UNIVERSIDADE ESTADUAL DE MARINGÁ
CENTRO DE TECNOLOGIA
DEPARTAMENTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

ANDRÉ PERES RAMOS

Colorização de sequências de imagens em tons de cinza utilizando descritores de textura e segmentação hierárquica.

Maringá
2018

ANDRÉ PERES RAMOS

Colorização de sequências de imagens em tons de cinza utilizando descritores de textura e segmentação hierárquica.

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Departamento de Informática, Centro de Tecnologia da Universidade Estadual de Maringá, como requisito parcial para obtenção do título de Mestre em Ciência da Computação.

Orientador: Prof. Dr. Franklin César Flores

Maringá
2018

Dados Internacionais de Catalogação na Publicação (CIP)
(eDOC BRASIL, Belo Horizonte/MG)

Ramos, André Peres.
R175c Colorização de sequências de imagens em tons de cinza utilizando descritores de textura e segmentação hierárquica / André Peres Ramos. – Maringá (PR), 2018.
93 f.

Orientador: Franklin César Flores

Dissertação (Mestrado em Ciência da Computação) – Universidade Estadual de Maringá.

1. Ciência da computação. 2. Processamento de imagens. 3. Fotografia – Coloração – Técnicas digitais. I. Flores, Franklin César. II. Universidade Estadual de Maringá. III. Título.

CDU 004.62

Elaborado por Maurício Amormino Júnior – CRB6/2422

FOLHA DE APROVAÇÃO

ANDRÉ PERES RAMOS

Colorização de sequências de imagens em tons de cinza utilizando descritores de textura e segmentação hierárquica

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Departamento de Informática, Centro de Tecnologia da Universidade Estadual de Maringá, como requisito parcial para obtenção do título de Mestre em Ciência da Computação pela Banca Examinadora composta pelos membros:

BANCA EXAMINADORA

Prof. Dr. Franklin César Flores
Universidade Estadual de Maringá – DIN/UEM

Profa. Dra. Valéria Delisandra Feltrim
Universidade Estadual de Maringá – DIN/UEM

Prof. Dr. Fabrício Martins Lopes
Universidade Tecnológica Federal do Paraná – PPGI/UTFPR-CP

Aprovada em: 12 de dezembro de 2018.
Local da defesa: Sala 101, Bloco C56, *campus* da Universidade Estadual de Maringá.

AGRADECIMENTOS

Agradeço a Deus por ter me abençoado nesta caminhada e as pessoas que contribuíram na realização deste trabalho. Em especial:

A minha família pelo carinho, apoio e incentivo.

Ao meu orientador professor Dr. Franklin César Flores pelo apoio, comentários e sugestões no desenvolvimento deste projeto.

Aos demais professores das disciplinas que cursei e aos colegas.

E ao Instituto Federal do Paraná pela liberação das atividades laborais para que fosse possível realizar este trabalho.

Colorização de sequências de imagens em tons de cinza utilizando descritores de textura e segmentação hierárquica.

RESUMO

A colorização é considerada o processo de adicionar cores a imagens e filmes em escala de cinza. Atualmente há uma vasta quantidade de filmes e imagens em escala de cinza, Esses materiais, em muitos casos, demandam colorização uma vez que quando coloridos se tornam mais agradáveis ao ser humano, facilitando a percepção de detalhes da cena. A colorização manual é tarefa árdua, que exige do usuário a definição de uma determinada cor para cada ponto da imagem, o que normalmente é feito por segmentação manual tornando o processo lento. Desta forma, este trabalho propõe um método para realizar a colorização de vídeos de forma assistida. No método proposto nesta dissertação, o usuário define regiões e cores de um primeiro *frame*. Na sequência o algoritmo segmenta o *frame* atual e o próximo utilizando um algoritmo de segmentação automática que gera regiões de forma controlada, como segmentação hierárquica ou por superpixel. Extrai-se descritores de textura LBP e Gabor além de informações de intensidade e então se realiza a comparação de cada região entre os dois *frames* encontrando a melhor correspondência. Por fim, as cores são propagadas na imagem por meio de um algoritmo que utiliza uma técnica de otimização para gerar cores mais naturais. Os experimentos realizados demonstraram que o método proposto gerou resultados superiores em relação à técnica manual, bem como em relação a técnicas do estado da arte, tanto no que se refere ao tempo, ao número de interações do usuário e quanto a qualidade final.

Palavras-chave: Processamento de imagens, colorização, segmentação.

Grayscale image sequence colorization using texture descriptors and hierarchical segmentation.

ABSTRACT

Colorization is considered the process of adding color to grayscale images and movies. Currently there are a vast amount of films and images in grayscale, these materials, in many cases, require colorization since to become more human-friendly, facilitating the perception of details of the scene. Manual colorization is an arduous task, requiring the user to define a specific color for each point in the image, which is usually done by manual segmentation, making the process slow. In this way, this work proposes a method to perform colorization of videos in an assisted way. In the method proposed in this dissertation, the user defines regions and colors of a first frame. The algorithm then segments the current and next frame using an automatic segmentation algorithm that generates regions in a controlled way, such as hierarchical or superpixel segmentation. We extract LBP and Gabor texture descriptors in addition to intensity information and then we compare each region between the two frames to find the best match. Finally, colors are propagated in the image through an algorithm that uses an optimization technique to generate more natural colors. The experiments demonstrated that the proposed method yielded superior results in relation to the manual technique, as well as in relation to state-of-the-art techniques, in terms of time, number of user interactions and final quality.

Keywords: : Image processing, colorization, segmentation.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS E ABREVIATURAS

CNN: *Convolutional Neural Network* - Rede Neural Convolucional.

CMY: *Cyan, Magenta, Yellow* - Ciano, Magenta, Amarelo.

CMYK: *Cyan, Magenta, Yellow, Black.* - Ciano, Magenta, Amarelo e Preto.

CPSNR: *Color Peak Signal to Noise Ratio*

HSV: *Hue, Saturation, Value* - Matiz, Saturação, Valor.

HSL: *Hue, Saturation, Lightness* - Matiz, Saturação, Luminosidade.

L*A*B: Padrão de cor onde L representa a Luminância, A e B as informações de crominância.

LBP: *Local Binary Pattern.*

RGB: *Red, Blue, Green* - Vermelho, Azul e Verde.

YUV: Padrão de cor onde Y representa a Luminância, U e V as informações de crominância.

SUMÁRIO

1

Introdução

A colorização é considerada o processo de adicionar cores a imagens e vídeos em tons de cinza. Apesar de atualmente ser alvo de diversos estudos, a colorização de imagens é tão antiga quanto a própria fotografia, sendo que há registros históricos, de colorização à mão, que datam desde 1842 (Yatziv e Sapiro, 2006).

Para os filmes, a colorização também existiu antes mesmo de haver técnicas de capturas coloridas, os *frames* eram coloridos de forma manual, um a um. Filmes datados desde 1902 como o filme *A Trip to the Moon* (A TRIP ..., 1902), tiveram versões coloridas de forma manual. Os laboratórios de colorização normalmente envolviam diversas pessoas e funcionavam como uma linha de montagem, de forma que cada indivíduo era responsável por uma das cores aplicadas (Duval e Wemaere, 2011).

Em 1932 a empresa Technicolor fabricou a câmera de três rolos de filme, a primeira a ser capaz de gravar os três canais de cores (Azul, Verde e Vermelho), entretanto a mesma se mostrou cara e extremamente complexa de operar, sendo que era exigida uma equipe inteira fora o seu tamanho que limitava as ações das cenas (Ball, 1935). Tudo isso limitou sua adoção em larga escala. Somente no final dos anos de 1960 que efetivamente os filmes começaram a serem capturados e exibidos de forma autenticamente colorida. Até então todo processo de colorização era manual (Yumibe, 2012). Na década de 1970, começaram a aparecer algumas técnicas de colorização digital, porém, ainda assim, de forma manual ou então usando técnicas de mapeamento de níveis de cinza, aplicados a cada *frame* (Ball, 1935). Mesmo atualmente, ainda há dificuldades de se realizar a colorização de vídeos de forma eficiente, apesar de diversos estudos direcionados a esse problema.

Com isso, boa parte das fotos e vídeos produzidos eram capturados e armazenados em escalas de cinzas, desprovidos de qualquer informação de cor, sendo apenas capturada a

informação de intensidade ou luminância da cena. Como as cores captadas pelos olhos e reconhecidas pelo cérebro humano são constituídas de três componentes primitivas (vermelho, verde e azul, respectivamente) (Gonzalez e Woods, 2007), não é possível recuperar com precisão a cor original a partir de uma imagem em escala de cinza, pois somente está disponível uma informação: a intensidade.

Com isso, o trabalho de colorização originalmente é manual e árduo, pois necessita que o usuário defina a cor para cada área da imagem, o que normalmente é feito através de uma exaustiva segmentação manual de cada região que possua uma determinada cor, para que então possa aplicar efetivamente a cor ao objeto de interesse, sendo a colorização da imagem finalizada somente após realizada essa atividade para todas as regiões. Em caso de vídeos, o trabalho aumenta de forma significativa (Levin et al., 2004; Yatziv e Sapiro, 2006), uma vez que o vídeo é a composição de várias imagens em sequência, que faz com que o usuário repita o trabalho descrito anteriormente centenas de vezes para conseguir apenas alguns segundos de vídeo.

