de *recall* por modelo

É interessante observar na Figura 5.9 como o valor de *recall* pode variar entre os modelos. No vídeo `boxing-fixeye`, por exemplo, o valor de *recall* do resultado do modelo PReMVOS é consideravelmente maior que o medido no resultado do modelo OSVOS. A Figura 5.10 mostra o resultado dos modelos para o vídeo. O gráfico mostrado na Figura 5.11 mostra que com a evolução do vídeo é notável como o modelo OSVOS perde a precisão e o *recall* enquanto o modelo PReMVOS se mantém.

5.10(a): Resultado do modelo OSVOS para o vídeo `boxing-fisheye`

5.10(b): Resultado do modelo PReMVOS para o vídeo `boxing-fisheye`

Figura 5.10: Comparação de resultados do modelo PReMVOS e OSVOS no vídeo `boxing-fisheye`

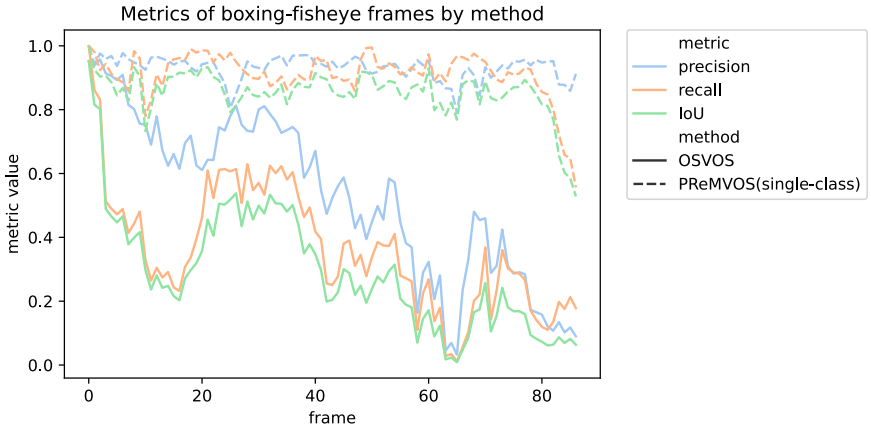


Figura 5.11: Métricas do vídeo `boxing-fisheye` por modelo

Por outro lado, é possível ver exemplos onde o modelo OSVOS superou o modelo PReMVOS. No vídeo `swing` é notável a diferença entre o *recall* médio entre os resultado dos modelos. A Figura 5.12 mostra os resultados

dos modelos OSVOS e PReMVOS no vídeo `swing`. Em seguida a Figura 5.13 mostra as métricas ao longo dos *frame* por método, onde é possível comparar a queda brusca dos valores de *precision*, *recall* e IoU no resultado do modelo PReMVOS.

5.12(a): Resultado do modelo OSVOS para o vídeo `swing`

5.12(b): Resultado do modelo PReMVOS para o vídeo `swing`

Figura 5.12: Comparação de resultados do modelo PReMVOS e OSVOS no vídeo `swing`

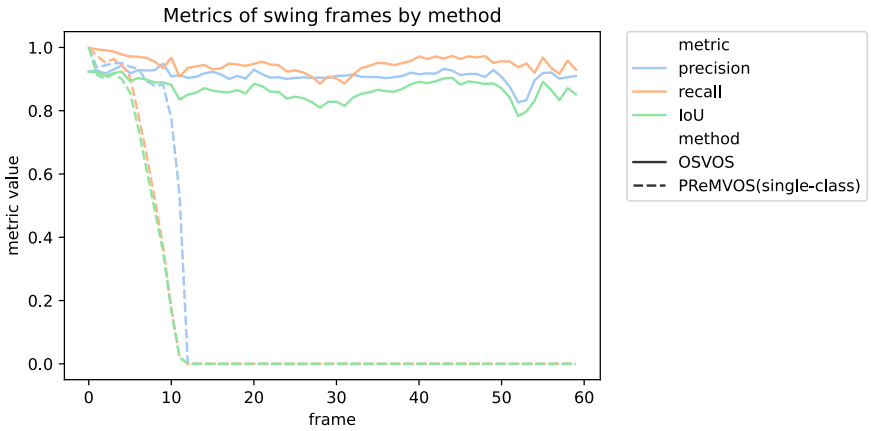


Figura 5.13: Métricas do vídeo `swing` por modelo

A Figura 5.12(a) mostra como a segmentação obtida pelo modelo OSVOS não diminui a precisão apesar da oclusão temporária, enquanto a Figura 5.12(b) mostra que a segmentação obtida pelo modelo PReMVOS perde precisão quando o objeto passou por oclusão. O gráfico mostrado na Figura 5.13 traduz essa percepção a partir das métricas ao longo dos *frames* do vídeo `swing`.

5.1.2 *Multiclass* VOS

A Figura 5.14 mostra as métricas de *recall*, *precision* e *IoU* para os vídeos com maior *recall* médio dos resultados do modelo PReMVOS na tarefa de segmentar mais de um objeto. Para essa execução nenhuma adaptação na entrada do modelo PReMVOS foi feita, diferente do *pipeline* executado para gerar os resultados da subseção 5.1.1. As análises abaixo mostram os resuldados considerando apenas os vídeos com mais de um objeto a ser segmentado, que são 57 dos 90 vídeos do *dataset* utilizado.

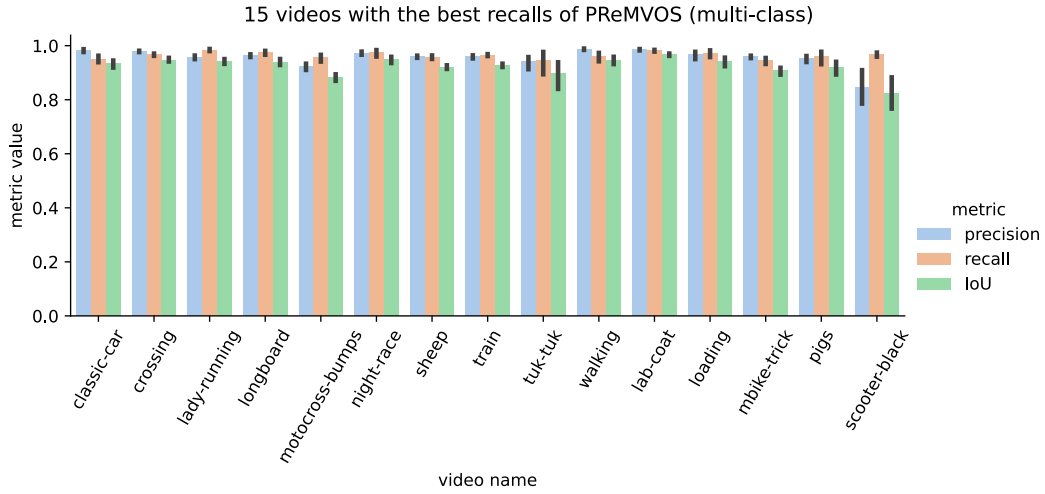


Figura 5.14: Métricas dos vídeos *multiclass* com maiores valores de *recall* para o modelo PReMVOS

Apesar do *pipeline* ser executado para todos os vídeos do *dataset*, a Figura 5.14 mostra os 15 melhores resultados para vídeos *multiclass* (*i. e.* vídeos que possuem mais de um objeto a ser segmentado).