Desta forma, vários estudos são conduzidos na tentativa de diminuir o esforço necessário para realizar a colorização, na procura de métodos que auxiliem na distribuição de cor, segmentação e classificação. Os métodos de colorização podem ser automáticos, quando não há interferência do usuário no que se refere a escolha das cores ou formas dos objetos a serem coloridos. Já os semiautomáticos, são os que a intervenção do usuário é necessária em alguns ou diversos pontos, para que o algoritmo possa realizar a colorização. Apesar de todos estudos, a quantidade destes que se dedicam à vídeos é pequena se comparado aos voltados para imagens, havendo, por tanto, ainda muito a ser desenvolvido nessa área.

Este trabalho aborda o problema de colorização de vídeos, tarefa que pode ser descrita como, colorir uma sequência de imagens de uma mesma cena com informações semânticas similares.

Na abordagem proposta nesta dissertação, o usuário inicialmente irá colorir uma primeira imagem da sequência por meio de uma ferramenta interativa, que o auxiliará na segmentação e definição de cores para cada região da imagem. Após isso, será realizada uma segmentação automática em toda sequência de forma a particionar cada *frame* em centenas de regiões. Na sequência, descritores de textura, bem como informações relativas à intensidade, são extraídas para compor um vetor de características. É realizada uma comparação entre os vetores de cada região do *frame* previamente colorido, com os do *frame* no qual se deseja colorir a fim de encontrar a melhor correspondência e efetuar a transferência das cores. Por fim, as cores são propagadas na imagem por meio de um algoritmo de otimização. O usuário pode, então, analisar o resultado aprovando ou

alterando, corrigindo eventuais erros ou definindo cores para eventuais novos objetos que apareçam na cena. O processo irá se repetir para o *frame* seguinte, onde o *frame* recém colorido servirá de referência para o processo descrito.

1.1 Objetivos

O objetivo principal deste trabalho é a proposta de um método assistido que seja capaz de realizar a colorização de vídeos pelo usuário, com o mínimo possível de interferência do mesmo, utilizando descritores de textura e informações de intensidade.

1.1.1 Objetivos Específicos

- Avaliar métodos de segmentação de imagens existentes no que tange à sua aplicabilidade ao problema apresentado;
- Avaliar a capacidade descritiva de métodos de extração de características de imagens, para comparação e classificação;
- Avaliar métodos de distribuição de cores em imagens em tons de cinza de forma a torna-las mais natural;
- Propor um *benchmark* de avaliação de métodos interativos de colorização de sequências de imagens.

1.2 Organização

O presente trabalho está assim organizado: o Capítulo 2 apresenta à revisão bibliográfica realizada para este trabalho bem como os trabalhos relacionados; o Capítulo 3,abrange a fundamentação teórica dos conceitos técnicos e descreve técnicas relevantes ao problema; o Capítulo 4 descreve em detalhes o método proposto; o Capítulo 5 traz o resultado de diversos experimentos realizados; por fim, o Capítulo 6 apresenta as conclusões deste trabalho.

2

Revisão de literatura

Este capítulo traz alguns trabalhos relacionados ao problema de colorização de vídeos em tons de cinza, bem como a de colorização de imagens em tons de cinza, uma vez que, como comentado, um vídeo pode ser dividido em uma sequência de imagens.

2.1 Trabalhos Relacionados

No seu trabalho, Levin et al. (2004) introduziram uma técnica que trata o problema de colorização como um problema de otimização, no qual parte do preceito de que *pixels* próximos com intensidades semelhantes devem ter a mesma informação de cor. Para o funcionamento da técnica, o usuário deve fornecer algumas informações de cor. Tais informações são representadas como rabiscos (*scribles*) coloridos na imagem. A partir desses rabiscos, o algoritmo propaga as cores para os demais *pixels* usando um algoritmo de otimização. A técnica apresenta resultados interessantes sendo bem realístico em várias situações, nas quais o observador tem dificuldade de perceber que a colorização foi feita artificialmente. A principal desvantagem da técnica está na questão que, dependendo da imagem, podem ser necessárias muitas intervenções do usuário para se ter um resultado satisfatório.

O trabalho de Yatziv e Sapiro (2006) aprimorou a técnica de Levin et al. (2004) tratando a imagem como um grafo, no qual o peso das arestas é representado pela diferença de luminância entre os *pixels*. O problema é então tratado de forma a encontrar as menores distâncias entre os *pixels* sem informação de crominância e os que tiveram a informação fornecida pelo usuário, por meio da utilização do algoritmo de Dijkstra.

Na linha do trabalho de Levin et al. (2004), os trabalhos de Qu et al. (2006) e Luan et al. (2007) utilizam descritores de texturas para reduzir a quantidade de rabiscos fornecidos pelo usuário. Em ambos trabalhos foi conseguida uma redução significativa de rabiscos para algumas imagens, entretanto para outras ainda foi necessária uma maior intervenção do usuário.

Em seu trabalho, Cheng et al. (2015) propuseram a utilização de uma Rede Neural para realizar a colorização de forma automatizada. Primeiramente, descritores de textura são extraídos de cada *pixel* da imagem e a rede utiliza tais descritores tanto para realizar o treinamento quanto para realizar a classificação. Após treinada, dada uma imagem em tons de cinza, são extraídos descritores da mesma e enviados para a Rede Neural. Para cada descritor de cada *pixel*, a rede neural fornece uma informação de crominância. Como se trata de um problema de classificação, a maior limitação reside no vício da Rede Neural em fornecer cores se baseando no conjunto de treinamento, na qual imagens que fogem à semântica do conjunto tendem a ter resultados ruins.

O método desenvolvido por Zhang et al. (2016) utiliza uma Rede Neural Convolucional (CNN), abordando o problema de colorização como um problema de classificação, para a colorização de forma totalmente automática. A rede é treinada para mapear a distribuição das escalas de cinza da imagem de entrada para valores de cores por meio de uma função objetivo elaborada para esse fim. Em experimentos realizados com indivíduos, 32% classificaram as imagens coloridas pelo algoritmo como reais. Apesar de apresentar bons resultados, em alguns conjuntos de imagens, a distribuição de cor fica com aparência artificial em outros.

Por sua vez Iizuka et al. (2016) utilizaram um conjunto de CNN para efetuar a colorização, sendo uma rede para características de baixo nível, uma característica de médio nível, outra para características globais e finalmente uma para realizar a colorização. Como saída, a rede fornece as informações de crominância preditas da imagem. As principais vantagens residem na possibilidade de processar imagens de alta resolução sem impacto significativo no desempenho. Como limitação, a rede tem dificuldade de colorir imagens que possuam informações semânticas muito distintas do conjunto de treinamento.

Já o método proposto por Bugeau et al. (2014) aborda a colorização automática por meio da utilização de imagens similares. Para uma imagem em tons de cinza que se deseja colorir, o usuário fornece uma outra imagem já colorida que possui semântica similar. Com isto, o algoritmo tenta mapear, para cada pixel em tons de cinza, o pixel da imagem colorida que possui melhor correspondência, e então é realizada a transferência de cor. No método proposto, os autores tentaram minimizar uma função, bem como manter uma coerência espacial. O resultado se mostra satisfatório para alguns dos

exemplos selecionados. A maior limitação consiste no fato da imagem escolhida ter que ser semanticamente similar a em tons de cinza, além de possuir as cores desejadas pelo usuário.

Uma abordagem baseada em imagens semelhantes também foi proposta por Gupta et al. (2012). No método proposto, ambas imagens são segmentadas utilizando a técnica de segmentação por superpixel e descritores de textura e de luminância são extraídos de todos as regiões de ambas imagens. Para cada região da imagem em tons de cinza é identificada a região da imagem colorida que possui mais similaridade e suas informações de cor são transferidas. Por fim, utiliza-se o algoritmo proposto por Levin et al. (2004), para uma distribuição mais uniforme da cor. O método apresenta bons resultados, sendo que a maior limitação reside no fato de que a imagem tem que possuir similaridade semântica com a imagem em tons de cinza bem como as cores desejadas pelo usuário.

Como uma evolução do seu trabalho anterior, Gupta et al. (2017) inclui um algoritmo de aprendizagem de máquina para realizar a colorização automática. Da mesma forma que em seu trabalho anterior, a imagem é segmentada por Superpixel e descritores de textura e luminância são extraídos da imagem. Após isso um algoritmo de aprendizagem de máquina denominado *Randomized decision forest* é utilizando, sendo o mesmo treinado para conseguir predizer as cores de uma imagem monocromática qualquer. Os resultados variam muito, sendo que imagens com menor semelhança às usadas no treinamento tendem a ter piores resultado.

Utilizando rabiscos para colorir, o método proposto por Teng et al. (2013) seleciona determinados *frames* chave. A partir disto, são calculados os vetores de movimento entre os *frames* chave e os próximos *frames*, deslocando-se os rabiscos de forma a seguir os elementos da imagem.

No trabalho de Pierre et al. (2017) são utilizadas técnicas de propagação de cor junto com a utilização de rabiscos. Primeiro é realizada a colorização de um determinado *frame*. A partir dessa, o algoritmo realiza uma propagação das cores para as cenas seguintes, terminado o processo. Caso o usuário não ache o resultado satisfatório, poderá adicionar mais rabiscos a um determinado *frame*, o algoritmo então realizará a correção e propagará para os seguintes.

Em trabalho similar desenvolvido por Xia et al. (2016), dado um *frame* previamente colorido, o algoritmo aplica uma técnica de *Optical Flow* nas sequências de imagens para estimar os vetores de movimento para os demais *frames*. Com os vetores calculados, é aplicado o algoritmo proposto por Yatziv e Sapiro (2006).

Por sua vez, o trabalho de Paul et al. (2017) se aproveita das características tridimensionais de um vídeo, em que o terceiro eixo é o tempo. O mesmo utiliza técnicas

espaço-temporárias, aplicadas ao volume 3D do vídeo para propagar as cores. Com isso, os autores conseguiram eliminar o uso de *Optical Flow* a coloração inicial fica a cargo do usuário.

3

Fundamentação Teórica

Este capítulo traz conceitos e fundamentos dos vários métodos utilizados neste trabalho.

3.1 Imagens e Vídeos Digitais

Seja $E \subset \mathbb{Z} \times \mathbb{Z}$ um subconjunto finito retangular de pontos. Seja $K = [0, k]$ um conjunto totalmente ordenado. Representaremos por $Fun[E, K]$ o conjunto de todas as funções $f : E \to K$. Esse conjunto representa todas as imagens em tons de cinza.

O valor k denota o valor máximo de intensidade possível para essas imagens, enquanto K representa todos os níveis de cinza observáveis em uma imagem em tons de cinza $f \in Fun[E, K]$, do preto (0) ao branco (k). No caso de imagens binárias teríamos $k = 1$ e $K = \{0, 1\}$. Para imagens em tons de cinza, é comum utilizar imagens com o pixel ocupando o espaço de um *byte*, neste caso temos que $k = (2^8) - 1 = 255$, ou seja, temos 256 níveis distintos de cinza, indo do preto (0) até o branco (255), sendo $K = \{0, 1, ...255\}$ ou $K = [0, 256)$.

As formas de representação de imagens demonstradas até agora abordam apenas imagens em tons de cinza, ou seja, imagens que não possuem informação de cor. Os pixels dessas imagens se limitam a representar somente a luminância, ou intensidade, e não as cores. Para termos uma imagem colorida se faz necessária uma informação adicional: a crominância, que é a informação de cor. Vários modelos de espaço de cor são encontrados na literatura, tais como, RGB, CMY, HSV, Lab, YUV, YCbCr (Gonzalez e Woods, 2007). Em alguns deles é possível separar a informação de crominância e de luminância.

Tomando como referência o modelo **CIE L*a*b*** ou simplesmente Lab, que será utilizado neste trabalho, temos que o canal L representa a luminância, enquanto os canais a e b representam a informação de crominância da imagem A Figura 3.1 ilustra o modelo Lab.

Podemos definir a informação de crominância de uma imagem colorida no sistema Lab como $\overline{Chr} = \overline{a} \times \overline{b}$ onde $\overline{a} = \{-1, ..., 1\}$, $\overline{b} = \{-1, ..., 1\}$ sendo que os conjuntos $\overline{a}$ e $\overline{b}$ representam os valores possíveis dos canais a e b do modelo Lab. Logo, a função que representa unicamente as informações de crominância no sistema Lab para uma imagem é de acordo com a Equação 3.1.

$$f : E \rightarrow \overline{Chr} \tag{3.1}$$

Uma imagem colorida seria a junção da informação de luminância com a informação de crominância que, no nosso contexto, pode-se representar como a Equação 3.2.

$$f : E \rightarrow \overline{Chr} \times K \tag{3.2}$$

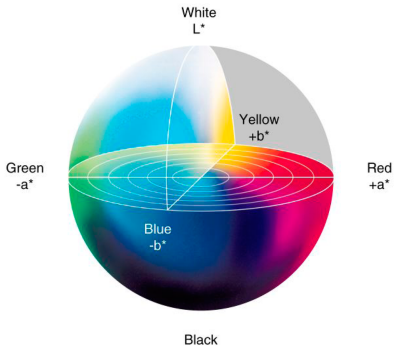


Figura 3.1: Modelo de cor Lab. Fonte: Wikimedia.

Observe que em relação as imagens em tons de cinza, o contradomínio da função apresentada na Equação 3.2 é maior. Note que, para um determinado k, uma imagem em tons de cinza apresenta um conjunto K de tonalidades e na Equação 3.2 temos o produto cartesiano de K por $\overline{Chr}$.

Para vídeos digitais, além das dimensões existentes em uma imagem é acrescida mais uma, que representa o tempo. Isso se dá pelo motivo que o vídeo é uma sequência de imagens apresentadas ao espectador com um intervalo de tempo entre elas. Por tanto, um vídeo, ou sequência de imagens em tons de cinza, pode ser representado por um conjunto de imagens conforme a Equação 3.3, sendo I o conjunto de todos os *frames* da sequência e I_i um *frame* de índice i da sequência .

$$I : \{I_1, I_2, ..., I_n\} : I_i \in Fun[E, K] \tag{3.3}$$

Para as informações de cores, representaremos conforme a Equação 3.4, sendo C o conjunto das informações de cor da imagem e C_i a informação de cor de um *frame* i.

$$C : \{C_1, C_2, ..., C_n\} : C_i \in Fun[E, \overline{Chr}] \tag{3.4}$$

Por fim, temos o conjunto das imagens coloridas, representado pela Equação 3.5.

$$O : \{O_1, O_2, ..., O_n\} : O_i \in Fun[\overline{Chr} \times K] \tag{3.5}$$

3.2 Descritores de textura

Apesar de ser intuitivo ao ser humano a percepção de texturas em imagens, para a computação o processo é mais complexo pois é puramente numérico. Para imagens em tons de cinza, Bharati et al. (2004) descrevem textura como sendo um atributo representando o arranjo espacial dos níveis de cinza dos *pixels* em uma região da imagem. Tais descritores podem ser utilizados para extrair características de forma a classificar parte ou toda uma imagem digital. Já Gonzalez e Woods (2007) descrevem a textura como um conjunto de características estáticas ou outras propriedades locais da imagem que sejam constantes, com pouca variação ou aproximadamente periódicas. Nas subseções seguintes abordaremos os descritores de texturas utilizados neste trabalho.

3.2.1 LBP

LBP ou (*Local Binary Pattern*) é um descritor de textura amplamente usado em diversos tipos de aplicações, sendo que o mesmo provou ser altamente discriminativo bem como possuir um baixo custo computacional comparado com outros descritores (Ahonen et al., 2006). O método visa encontrar um histograma de padrões binários locais para serem utilizados como descritores de textura. O LBP permite a definição do raio e do número de *pixels* vizinhos para se fazer o cálculo, possibilitando a operação em diversas escalas de textura. A Figura 3.2 demonstra a influência dos parâmetros de raio R e do número de vizinhos P.

Para cada *pixel* da imagem, o algoritmo compara a intensidade do *pixel* central com a dos vizinhos, dentro da vizinhança determinada pelos parâmetros R e P. De acordo com

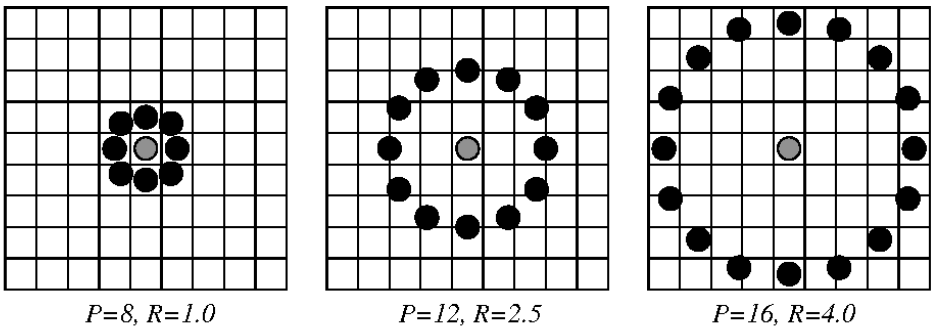


Figura 3.2: Influência dos parâmetros do LBP (Mäenpää, 2003)

Ojala et al. (2002), o algoritmo determina o valor 1, se a intensidade do *pixel* vizinho é igual ou superior a do central, e o valor 0 em caso contrário, formando uma matriz binária.

Com isto, o algoritmo então considera que cada *pixel* alcançado pelo raio R tenha uma numeração crescente, indo de 1 até P. Para cada *pixel* da vizinhança que o LBP encontrou o valor 1, ele efetua a soma do valor de 2 elevado à numeração designada para aquele pixel. A Figura 3.3 demonstra o cálculo. Feito isto para todos os *pixels* da imagem, é calculado o histograma do resultado. Este histograma é então utilizado como descritor para classificação e comparação.

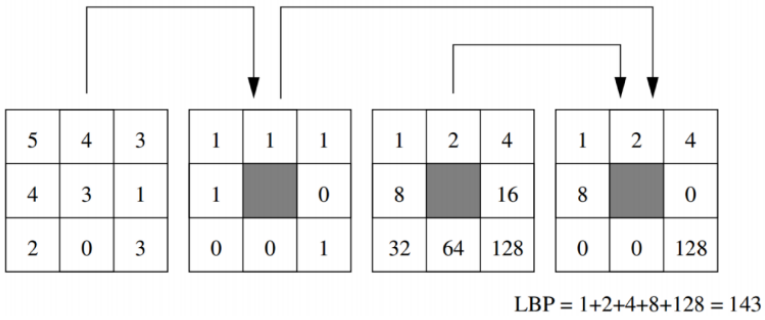


Figura 3.3: Exemplo de cálculo do LBP (Mäenpää, 2003)

As Equações 3.6 e 3.7 demonstram o cálculo do LBP, onde Ge é valor de intensidade do *pixel* central, Gp o valor de intensidade dos vizinhos, P o número de pontos e R o raio.

$$LBP_{P,R} = \sum_{p=1}^{P} 2^{(p-1)} \times X(Gp - Ge) \tag{3.6}$$

$$X(x) = \begin{cases} 1 & \text{if } x \geq 0, \\ 0 & \text{caso contrário.} \end{cases} \tag{3.7}$$

Em sua forma de cálculo padrão, o LBP com parâmetros P = 8 e R = 1 pode gerar um total de 256 valores distintos, o que exigiria um histograma de 256 posições. Entretanto, de acordo com Mäenpää (2003) é possível levar em consideração apenas os padrões uniformes, que se caracterizam por ter, no máximo, duas ocorrências do valor "01"ou "10"em sua representação binária, com isso o total possível de valores é reduzido para 59, o que possibilita criar um histograma de 59 posições.