Ainda na Figura 5.14 é notável que o vídeo `scooter-black` tem valores de *precision* e *IoU* consideravelmente abaixo do *recall*. A Figura 5.15 mostra a segmentação gerada pelo modelo. Os objetos a serem segmentados são uma scooter e seu motorista. Entretanto, a partir de certo *frame*, um carro é classificado erroneamente como a scooter. Esse tipo de falha diminui o valor das métricas *precision* e *IoU*.

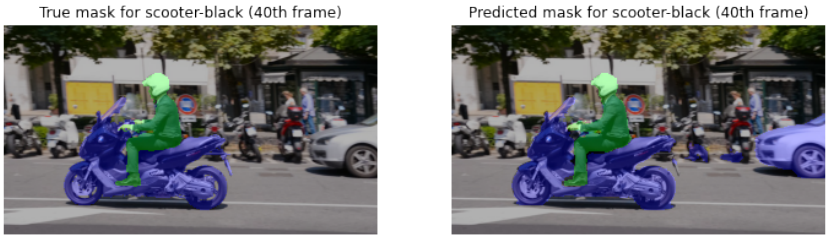


Figura 5.15: Comparação do resultado verdadeiro e do modelo PReMVOS aplicado ao vídeo `scooter-black`

A Figura 5.16 mostra as métricas de *recall*, *precision* e *IoU* para os vídeos com menor *recall* médio dos resultados do modelo PReMVOS na tarefa de segmentar mais de um objeto.

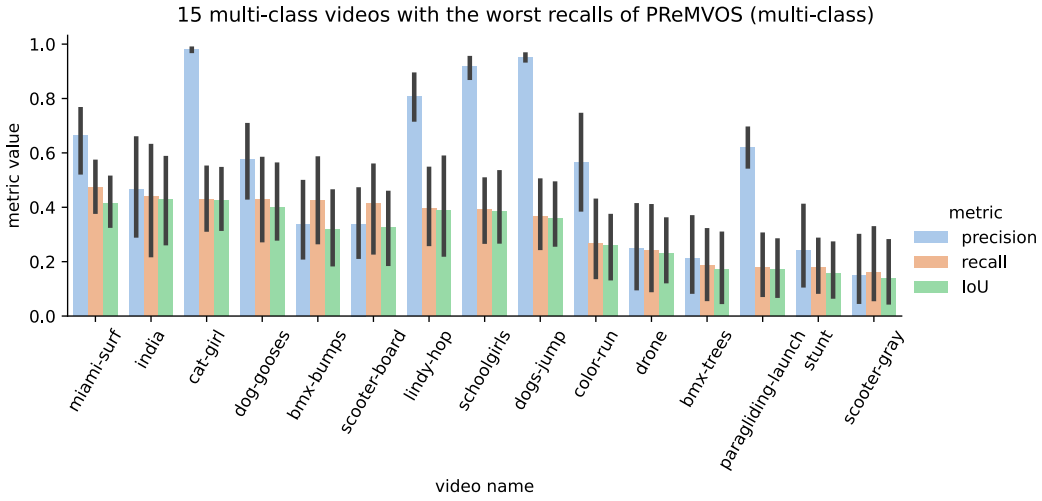


Figura 5.16: Métricas dos vídeos *multiclass* com menores valores de *recall* para o modelo PReMVOS

A partir do resultado mostrado na Figura 5.16 é possível notar alguns exemplos onde a métrica de *precision* se manteve alta enquanto as outras métricas não. Um desses vídeos é o `lindy-hop`, cujo resultado do modelo é mostrado na Figura 5.17.

A Figura 5.17 mostra a máscara esperada e a obtida pelo modelo PReMVOS em alguns *frames* do vídeo `lindy-hop`. O vídeo possui 8 objetos para serem segmentados. Analisando o resultado do modelo PReMVOS é

possível notar como alguns dos objetos deixam de ser segmentados ao longo do vídeo. Segundo a Figura 5.17, no 1^o *frame* o modelo segmenta os 8 objetos, no 20^o *frame* o modelo passa a identificar 6 objetos e no 30^o *frame* o modelo identifica apenas 4 objetos.

Figura 5.17: Resultado do modelo PReMVOS aplicado ao vídeos *multiclass* `lindy-hop`

5.2 Resultados da etapa de VI

Nessa seção vamos explorar os resultados obtidos da etapa de VI. Para explorar esses resultados vamos comparar os resultados preditos (*i.e.* quando o modelo recebe como máscara o resultado de uma modelo VOS) com resultados pseudo-verdadeiros (*i.e.* quando o modelo recebe como máscara as anotações verdadeiras).

A expectativa da análise é medir o quanto máscaras preditas podem interferir no resultado do modelo de VI. Para isso vamos utilizar a métrica *SSIM*, que é uma métrica de análise de similaridade estrutural entre imagens.

Nesse caso também iremos expor as diferenças entre as remoções de um ou múltiplos objetos dos vídeos.

5.2.1 *Singleclass* VI

A Figura 5.18 mostra a métrica de MSSIM nos 15 vídeos com maior MSSIM e que foram reconstruídos utilizando como anotação o resultado do modelo OSVOS.