3.2.2 Filtro de Gabor

O filtro de Gabor é um descritor de textura conhecido e útil no contexto de processamento de imagens e visão computacional. Ele possui a característica de conseguir extrair informações sobre textura tanto no domínio do espaço, quanto no da frequência de uma imagem (Yang et al., 2003). O filtro utilizado para imagens é uma evolução da sua versão original de uma dimensão, famoso, também, por ter sido inspirado no comportamento dos neurônios do córtex visual dos mamíferos (Daugman, 1985).

A extração de texturas por meio do filtro de Gabor se dá pela criação de um *kernel* ou filtro gaussiano. Esse filtro é então aplicado por meio de um processo de convolução na imagem e resultado é o descritor de textura.

A geração de um filtro bidimensional se dá pelas Equações 3.8 e 3.9. Onde B e C são fatores de normalização previamente definidos, f é a frequência desejada, θ e σ se referem à direção e intensidade.

$$G_c[i,j] = Be^{-\frac{(i^2+j^2)}{2\sigma^2}} \cos(2\pi f(i\cos\theta + j\sin\theta)) \tag{3.8}$$

$$G_s[i,j] = Ce^{-\frac{(i^2+j^2)}{2\sigma^2}} \sin(2\pi f(i\cos\theta + j\sin\theta)) \tag{3.9}$$

A Figura 3.4 demonstra um filtro de Gabor, a Figura 3.5 mostra o resultado da convolução do filtro da Figura 3.4 em uma imagem em tons de cinza.

Normalmente são utlizados vários filtros, com diversas orientações e intensidades, em uma mesma imagem gerando um banco de descritores. Isso é feito porque apenas um filtro pode não ser descritivo o suficiente.

3.3 Segmentação de Imagens

A segmentação de imagem consiste em particionar uma imagem R em sub-regiões, $R_1, R_2, R_3, ...R_n$, de tal modo que a intersecção de todas as regiões seja um conjunto

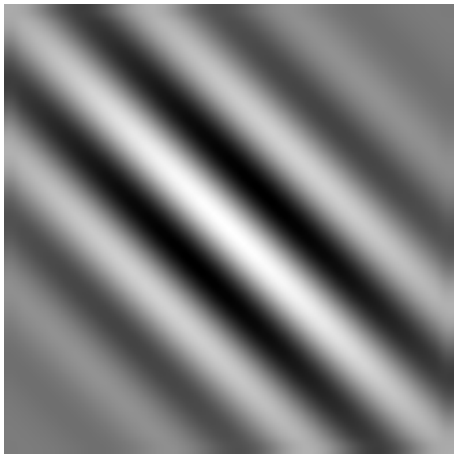


Figura 3.4: Filtro de Gabor orientação 2.35 e intensidade exp 2

Figura 3.5: Resultado da aplicação do filtro de Gabor em uma imagem.

vazio $\bigcap_i R_i = \emptyset$. Por outro lado a união de todas as regiões resulta na imagem original $\bigcup_i R_i = R$, ou seja, não existe *pixel* presente em mais de uma região, bem como todo *pixel* deve estar presente em uma região. Normalmente é utilizada para conseguir separar objetos de interesse do restante da imagem. Várias técnicas são utilizadas para realizar diversos tipos de segmentações, algumas delas são totalmente automáticas enquanto outras necessitam da assistência do usuário. Aqui apresentaremos alguns conceitos e métodos de segmentação que são relevantes para o presente trabalho.

3.3.1 Watershed

O algoritmo de *Watershed* é um algoritmo de segmentação morfológica de imagens, proposto por Digabel e Lantuéjoul (1977) e posteriormente melhorado por diversos autores (Beucher, 1979; Vincent e Soille, 1991). Nessa técnica, a imagem é interpretada como uma estrutura topológica, onde os *pixels* com maior valor, ou seja, com maior intensidade, são

considerados os pontos mais altos chamados de picos da região topográfica, enquanto os de menor valor são considerados os pontos mais baixos, chamado de vales, ou ponto mínimo, dessa região.

A partir disso o algoritmo simula como se a partir dos pontos de mínimo regional, a estrutura topográfica começasse a ser inundada, com a água subindo pelos mínimos regionais. Em determinado momento, quando as águas provenientes de dois mínimos regionais distintos forem se encontrar, barreiras são criadas para impedir que tais águas se misturem: essas barreiras representam as linhas de partição de água, ou Watershed (Vincent, 1992). A aplicação usual do Watershed é no gradiente morfológico da imagem, onde as bordas dos objetos estão realçadas.

Embora o algoritmo original seja totalmente automático, algumas variações do mesmo permitem que a segmentação seja assistida: um exemplo é o uso do Watershed com marcadores (Beucher e Meyer, 1992). Isso permite, por exemplo, que o usuário possa de forma iterativa segmentar as imagens conforme as regiões de interesse, até alcançar o particionamento desejado (Dougherty, 1992).

Como a segmentação por Watershed utiliza todos os mínimos regionais do gradiente da imagem como marcadores, acaba-se por ter uma quantidade excessiva de marcadores que, consequentemente, irá gerar um número muito elevado de regiões, o que resulta em um problema de super-segmentação. Tal supersegmentação é normalmente indesejada, pois gera regiões com poucos pixels sendo muitas vezes difícil extrair informações relevantes dessa região. Para amenizar esse problema diversas técnicas podem ser utilizadas, como a segmentação hierárquica.

3.3.2 Segmentação hierárquica

A segmentação hierárquica constitui um conjunto de segmentações de uma mesma imagem, com diferentes níveis de detalhes, no qual as segmentações com níveis mais grosseiros de segmentação são produzidas pela junção de regiões das segmentações de níveis mais refinado (Meyer, 2001).

A Figura 3.6 ilustra um processo de segmentação hierárquica. Observe que nos primeiros níveis há um particionamento mais grosseiro que pode ser obtido pela junção de regiões dos níveis posteriores mais refinados.

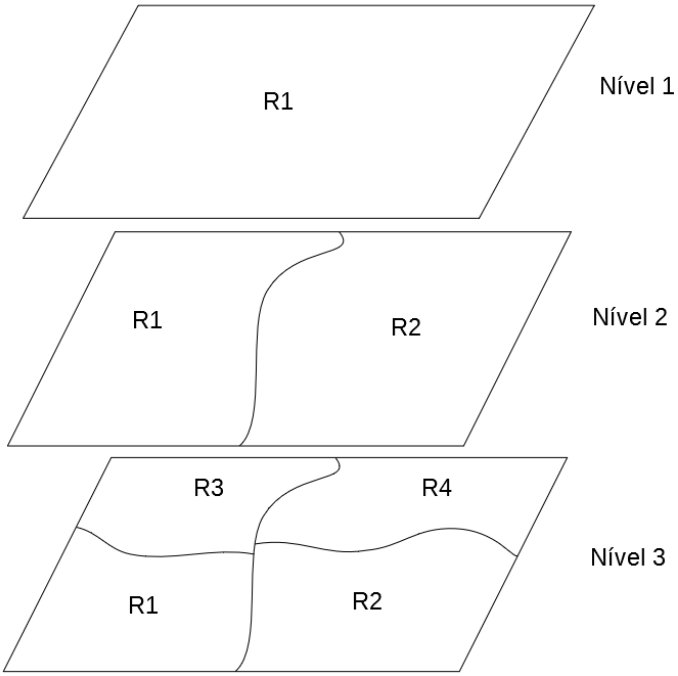


Figura 3.6: Segmentação hierárquica

O conceito de segmentação hierárquica pode também ser aplicado na segmentação por watershed.

Existem, na literatura, diversas formas de se extrair uma segmentação hierárquica por meio de watershed. Várias delas se baseiam no controle da quantidade de marcadores, selecionando apenas os que considera mais relevantes para o problema.

Uma das técnicas para se obter uma segmentação hierárquica por meio da segmentação de watershed é a utilização de valores de extinção. O objetivo desses valores de extinção é remover os mínimos regionais que não atendam a um certo critério.

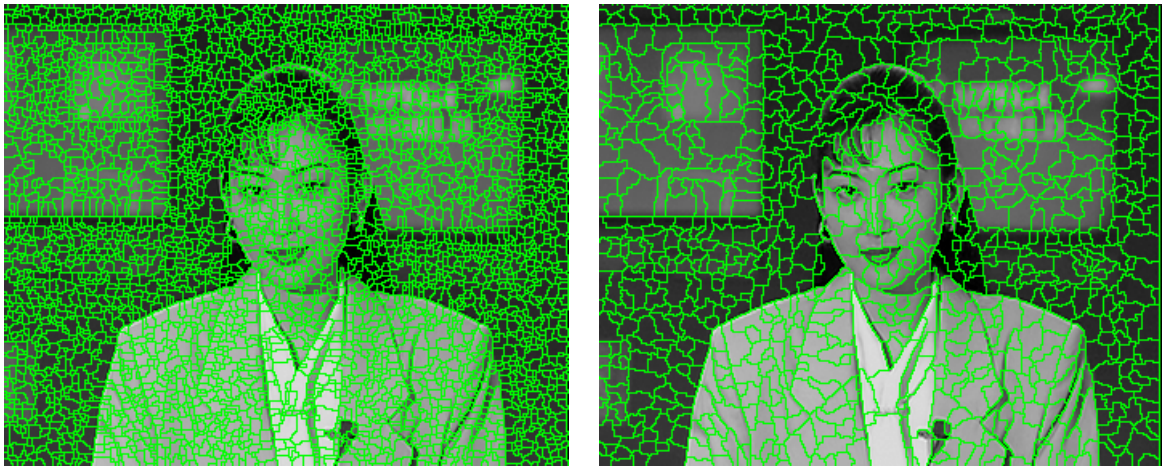
O valor de extinção de um mínimo regional, para qualquer atributo crescente, é o valor maximal de um filtro de atributo, tal que o mínimo ainda exista após a filtragem (Silva e de Alencar Lotufo, 2008).

Neste trabalho foi utilizado o valor de extinção por área (Vachier, 1995). Tal extinção consiste em eliminar os mínimos regionais cuja a área seja inferior a um determinado valor de extinção. Desta forma, é possível realizar um controle do refinamento da segmentação, variando o valor de extinção. Também é possível ter um controle sobre o número de regiões geradas. Uma vez calculado o valor de extinção para cada mínimo regional, basta selecionar os n mínimos regionais com maiores valor.

Além do critério de extinção por área, é possível também utilizar outros critérios para valores de extinção, como volume e altura. Entretanto, conforme demonstrado nos experimentos, o valor por área se mostrou ligeiramente melhor para o presente problema.

A implementação de filtro de extinção por área pode ser feita por meio da aplicação de sucessivas operações de fechamento no gradiente da imagem (Vachier, 1995), ou então, por meio da implementação de uma *max-tree* da imagem (Souza et al., 2015).

A Figura 3.7 ilustra a segmentação por Watershed na forma padrão e a segmentação usando valor de extinção por área. A Figura 3.7(a) apresenta a imagem supersegmentada. A Figura 3.7(b) apresenta a mesma segmentação com um refinamento menor, obtida pela escolha dos 950 mínimos regionais com maiores valores de extinção por área. As linhas de Watershed na Figura 3.7(b) são um subconjunto das linhas da Figura 3.7(a).



(a) (b)

Figura 3.7: Exemplo de imagem segmentada pelo Watershed padrão e segmentada por Watershed após o gradiente da imagem passar por um filtro de extinção por área.

3.3.3 Superpixel

Superpixel é um método também conhecido de segmentação de imagens. O método consiste em segmentar a imagem em grupos de *pixels* com significado semelhante, de forma a diminuir a redundância da imagem. Portanto, cada superpixel deve ter uma aparência natural, sem se sobrepor a múltiplos objetos e logo cada objeto da imagem pode estar associado a um ou mais superpixels (Li e Chen, 2015). Desta forma, temos

como resultado final uma imagem supersegmentada onde cada superpixel pertence a um único objeto e cada objeto possui um ou mais superpixel.

O número desejado de superpixels é normalmente fornecido pelo usuário de forma a controlar o nível de supersegmentação. A técnica consiste em um pré-processamento da imagem, de forma a trabalhar com uma quantidade menor de objetos do que se trabalharia com cada *pixel* individualmente.

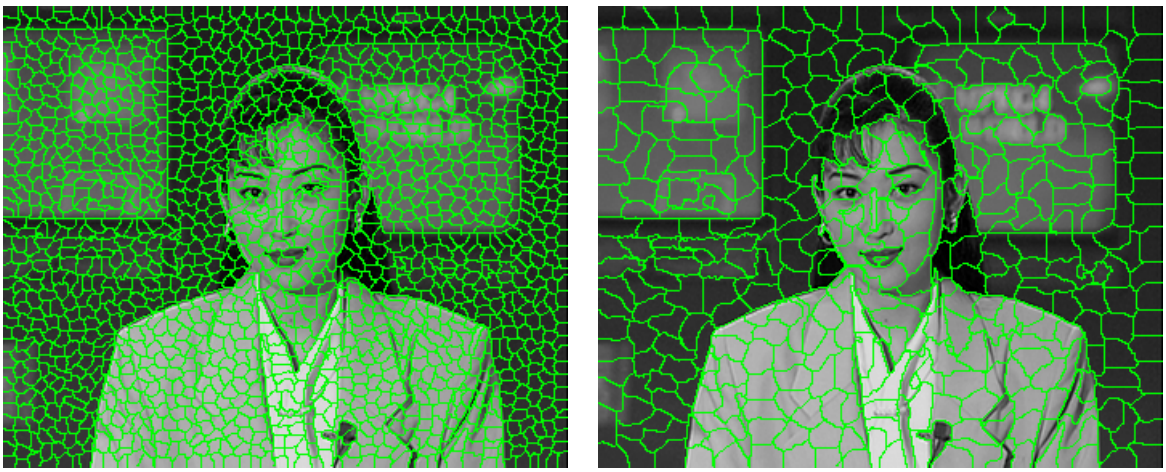
Para a geração desses resultados vários métodos são propostos na literatura, cada um com a sua vantagem e desvantagem indo desde tempo de execução até a quantidade de parâmetros fornecido pelo usuário (Achanta et al., 2012).

Os algoritmos atuais de geração de superpixel podem ser divididos, em sua maioria, em dois grupos: os baseados em Grafos e os baseados em métodos de Gradiente Ascendente. Os modelos baseados em Grafos tratam a imagem como um Grafo, no qual cada *pixel* é um vértice e as arestas interligam os *pixels* vizinhos tendo como peso um certo grau de similaridade entre os dois. Os métodos baseados em Gradiente Ascendente operam a partir de um certo particionamento e iterativamente refinam cada grupo até um certo critério de parada seja encontrado (Achanta et al., 2012). No presente trabalho foi utilizada uma implementação baseada em Gradiente Ascendente, denominada Seeds: superpixel, proposta por Van den Bergh et al. (2012). A escolha foi devido ao seu bom desempenho em comparação com outros métodos da literatura.

A Figura 3.8 demonstra a aplicação do algoritmo de superpixel em uma imagem. Na Figura 3.8(a), o Algoritmo aplicado com parâmetro de 3000 regiões; na Figura 3.8(b), o algoritmo foi aplicado na mesma imagem, porém com parâmetro de 1000 regiões.

3.4 Colorização por Otimização

Um dos trabalhos conhecidos na literatura de colorização de imagens é o de Levin et al. (2004). Os autores propuseram uma nova abordagem para colorização de imagens, na qual tratam o problema de colorização como um problema de otimização. Assume-se que *pixels* vizinhos com intensidade similares devem possuir a mesma informação de cor, desta forma o trabalho, do algoritmo é propagar as cores dos *pixel*, que já possuem informação de cor para os demais seguindo uma restrição imposta. O algoritmo trabalha utilizando o espaço de cor YUV, no qual Y representa a luminância da imagem e U e V as informações de cor. Dado um volume de entrada Y(x,y,t) a saída do algoritmo são dois volumes *U(x,y,t)* e *V(x,y,t)* que representam os canais de crominância. Para simplificação o trio (x,y,t) será tratado como as letras **p,q**, sendo Y(**p**) a luminância de um determinado *pixel*.



(a) (b)

Figura 3.8: Exemplo de segmentação por superpixel.

Logo, o objetivo é minimizar a diferença de cor U($\mathbf{p}$) no *pixel* $\mathbf{p}$ com a média ponderada das cores dos *pixels* vizinhos, seguindo a Equação 3.10.

$$J(U) = \sum_{\mathbf{p}} \left(U(\mathbf{p}) - \sum_{\mathbf{q} \in N(\mathbf{p})} w_{\mathbf{pq}} U(\mathbf{q}) \right)^2 \tag{3.10}$$

Para poder propagar as cores alguma informação de cor deve ser fornecida ao algoritmo. Para isto, o usuário realiza rabiscos coloridos na imagem, nos lugares onde deseja aquela determinada cor. O lugar marcado pelo rabisco do usuário permanecerá com a cor informada enquanto o restante da imagem será colorida seguindo a restrição.

4

Método Proposto

Neste capítulo, é apresentado o método proposto para colorização de sequências de imagens em escala de cinza, por meio do uso de descritores de textura.

4.1 Visão Geral

O método proposto tem como seu principal objetivo realizar a colorização de um vídeo, ou uma sequência de imagens, em tons de cinza de forma assistida. O objetivo é que o usuário consiga realizar a colorização de modo que a tarefa seja executada em um tempo consideravelmente menor e com redução do número de interferências do usuário no processo. Para isso, em um primeiro momento, o usuário deverá realizar a colorização de um primeiro *frame* da sequência para se ter uma referência inicial. Propõe-se o uso de uma ferramenta que auxilie o usuário nesta primeira tarefa, de tal modo que, por meio de um algoritmo de segmentação assistida, será realizada a segmentação do *frame* em regiões de interesse e para cada região será definida uma cor escolhida livremente pelo usuário. Além disso, também será possível utilizar uma ferramenta no estilo de "pincel", de forma que possa ser feito um refinamento da colorização manual por meio de riscos na imagem.