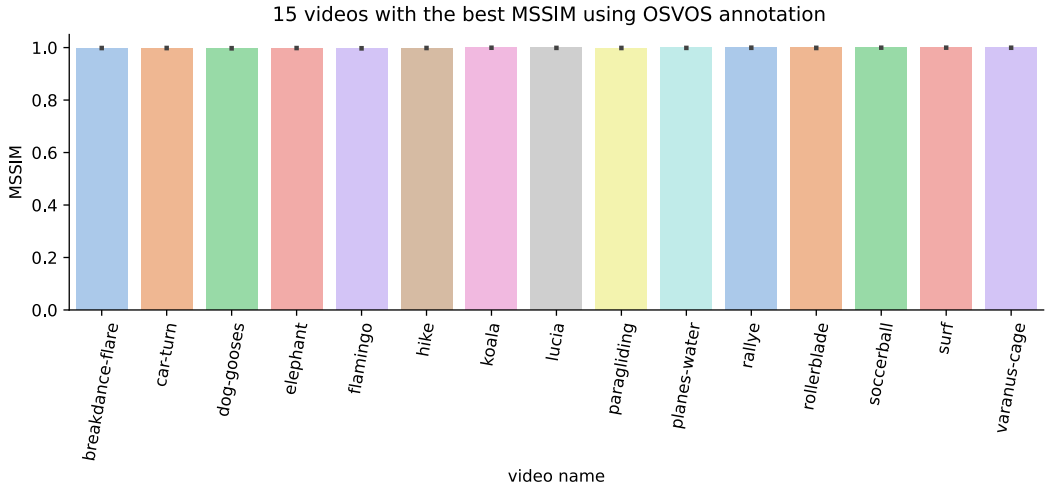
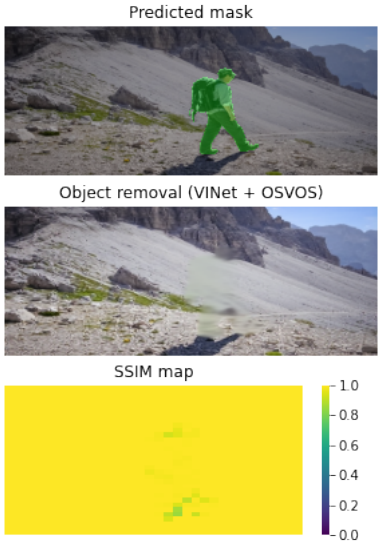
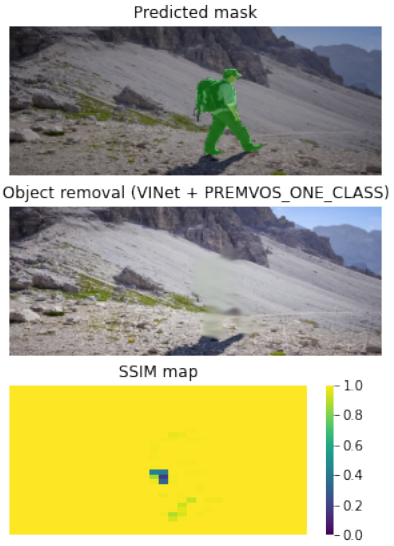


Figura 5.18: Distribuição do MSSIM dos 15 vídeos, gerados com a anotação do modelo OSVOS, com maior valor de MSSIM médio.

Dentre os vídeos mostrados na Figura 5.18, vamos explorar um dos resultados. A Figura 5.19 mostra o resultado do modelo VINet aplicado ao vídeo `hike`. É possível notar na Figura 5.19 que o resultado de ambos os modelos é muito similar, entretanto, analisando o mapa de SSIM, notamos uma região onde o resultado que utilizar a máscara do PReMVOS obteve menor similaridade ao resultado pseudo-verdadeiro. Analisando o resultado na Figura 5.19(b) é possível notar que modelo PReMVOS deixou de incluir uma pequena região da mochila da imagem, e isso impactou o resultado do modelo de VI quando comparado com o resultado pseudo-verdadeiro.



5.19(a): Resultado utilizando modelo OSVOS



5.19(b): Resultado utilizando modelo PReMVOS

Figura 5.19: Resultados do vídeo `hike`: Máscara predita pelo modelo; resultado do modelo VINet utilizando a máscara predita; Mapa de SSIM.

A Figura 5.20 mostra a métrica de MSSIM nos 15 vídeos com menores MSSIM e que foram reconstruídos utilizando como anotação o resultado do modelo OSVOS.

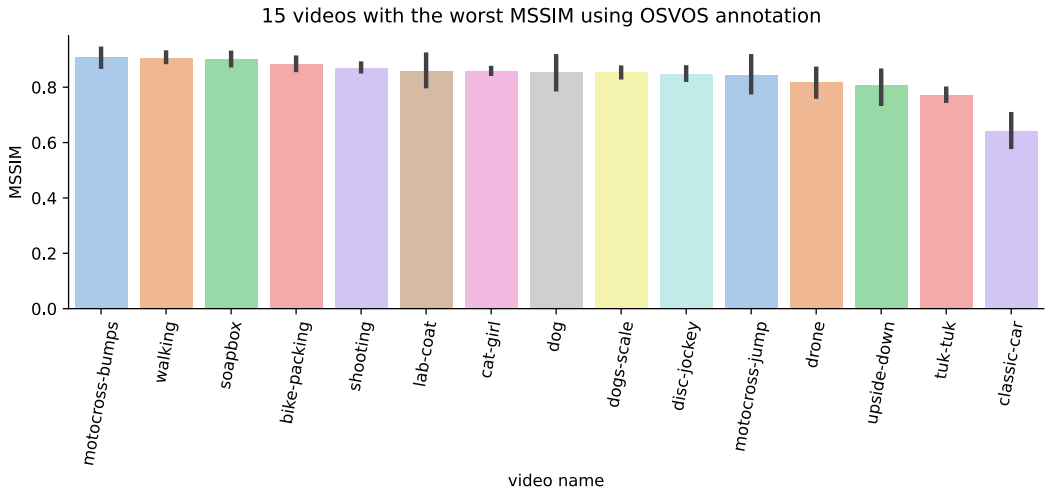
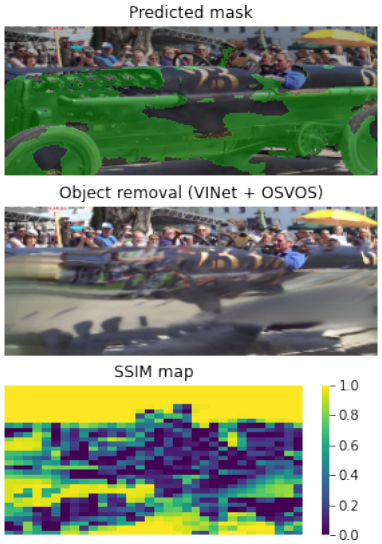
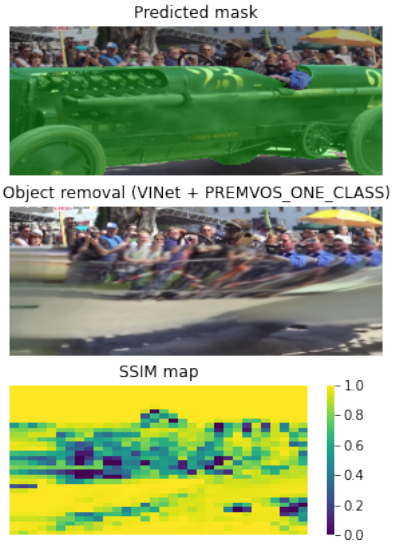


Figura 5.20: Distribuição do MSSIM dos 15 vídeos, gerados com a anotação do modelo OSVOS, com menores valor de MSSIM médio.