Tendo esse primeiro *frame* colorido manualmente, o método utiliza a imagem segmentada e colorida como referência para realizar de forma automática a colorização do *frame* seguinte. Para isso, são extraídas informações de textura e de intensidade de ambos os *frames*, essas informações são agrupadas por regiões, definidas por um algoritmo de segmentação, o que resulta em um vetor de características para cada região de cada *frame*. Para cada região do *frame* atual, denominado como o *frame* a ser colorido, são comparados

os vetores de características deste com os das regiões do *frame* de referência, que é o *frame* anterior, a fim de encontrar a melhor correspondência.

Como resultado, cada região da imagem terá uma cor associada, proveniente do *frame* anterior. Para isso, utiliza-se um algoritmo específico de colorização, descrito no trabalho de Levin et al. (2004) para realizar a distribuição das cores gerando o *frame* colorido. Neste momento, da-se ao usuário a chance de se realizar uma conferência do resultado, no qual o mesmo poderá alterar as cores designadas pelo método, ou adicionar novas cores para eventuais novos objetos que possam surgir. Após estar satisfeito com o resultado, o usuário dá início ao processo automático que iniciará a colorização do *frame* subsequente. Tal processo se repete até a conclusão da sequência de imagens.

Desta forma, o método proposto pode ser dividido em seis principais estágios, sendo eles:

1. Segmentação e colorização manual do primeiro *frame*, realizada pelo usuário;
2. Segmentação automática de todos os *frames* da sequência;
3. Extração de informações de textura e luminância do *frame* atual, a ser colorido e do *frame* anterior, caso ainda não tenham sido extraído;
4. Comparação das informações de textura e luminância extraídas do *frame* atual e anterior para encontrar a melhor correspondência;
5. Transferência das cores do *frame* anterior para o atual de acordo com as correspondências encontradas;
6. Conferência e intervenção do usuário.

Detalhes de cada um desses estágios serão descritos a seguir.

4.2 Algoritmo

Um exemplo de implementação do método proposto é apresentado no Algoritmo 1, que tem como objetivo facilitar a compreensão do método.

Conceitos preliminares

Para compreender o funcionamento do algoritmo reforçamos alguns conceitos já descritos no Capítulo anterior do trabalho bem como também se faz necessária a introdução de conceitos adicionais.

Seja $I : \{I_1, I_2, ...I_n\} : I_i \in Fun[E, K]$ a sequência original, em escala de cinza, que se deseja colorir. Seja $C : \{C_1, C_2, ...C_n\} : C_i \in Fun[E, \overline{Chr}]$ o conjunto de informações de crominância da sequência, sendo $C_i : E \rightarrow \overline{a} \times \overline{b}$ a informação de crominância para o *frame* i da sequência. Seja $O : \{O_1, O_2, ...O_n\} : O_i \rightarrow \overline{Chr} \times K$ a sequência final totalmente colorida, onde O_i é o frame i já colorido.

Tomamos como $S : \{S_1, S_2, ..., S_n\}$ a sequência I segmentada por meio de um algoritmo de segmentação hierárquica automática, como Watershed ou superpixel, onde $\mathbf{r}$ representa uma das regiões de S_i.

Seja $U(i, \mathbf{r}, rad)$ o conjunto de todas as regiões de S_{i-1} que possuem pontos dentro de um raio rad predeterminado a partir da borda da região $\mathbf{r} \in S_i$, sendo estas regiões denominadas regiões candidatas de $\mathbf{r}$.

Seja $FD_{\mathbf{r}}(\mathbf{t})$ (veja Equação 4.9) uma função que calcula a soma ponderada das distâncias entre os vetores de características das regiões $\mathbf{r}$ e $\mathbf{t}$. Seja $MD_i(\mathbf{r})$ a função que retornará a melhor correspondência entre a região $\mathbf{r} \in S_i$ e os elementos do conjunto $U(i, \mathbf{r}, rad)$, dada por,

$$MD_i(\mathbf{r}) = \mathbf{t} \in S_{i-1} : FD_{\mathbf{r}}(\mathbf{t}) = \min_j \{FD_{\mathbf{r}}(t_j) : t_j \in U(i, \mathbf{r}, rad)\} \tag{4.1}$$

Seja $Segment(I)$ a função que realiza a segmentação automática para cada I_i gerando o conjunto S. Seja $Colorize(I_i, C_i)$ uma função em que dado um *frame* I_i e uma informação de crominância C_i, a mesma retorna um *frame* que foi colorido através de um algoritmo de otimização.

4.3 Segmentação

Como citado anteriormente, o agrupamento das características da imagem, bem como a realização da colorização são feitos baseados em regiões da imagem. Para gerar essas regiões se faz necessária a utilização de um algoritmo automático de segmentação. Antes do processo iniciar, fornecemos ao algoritmo de segmentação o conjunto de imagens em escala de cinza I para que o mesmo forneça a versão segmentada S. As regiões geradas pelo algoritmo devem ser capazes de satisfatoriamente separar os objetos da imagem, podendo haver mais de uma região para o mesmo objeto, evitando que uma região acabe por pertencer a dois objetos de interesse, ao mesmo tempo na imagem. Por outro lado, é importante destacar que uma quantidade excessiva de regiões na imagem pode causar outros efeitos colaterais, como um volume alto de processamento e prejudicar a qualidade descritiva dos atributos extraídos.

Algoritmo 1 Algoritmo para colorização.

Entrada - Sequência em escala de cinza I
Saída - Sequência colorida O

$S \leftarrow segment(I)$
for all segment $\mathbf{r} \in S_1$ **do**
  //Usuário efetua a colorização manual de S_1
  defina crominância (a,b) para a região $\mathbf{r} : a \in \overline{a}, b \in \overline{b}$;
  $C_1(x) \leftarrow (a, b), \forall x \in \mathbf{r}$;
end for
for i in $[2..n]$ **do**
  for all regiões $\mathbf{r}$ in S_i **do**
    $\mathbf{t} \leftarrow MD_i(\mathbf{r})$;
    $C_i(x) \leftarrow C_{i-1}(y), \forall x \in \mathbf{r}, y \in \mathbf{t}$;
  end for
  //Usuário interage com resultado em C_i
  $O_i \leftarrow Colorize(I_i, C_i)$;
end for

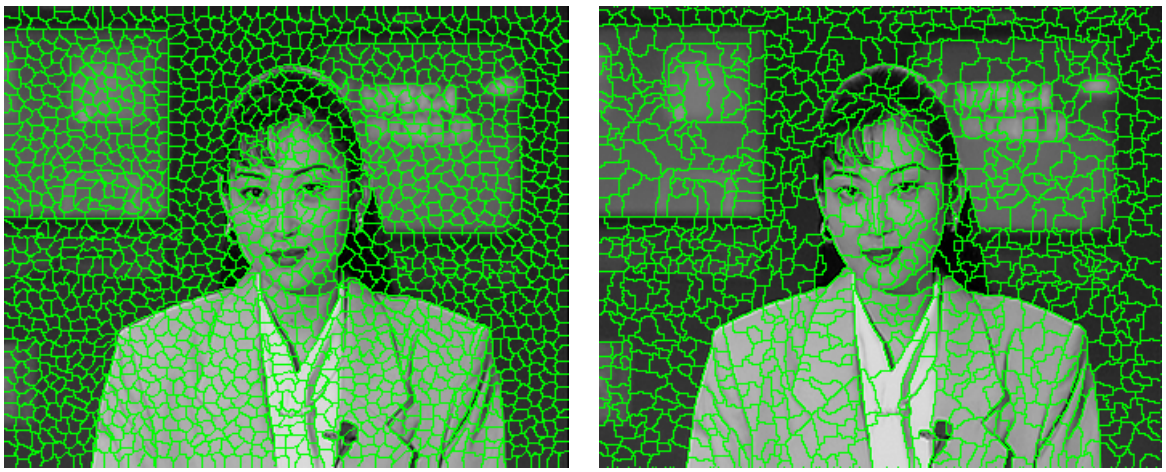
Portanto, é necessário equacionar a quantidade de regiões geradas pelo algoritmo a fim de se conseguir bons descritores, bem como a capacidade de se isolar com exito objetos da imagem. Para este trabalho, demonstramos a utilização com dois algoritmos de segmentação distintos sendo eles o algoritmo de geração de superpixel (Li e Chen, 2015), também denominado algoritmo de agrupamento por superpixel, ou simplesmente representação por superpixel e o algoritmo de Watershed (Vincent e Soille, 1991) usando valor de extinção por área.

O algoritmo de superpixel, por ter como base uma técnica de agrupamento, fornece a facilidade de se controlar a quantidade de regiões na imagem, bem como possue pouca variação de tamanho entre as regiões. O *Watershed*, por sua vez, demonstrou ter uma capacidade melhor de isolar objetos da imagem. Detalhes do desempenho de ambos para o presente método serão relatados no Capítulo 5. A Figura 4.1 mostra um exemplo dos resultados dos algoritmos de segmentação citados.

Importante salientar que vários outros algoritmos de segmentação poderiam ser utilizados.