Dentre os vídeos mostrados na Figura 5.20, vamos explorar um dos de mais baixo MSSIM. A Figura 5.21 mostra o resultado do modelo VINet aplicado ao vídeo `classic-car`, que obteve o mais baixo MSSIM. A Figura 5.21 mostra o resultado a parir do resultado dos dois modelos (*i.e.* OSVOS e PReMVOS) e é possível notar que, pelo mapa de similaridade, a utilização do modelo PReMVOS gera um resultado mais similar ao pseudo-verdadeiro.



5.21(a): Resultado utilizando modelo OSVOS



5.21(b): Resultado utilizando modelo PReMVOS

Figura 5.21: Resultados do vídeo `classic-car`: Máscara predita pelo modelo; resultado do modelo VINet utilizando a máscara predita; Mapa de SSIM.

Entre os resultados medianos, vimos que no vídeo `parkour` o modelo OSVOS tem menos precisão do que o modelo PReMVOS ao longos dos *frames* do vídeo, como é mostrado na Figura 5.22. Entretanto, essa queda de precisão afeta sutilmente o resultado final, como mostrado na Figura 5.23.

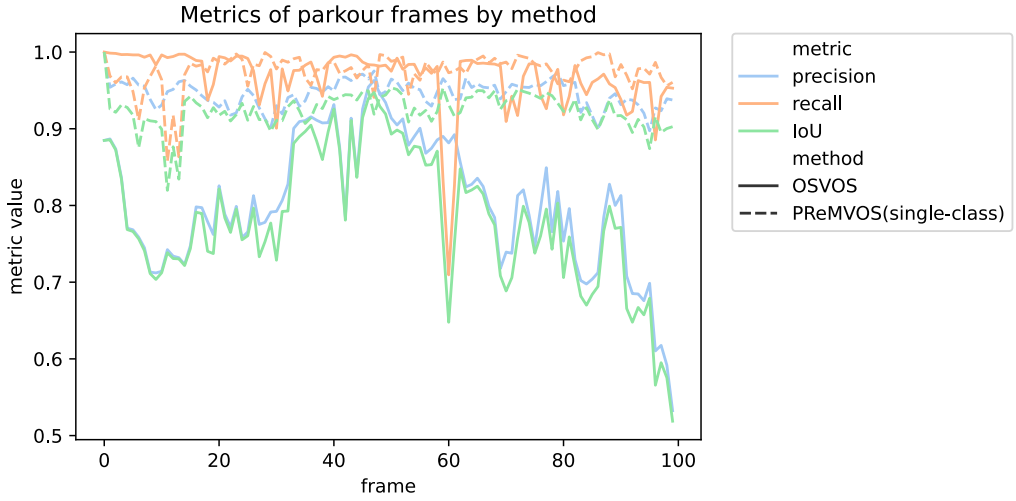
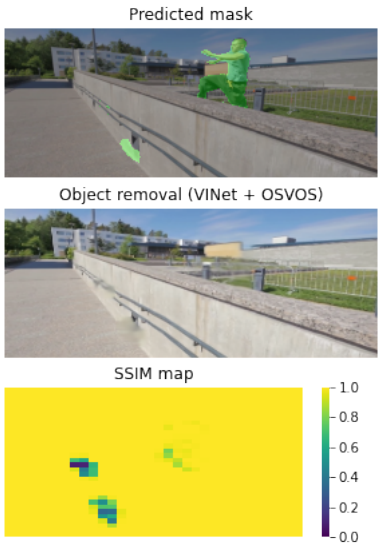
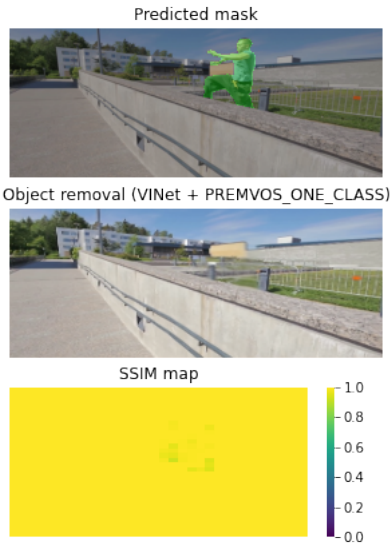


Figura 5.22: Métricas obtidas na aplicação dos modelos de VOS ao longo dos *frames* do vídeo `parkour`



5.23(a): Resultado utilizando modelo OSVOS



5.23(b): Resultado utilizando modelo PReMVOS

Figura 5.23: Resultados do vídeo `parkour`: Máscara predita pelo modelo; resultado do modelo VINet utilizando a máscara predita; Mapa de SSIM.

A Figura 5.24 mostra as métricas de MSSIM nos 15 vídeos com menores MSSIM. A avaliação é feita a partir do resultado do modelo VINet utilizando as anotações geradas pelo modelo PReMVOS.

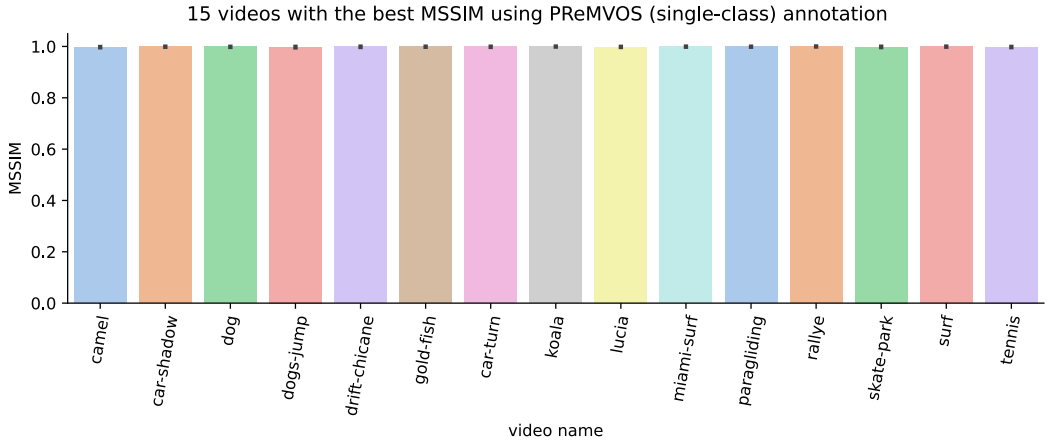
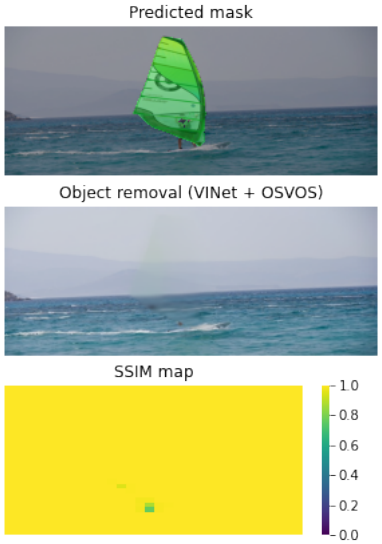
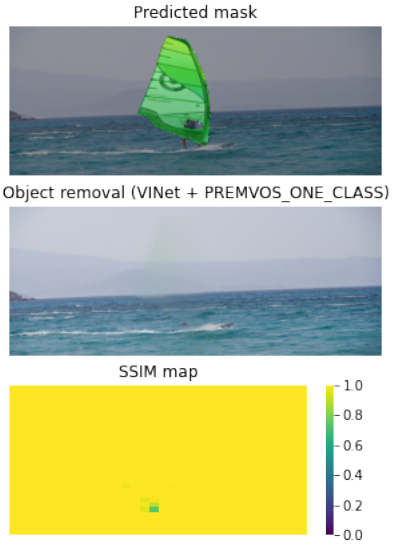


Figura 5.24: Distribuição do MSSIM dos 15 vídeos, gerados com a anotação do modelo PReMVOS, com maiores valor de MSSIM médio.

Dentre os vídeos mostrados na Figura 5.24, vamos explorar um dos vídeos. A Figura 5.25 mostra o resultado do modelo VINet aplicado ao vídeo `surf`. Na Figura 5.25 é mostrado o resultado do vídeo `surf` utilizando ambos os modelos, onde é possível notar que o modelo influencia pouco no resultado final.



5.25(a): Resultado utilizando modelo OSVOS



5.25(b): Resultado utilizando modelo PReMVOS

Figura 5.25: Resultados do vídeo `surf`: Máscara predita pelo modelo PReMVOS; resultado do modelo VINet utilizando a máscara predita; Mapa de SSIM.

A Figura 5.26 mostra a métrica de MSSIM nos 15 vídeos com menores MSSIM e que foram reconstruídos utilizando como anotação o resultado do modelo PReMVOS.

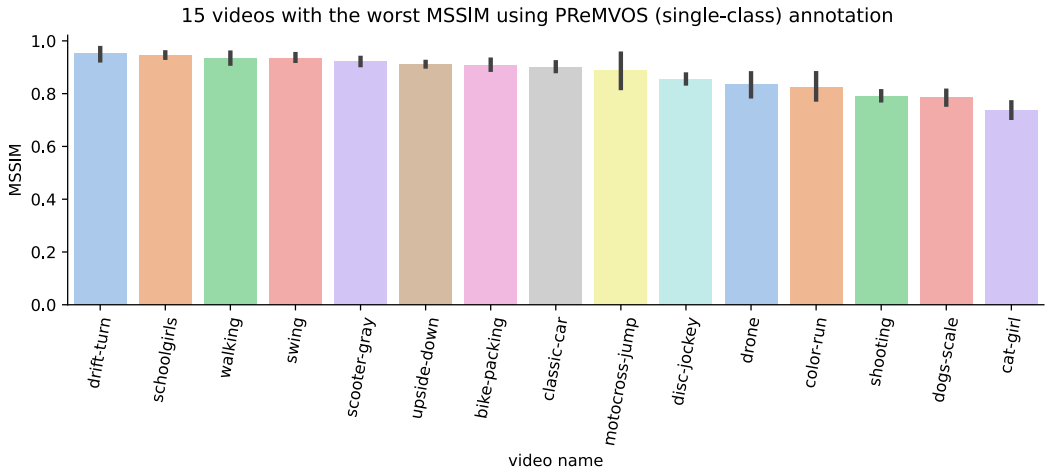
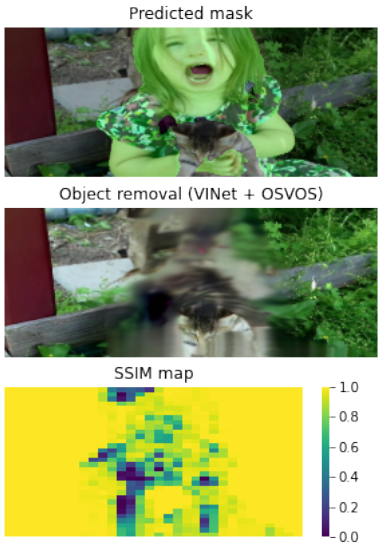


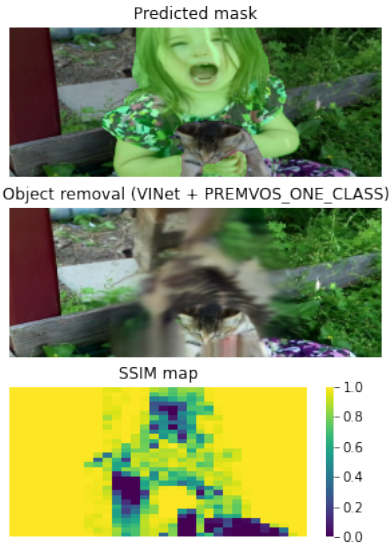
Figura 5.26: Distribuição do MSSIM dos 15 vídeos, gerados com a anotação do modelo PReMVOS, com menores valor de MSSIM médio.

Dentre os vídeos mostrados na Figura 5.26, vamos explorar um dos vídeos. A Figura 5.27 mostra o resultado do modelo VINet aplicado ao vídeo `cat-girl`, o de menor MSSIM.

Na Figura 5.27 estamos comparando a aplicação dos modelos OSVOS e PReMVOS no vídeo `cat-girl` para avaliarmos como o resultado de mais baixo MSSIM quando utilizamos o modelo PReMVOS se comporta com a utilização do modelo OSVOS. Nesse exemplo o modelo VI aparenta dificuldade de reconstruir a área de maneira convincente, independente do modelo de VOS utilizado para estimativa das máscaras.



5.27(a): Resultado utilizando modelo OSVOS



5.27(b): Resultado utilizando modelo PReMVOS

Figura 5.27: Resultados do vídeo `cat-girl`: Máscara predita pelo modelo; resultado do modelo VINet utilizando a máscara predita; Mapa de SSIM.