4.4 Colorização Manual do Primeiro Quadro

Como o objetivo do método é realizar a colorização de uma sequência de imagem em tons de cinza, parte-se do princípio que não há informação de cor prévia disponível. Assim, para ser possível que o método realize a sua parte automática de colorização é necessária uma



(a) (b)

Figura 4.1: Exemplos dos algoritmos de segmentação utilizados: (a) superpixel; (b) *Watershed* com extinção por área.

primeira informação de cor. Tal informação é provida pelo usuário através de um processo de colorização manual de um primeiro *frame*, normalmente o primeiro da sequência.

O processo puramente manual de colorização, mesmo sendo de apenas uma imagem, é demorado e exige muito esforço do usuário. Para auxiliá-lo neste processo, utiliza-se uma ferramenta, que faz uso de algoritmos de segmentação e outros meios para facilitar esse processo manual. Para essa etapa é utilizado uma implementação do algoritmo de Watershed utilizando marcadores fornecidos pelo usuário (Vincent, 1992; Vincent e Soille, 1991).

O usuário adiciona livremente marcadores na imagem, que são representados como riscos. Conforme cada marcador é inserido, em tempo real, é calculado o gradiente da imagem e executado o algoritmo de Watershed utilizando os marcadores fornecidos e, então, são exibidos para o usuário as linhas de segmentação demonstrando as regiões geradas. O usuário pode, em seguida, adicionar e remover quantos marcadores forem necessários até estar satisfeito com a segmentação resultante. Após esse processo, basta selecionar a região e definir a cor desejada, repetindo o processo para todas as regiões.

Além disso, uma ferramenta livre, no estilo das ferramentas de Pincel existentes em aplicativos de edição de imagem, é disponibilizada para que o usuário possa fazer pequenos retoques onde eventualmente a segmentação não tenha sido precisa o suficiente. A Figura 4.2 demonstra as ferramentas propostas para a realização da colorização manual.

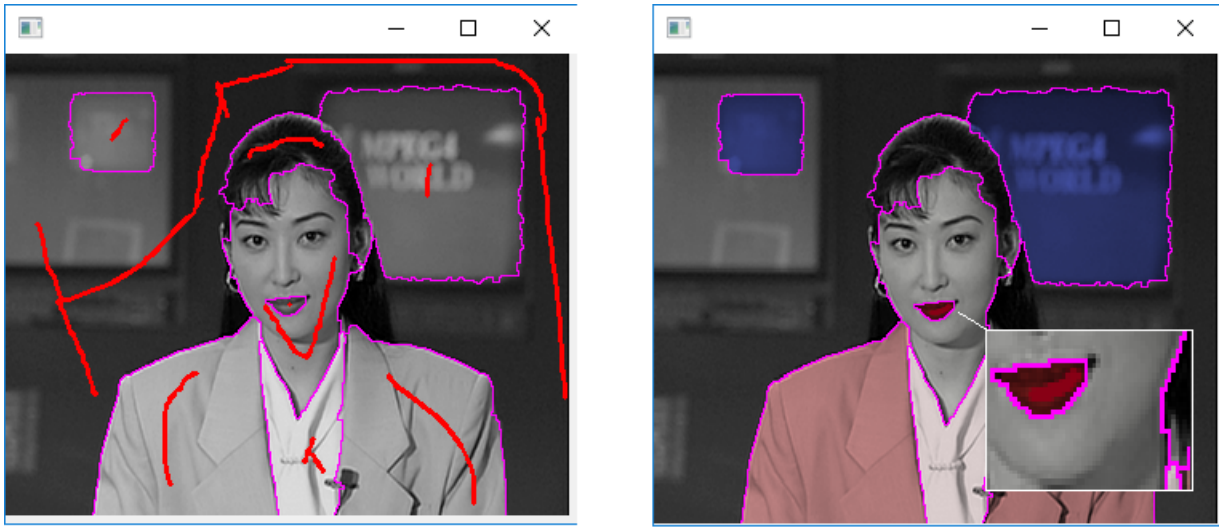


Figura 4.2: Ferramenta para colorização manual, à esquerda a segmentação usando marcadores (em vermelho). À direita a ferramenta de pincel para retocar pequenas áreas.

4.5 Extração de Características

Uma das etapas mais importantes do método aqui apresentado é a extração de descritores de textura e informações de intensidade, pois é por meio dessas informações que será definida a designação das cores para o *frame* atual. Para essa etapa, foram estudados vários descritores de textura e características na literatura a fim de encontrar os mais promissores para o presente problema. Após uma análise criteriosa de suas aplicações, resultados em outros problemas, inclusive de colorização, bem como em problemas de classificação envolvendo imagens, se optou pela utilização de dois descritores de textura, sendo um os Filtros de Gabor, e o outro o Local Binary Pattern (LBP), ambos já introduzidos na fundamentação teórica do presente trabalho. Também optou-se pela utilização das informações de intensidade como mais uma característica.

Para cada região de ambos os *frames* se gera um vetor de características de 101 dimensões, composto dos três tipos de características citadas, sendo um subvetor de 40 dimensões para o Descritor de Gabor, um subvetor de 59 dimensões composto do histograma do LBP e por fim um subvetor de 2 dimensões para as informações de intensidades. Tais subvetores são extraídos e calculados conforme demonstrado a seguir.

4.5.1 Intensidade

Como a proposta é transferir as cores entre imagens semelhantes (dois *frames* consecutivos) informações de intensidade se tornam muito relevantes para discriminar regiões da imagem (Bugeau et al., 2014). Extrair, porém, informações somente da intensidade da região podem não ser discriminativas o suficiente. Por esse motivo, extrai-se informação de intensidade da região bem como das intensidades das regiões vizinhas, o que gera o vetor de duas dimensões para intensidade.

A primeira posição do vetor corresponde à média da intensidade dos pontos dentro da própria região. Assim, para um *frame* i, seja n o número de pontos dentro da região $\mathbf{r} \in I_i(x)$ e $I_i(x)$ o valor de intensidade do pixel x, a primeira dimensão é calculada conforme a Equação 4.2.

$$FI(\mathbf{r}) = \frac{1}{n} \sum_{x \in \mathbf{r}} I_i(x) \tag{4.2}$$

Para a segunda dimensão, consideramos M como o número de regiões vizinhas de $\mathbf{r}$ e $\mathbf{r}_k \in N(\mathbf{r})$ uma região vizinha de $\mathbf{r}$, sendo, por tanto, $N(\mathbf{r})$ o conjunto de regiões vizinhas de $\mathbf{r}$, com isso, a segunda dimensão deste vetor de intensidade é calculada conforme a Equação 4.3.

$$FN(\mathbf{r}) = \frac{1}{M} \sum_{\mathbf{r}_k \in \ N(\mathbf{r})} FI_i(\mathbf{r}_k) \tag{4.3}$$

4.5.2 Filtros de Gabor

Como já citado, os Filtros de Gabor são bem populares na área de processamento de imagem, sendo majoritariamente utilizados para extração de descritores de textura (Manjunath e Ma, 1996). A extração ocorre pela da aplicação dos filtros sobre a imagem por meio do processo de convolução. O resultado desse processo são os descritores de Gabor, sendo comum a extração de diversos desses descritores para se gerar um banco de descritores.

No presente trabalho, é utilizado um total de 40 filtros diferentes, variando o valor das orientações e das escalas dos filtros. Foram utilizados 8 tipos de orientações, variando de 0 até $7\pi/8$ bem como 5 escalas exponenciais diferentes $exp(i \times \pi), i = \{0, 1, 2, 3, 4\}$. Seja G_d o resultado da aplicação de um dos Filtros de Gabor descritos. Seja $G = \{G_1, G_2, G_3, \ldots G_{40}\}$ o conjunto dos resultados das aplicações dos 40 filtros.

Seja $\mathbf{r} \in S_i$. Seja G_i um dos Filtros de Gabor supracitados.O resultado médio do filtro G_i para a região $\mathbf{r}$ é dado por:

$$FG(\mathbf{r}, G_d) = \frac{1}{n} \sum_{x \in \mathbf{r}} G_d(x) \tag{4.4}$$

O descritor do Filtro de Gabor para uma região $\mathbf{r} \in S_i$ é dado pelo vetor de características:

$$V = \{FG(\mathbf{r}, G_i), i = 1, \ldots, 40\} = \{FG(\mathbf{r}, G_1), FG(\mathbf{r}, G_2), FG(\mathbf{r}, G_3), \ldots FG(\mathbf{r}, G_{40})\}. \tag{4.5}$$

4.5.3 LBP

No presente trabalho, aplicamos o algoritmo do LBP no *frame* utilizando um raio de 1 e 8 como o número total de pontos vizinhos e levamos em consideração apenas os padrões uniformes. Como o resultado do cálculo de LBP gera um vetor da mesma dimensão da imagem, calcula-se um histograma levando em consideração os pontos pertencentes à região que se deseja obter o descritor. Tal histograma é então utilizado como descritor de textura da região.

A Figura 4.3 demonstra a geração do LBP para uma região. Na parte superior, tomando como referência o LBP de uma imagem com dimensões 15 por 15, onde a área sombreada corresponde à região que se deseja extrair o descritor, calcula-se o histograma somente dos pontos dentro dessa região, mostrado na parte inferior da imagem. Tal histograma servirá como descritor de textura.