5.2.2 *Multiclass* VI

Nessa seção vamos nos concentrar nos resultados do modelo VINet na remoção de mais de um objeto do vídeo. Para isso estamos utilizando o modelo PReMVOS para gerar as máscaras das área de remoção dos vídeos *multiclass*.

A Figura 5.28 mostra a métrica de MSSIM nos 15 vídeos *multiclass* com maiores MSSIM e que foram reconstruídos utilizando como anotação o resultado do modelo PReMVOS. Nesse caso foram considerados 57 dos 90

vídeos do *dataset* que possuem mais de um objeto para ser segmentado e removido.

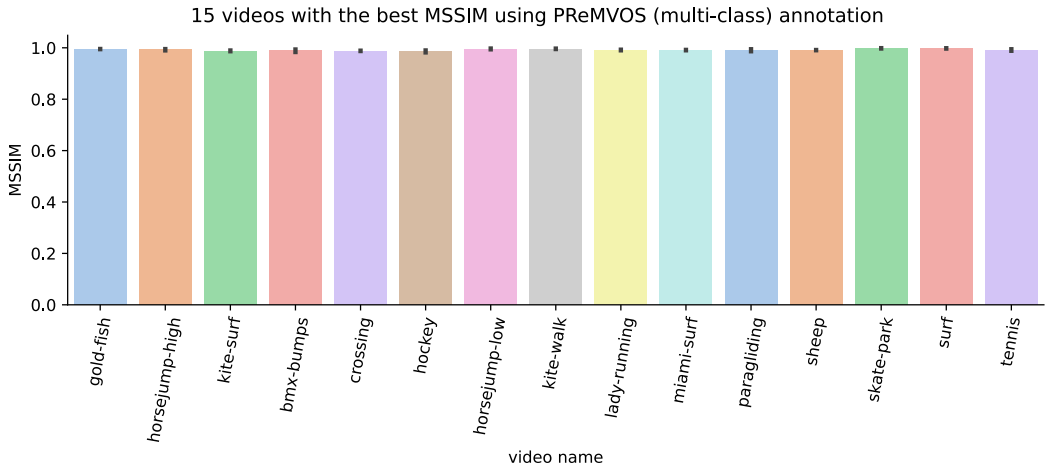


Figura 5.28: Distribuição dos valores de MSSIM dos 15 vídeos *multiclass* com maior valor de MSSIM médio, gerados pelo modelo VINet utilizando o resultado do modelo PReMVOS.

A Figura 5.29 mostra o resultado do modelo VINet aplicado ao vídeo `tennis` quando recebe como entrada o resultado do modelo PReMVOS. Nesse caso o modelo PReMVOS não segmenta corretamente a raquete do tenista, logo no mapa de similaridade a região da raquete aparece como pouco similar. Apesar disso, a área que ficou afetada pela má reconstrução da imagem ainda é pequena quando consideramos a imagem por completo.

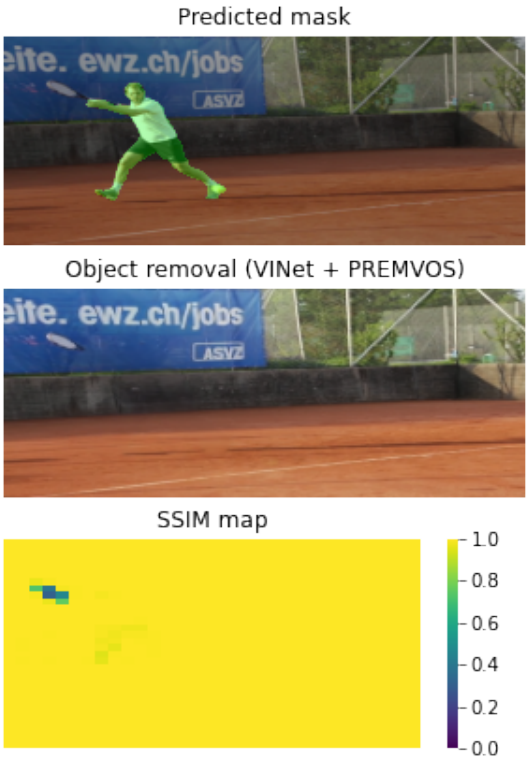


Figura 5.29: Resultados do vídeo *multiclass* `tennis`: Máscara predita pelo modelo PReMVOS; resultado do modelo VINet utilizando a máscara predita Mapa de SSIM.

A Figura 5.30 mostra a métrica de MSSIM nos 15 vídeos *multiclass* com menores MSSIM e que foram reconstruídos utilizando como anotação o resultado do modelo PReMVOS.

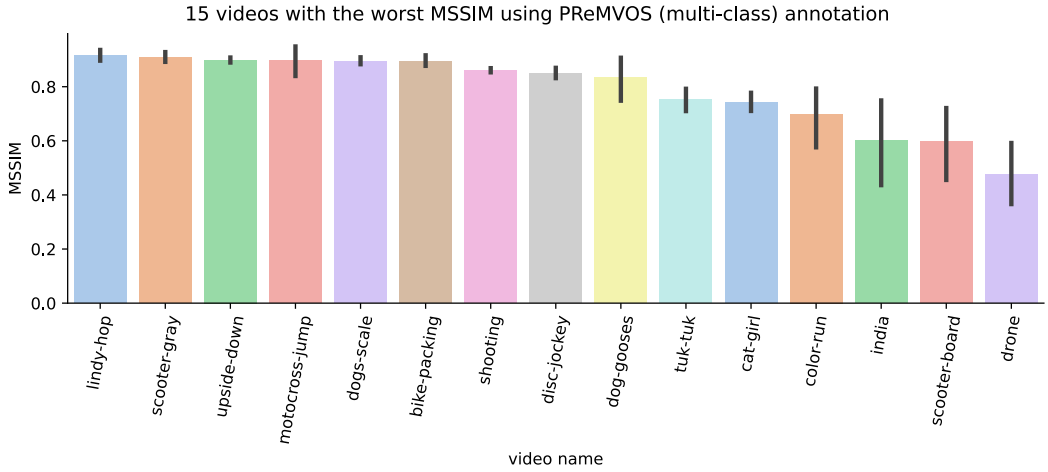


Figura 5.30: Distribuição dos valores de MSSIM dos 15 vídeos *multiclass* com menores valor de MSSIM médio, gerados pelo modelo VINet utilizando o resultado do modelo PReMVOS.

A Figura 5.31 mostra o resultado do modelo VINet aplicado ao vídeo `drone` quando recebe como entrada o resultado do modelo PReMVOS. Esse vídeo possui 5 objetos para serem segmentados, que sofrem oclusão ao longo do vídeo. A Figura 5.31 mostra um *frame* onde o modelo PReMVOS consegue segmentar apenas 2 objetos, sendo um segmentado de maneira errada (*i.e.* a área em verde estava representando outro objeto em *frames* anteriores e passou a demarcar uma área errada).

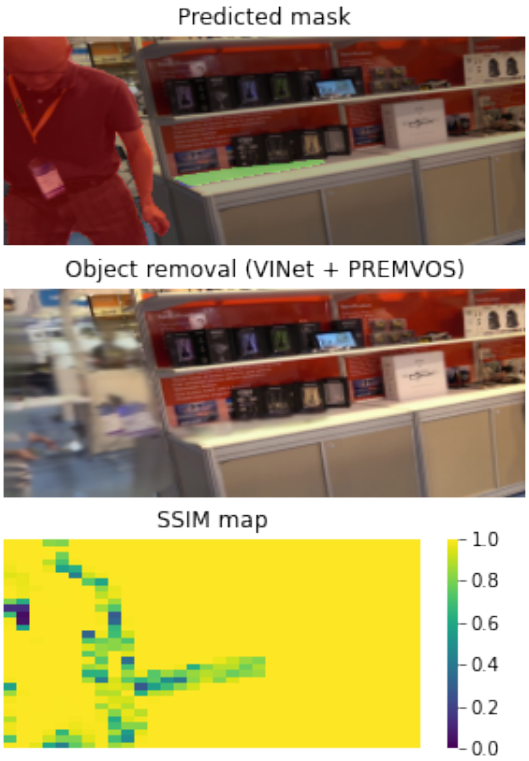


Figura 5.31: Resultados do vídeo *multiclass* `drone`: Máscara predita pelo modelo PReMVOS; resultado do modelo VINet utilizando a máscara predita; Mapa de SSIM.

6
Conclusão e Trabalhos Futuros

Este trabalho propôs a utilização de modelos VOS para simplificar a aplicação de modelos de VI. Para a tarefa de VOS dois modelos foram testados enquanto para a tarefa de VI um modelo foi utilizado.

Para solucionar essa tarefa propomos utilizar modelos de VOS Semi-supervisionados. Escolhemos dois modelos desse tipo para comparar seus resultados quando aplicados a esta tarefa: o modelo OSVOS e o modelo PReMVOS.

Modelo de VOS Semi-supervisionados baseados em CNN são classificados em dois tipos: baseados em movimento ou baseados em aparência. Nesse trabalho testamos um modelo de cada tipo.

Como modelo baseado em aparência utilizamos o modelo OSVOS. Esse tipo de modelo busca aprender características do objeto que esta segmentando para que possa encontra-lo nos outros *frames*. Quando esse modelo foi proposto em 2017 se tornou o novo estado-da-arte na solução do problema de VOS Semi-supervisionados, o que nos estimulou a aplica-lo neste trabalho. Atualmente é um modelo de referência quando novos modelos para solucionar essa tarefa são desenvolvidos.

Uma limitação do modelo OSVOS é que ele segmenta apenas um objeto por vídeo, entretanto já existem outros modelos que superam essa limitação, como o Multi-OSVOS. Outro problema é a possível falta de coerência temporal das segmentações ao longo do vídeo, uma vez que ele não utiliza atributos sobre a movimentação no vídeo.

Como modelo baseado em movimento utilizamos o modelo PReMVOS. Esse tipo de modelo utiliza atributos de movimentação do vídeo para encarar a tarefa como uma propagação de máscara. O modelo PReMVOS nos despertou interesse ao mostrar ter bons resultados e vencer a competição *YouTube VOS Challenge 2018*. Um possível problema encontrado nesse tipo de modelo é a propagação de erros ao longo do vídeo. O exemplo mostrado na Figura 5.15 mostra como um erro de segmentação foi propagado até que uma máscara passe a segmentar outro objeto fora do objetivo.

Podemos observar na seção 5.1 que em casos de oclusão o modelo PReMVOS não propaga a máscara corretamente, enquanto o modelo OSVOS

consegue lidar com oclusões. A Figura 5.12 e o gráfico mostrado na Figura 5.13 é um exemplo onde a oclusão do objeto a ser segmentado gerou uma falha no resultado do modelo PReMVOS, enquanto o modelo OSVOS não teve problemas.

Por outro lado, caso o objeto no *frame* anotado sofra deformações ou mudanças de perspectiva ao longo do vídeo, o modelo OSVOS apresenta dificuldade em identificar o objeto anotado, uma vez que o objeto de referência mudou ao longo do vídeo. Nesse cenário o modelo PReMVOS consegue manter a segmentação coerente, uma vez que utiliza atributos de movimento. Um exemplo desse caso é a Figura 5.10 e o gráfico mostrado na Figura 5.11, que mostram um exemplo onde o modelo OSVOS tem uma queda em suas métricas ao longo do vídeo, enquanto o modelo PReMVOS manteve resultados satisfatórios.

Para a tarefa de VI, que foi aplicada com objetivo de remover o objeto segmentado do vídeo, escolhemos o modelo VINet. Escolhemos esse modelo por ter apresentado bons resultados quando comparado com outros modelos.

Como trabalho futuro sugerimos a utilização de modelos baseados em movimento em um vídeos sem oclusão do objeto e modelos baseado em detecção em vídeos com oclusão. Esse tipo de seleção pode melhorar significativamente o resultado final.

Outra evolução possível é inclusão de anotações intermediárias para ajustar problemas na etapa de segmentação. Uma ideia é estudar a viabilidade de evoluir os modelos de VOS para sinalizar quando é necessário corrigir a segmentação. Caso seja viável, o modelo pode sinalizar para o usuário quando seu resultado deve ser corrigido. Esse tipo de estudo pode ser conduzido incluindo modelo de VOS interativos.

Algo que não foi incluído neste trabalho, mas pode ser discutido em trabalhos futuros é a maneira que os modelos de VI preenchem áreas que estão oclusas durante o vídeo todo. No vídeo do exemplo mostrado na Figura 5.27, a menina que deve ser removida do vídeo permanece o vídeo todo a frente de uma região do segundo plano. Seria interessante estudar a viabilidade do modelo VI sinalizar que não tem informações o suficiente do segundo plano para remover determinado objeto.

Ainda como trabalho futuro sugerimos a implantação de um modelo de VOS mais simples, uma vez que notamos que a precisão as máscaras de VOS, quando o objetivo é aplicar modelos de VI, interfere de maneira sutil no resultado, como mostrado na Figura 5.23. Uma avaliação interessante seria a de estudar como a utilização de máscaras rústica com um valor de *recall* satisfatório, sem passar por etapas de refinamento, pode impactar no resultado

do modelo de VI.

Referências bibliográficas

[1] EBDELLI, M.. **Video inpainting techniques : application to object removal and error concealment**. 2014.

[2] YAO, R.; XIA, S.; ZHAO, J.; ZHOU, Y. ; LIN, G.. **Video Object Segmentation and Tracking: A Survey**. Technical report, 2019.

[4] LUITEN, J.; VOIGTLAENDER, P. ; LEIBE, B.. **PReMVOS: Proposal-generation, Refinement and Merging for Video Object Segmentation**. Technical report.

[5] CAELLES, S.; MANINIS, K.-K.; PONT-TUSET, J.; LEAL-TAIXÉ, L.; CREMERS, D.; VAN GOOL, L.; ZÜRICH, E. ; MÜNCHEN, T. U.. **One-Shot Video Object Segmentation**. Technical report, 2017.

[6] KIM, D.; WOO, S.; LEE, J.-Y. ; RESEARCH, A.. **Deep Video Inpainting**. Technical report, 2019.

[7] PONT-TUSET, J.; PERAZZI, F.; CAELLES, S.; ARBELÁEZ, P.; SORKINE-HORNUNG, A. ; VAN GOOL, L.. **The 2017 davis challenge on video object segmentation**. arXiv:1704.00675, 2017.

[8] VOIGTLAENDER, P.; KRAUSE, M.; SA O$^{\smile}$ SEP, A.; LUITEN, J.; BALACHANDAR, B.; SEKAR, G.; GEIGER, A. ; LEIBE, B.. **MOTS: Multi-Object Tracking and Segmentation**. Technical report.

[9] LI, Z.; HOIEM, D.. **Learning without Forgetting**. Technical report.

[10] ZHANG, A.; LIPTON, Z. C.; LI, M. ; SMOLA, A. J.. **Dive into Deep Learning**. 2020. `https://d2l.ai`.

[11] DENG, J.; DONG, W.; SOCHER, R.; LI, L.-J.; LI, K. ; FEI-FEI, L.. **ImageNet: A Large-Scale Hierarchical Image Database**. In: CVPR09, 2009.

[12] LE, T. T.; ALMANSA, A.; GOUSSEAU, Y. ; MASNOU, S.. **Object removal from complex videos using a few annotations**. 5(3):267–291, 2019.

[13] KHOREVA, A.; BENENSON, R.; ILG, · . E.; BROX, T.; SCHIELE, B.; GOOGLE, R. B. ; ILG, E.. **Lucid Data Dreaming for Video Object Segmentation**. Technical report.

[16] CHEN, L.-C.; PAPANDREOU, G.; MEMBER, S.; KOKKINOS, I.; MURPHY, K. ; YUILLE, A. L.. **DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs**. Technical report.

[17] LEZAMA, J.; ALAHARI, K.; SIVIC, J. ; LAPTEV, I.. **Track to the future: Spatio-temporal video segmentation with long-range motion cues**. p. 3369–3376. Institute of Electrical and Electronics Engineers (IEEE), oct 2011.

[18] BEAUCHEMIN, S.; BARRON, J.. **The computation of optical flow.** ACM Computing Surveys (CSUR), 27:433–466, 09 1995.

[19] SUN, D.; YANG, X.; LIU, M.-Y. ; NVIDIA, J. K.. **PWC-Net: CNNs for Optical Flow Using Pyramid, Warping, and Cost Volume**. Technical report.

[20] BAKER, S.; ROTH, S.; SCHARSTEIN, D.; BLACK, M. J.; LEWIS, J. P. ; SZELISKI, R.. **A database and evaluation methodology for optical flow**. In: 2007 IEEE 11TH INTERNATIONAL CONFERENCE ON COMPUTER VISION, p. 1–8, 2007.

[22] HE, K.; GKIOXARI, G.; DOLLÁR, P. ; GIRSHICK, R. B.. **Mask R-CNN**. CoRR, abs/1703.06870, 2017.

[23] HE, K.; ZHANG, X.; REN, S. ; SUN, J.. **Deep residual learning for image recognition**. CoRR, abs/1512.03385, 2015.

[24] CHEN, L.; ZHU, Y.; PAPANDREOU, G.; SCHROFF, F. ; ADAM, H.. **Encoder-decoder with atrous separable convolution for semantic image segmentation**. CoRR, abs/1802.02611, 2018.

[25] WU, Z.; SHEN, C. ; VAN DEN HENGEL, A.. **Wider or deeper: Revisiting the resnet model for visual recognition**. CoRR, abs/1611.10080, 2016.

[26] LIN, T.; MAIRE, M.; BELONGIE, S. J.; BOURDEV, L. D.; GIRSHICK, R. B.; HAYS, J.; PERONA, P.; RAMANAN, D.; DOLLÁR, P. ; ZITNICK, C. L.. **Microsoft COCO: common objects in context**. CoRR, abs/1405.0312, 2014.

[28] PÉREZ, P.; GANGNET, M. ; BLAKE, A.. **Poisson Image Editing**. Technical report, 2003.

[29] LE, T.; ALMANSA, A.; GOUSSEAU, Y. ; MASNOU, S.. **Motion-consistent video inpaint-ing**. 2017.

[30] WANG, Z.; BOVIK, A.; SHEIKH, H. ; SIMONCELLI, E.. **Image quality assessment: From error visibility to structural similarity**. Image Processing, IEEE Transactions on, 13:600 – 612, 05 2004.

[31] HAN, B.; DAVIS, L.. **Density-based multifeature background subtraction with support vector machine**. IEEE transactions on pattern analysis and machine intelligence, 34:1017–23, 12 2011.

[32] MANINIS, K.; CAELLES, S.; PONT-TUSET, J. ; GOOL, L. V.. **Deep extreme cut: From extreme points to object segmentation**. CoRR, abs/1711.09081, 2017.

[33] HARTMANN, G.; GRUNDMANN, M.; HOFFMAN, J.; TSAI, D.; KWATRA, V.; MADANI, O.; VIJAYANARASIMHAN, S.; ESSA, I.; REHG, J. ; SUKTHANKAR, R.. **Weakly supervised learning of object segmentations from web-scale video**. volumen 7583, p. 198–208, 10 2012.

[34] WANG, Q.; ZHANG, L.; BERTINETTO, L.; HU, W. ; TORR, P. H. S.. **Fast online object tracking and segmentation: A unifying approach**. CoRR, abs/1812.05050, 2018.

[35] HUANG, J.-B.; KANG, S. B.; AHUJA, N. ; KOPF, J.. **Temporally coherent completion of dynamic video**. ACM Transactions on Graphics (TOG), 35(6):196, 2016.