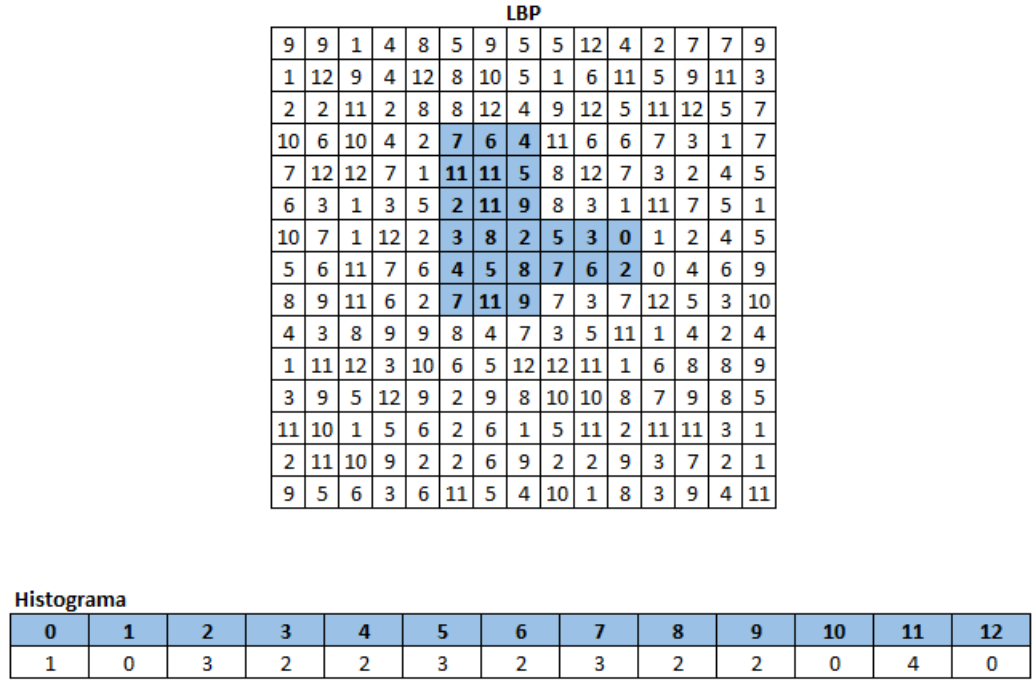


Figura 4.3: Demonstração do cálculo do LBP para uma região.

Uma vez que o padrão uniforme pode gerar até $N(N-1)+3$ (Mäenpää, 2003) possíveis valores, onde N representa o total de pontos vizinhos e estamos utilizando 8 pontos vizinhos, temos um total de 59 possíveis valores. Como está sendo utilizado o histograma como descritor, o nosso vetor para o LBP deve ter, por tanto, 59 posições.

4.5.4 Comparação de Características

Nesta seção é tratada a forma de comparação das características, ou seja, as informações de textura e luminância, de forma encontrar a melhor correspondência entre duas regiões. Primeiro, será tratada a área de busca, que se entende como a forma de se escolher as regiões candidatas, do *frame* de referência, para uma região do *frame* atual. Na sequência será tratada a forma que se é feita a comparação para determinar o grau de similaridade entre duas regiões.

Área de Busca

Para determinar a cor a ser atribuída a uma região do *frame* atual, precisamos encontrar a região mais similar dentre as do *frame* de referência. Intuitivamente, pode-se realizar uma busca exaustiva de forma a realizar uma comparação de todas as regiões de um *frame* com todos do *frame* de referência. Apesar de aparentar ser eficaz, essa abordagem traz alguns problemas para o presente caso.

Primeiramente, isso geraria um problema de solução em tempo exponencial. Apenas exemplificando, imaginando dois *frames* com 2.000 regiões cada, seria necessário se realizar um total de 4 milhões de comparações. Como veremos adiante, cada comparação demanda diversos cálculos de distância e ponderação para se chegar ao valor de similaridade. Logo, mesmo com a capacidade computacional que possuímos atualmente, isso ainda gerará um processo relativamente lento. Como se deseja tornar o processo o mais rápido possível para o usuário, é interessante reduzir essa quantidade de comparações.

Segundo, uma comparação completa pode aumentar a possibilidade de erros na designação de cores. Isto ocorre, pois como há uma comparação, uma região pode encontrar como melhor correspondência um *frame* distante espacialmente, no *frame* de referência. A título de exemplo, consideramos que uma região, a qual se deseja atribuir uma cor, fique posicionada no canto superior direito e o mesmo acaba encontrando como sua melhor correspondente, no *frame* de referência, uma região que se localiza no canto inferior esquerdo. Como estamos lidando com dois *frames* sequenciais, espera-se que a mudança dos objetos seja gradual, por tanto, a possibilidade de a região efetivamente representar o objeto que se está rastreando é baixa.

Para solucionar os problemas relatados nos parágrafos anteriores, partimos do princípio que as alterações espaciais dos objetos da sequência, entre dois *frames*, são baixas. Com isso, podemos assumir que a região que desejamos encontrar está dentro de uma área limitada, dentro de um certo raio da área da região atual.

Tomamos como $\mathbf{r} \in S_i$ a região que se deseja encontrar a cor e como $U(i, \mathbf{r}, rad)$ uma função que irá retorna as regiões candidatas de S_{i-1} para serem comparadas com $\mathbf{r}$. As regiões candidatas $U(i, \mathbf{r}, rad)$ são encontradas selecionando somente as regiões de S_{i-1} que tenham pixel dentro de uma área formada pelo raio rad a partir da borda da região atual. Calcula-se a dilatação morfológica de $\mathbf{r}$ por um elemento estruturante $Rd(rad)$ que possui o formato de disco de raio rad. O resultado dessa dilatação denominaremos de $SA(\mathbf{r}, rad)$.

A máscara binária resultante da dilatação é usada no *frame* de referência. Cada região do *frame* S_{i-1} que tenha pontos dentro da área delimitada pela máscara fará parte do conjunto de regiões candidatas.

A área de busca relativa à região $\mathbf{r} \in S_i$ é dada por $U(i, \mathbf{r}, rad) = \{\mathbf{r}_j \in S_{i-1} : \exists x \in \mathbf{r}_j : x \in SA(\mathbf{r}, rad)\}$,

A Figura 4.4 ilustra o processo da seleção das regiões candidatas conforme a área de busca. A Figura 4.4(a) demonstra a região a ser processado (em vermelho) do *frame* a ser colorido; na Figura 4.4(b) a área de busca criada após o processo de dilatação morfológica da região; a Figura 4.4(c) demonstra essa mesma área, porém no *frame* anterior, que já foi colorido; por fim, a Figura 4.4(d) demonstra as regiões do *frame* anterior que serão utilizadas na comparação com a região da Figura 4.4(a). Todas as regiões que possuem pontos na área de busca são selecionados.

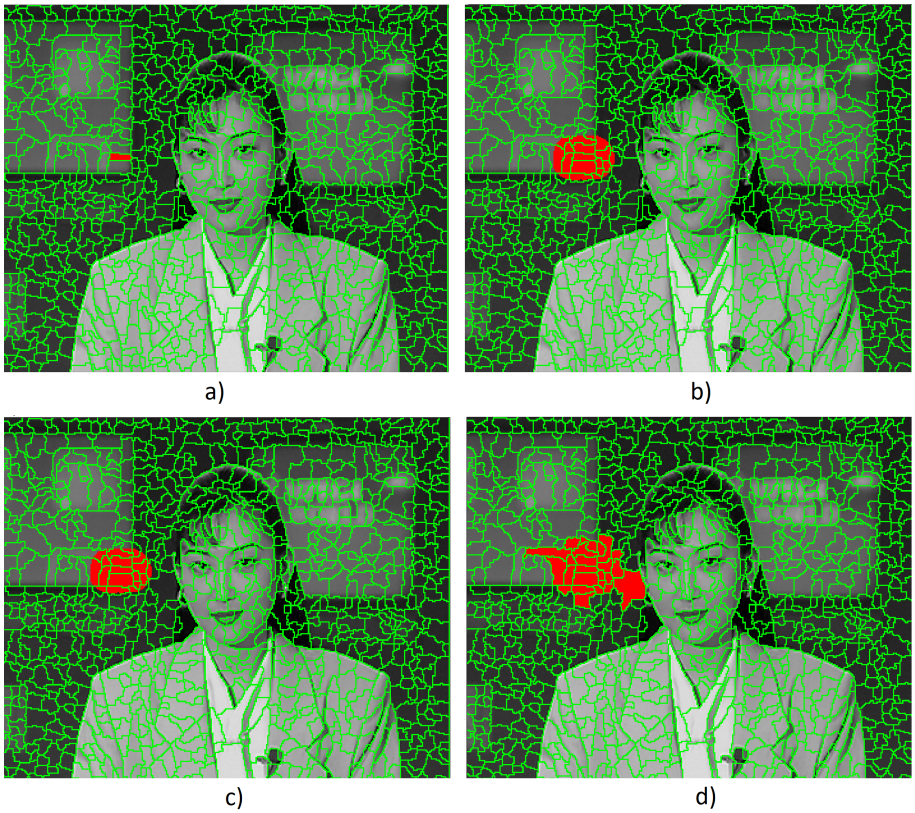


Figura 4.4: Demonstração do processo de seleção da área de busca para uma região.

Com esse procedimento, consegue-se reduzir drasticamente a quantidade de comparações necessárias para se encontrar um bom resultado, pois fica-se restrito a apenas uma região, bem como consegue-se reduzir o volume de erros de designação de cores. Para verificar a eficácia desse método de limitação da área de busca, alguns experimentos

foram realizados, verificando o tempo de processamento com e sem a limitação da área de busca, bem como foi avaliado o erro total comparado com um Padrão Ouro, no qual foi manualmente selecionada a cor correta de cada região. O experimento foi feito usando os 150 primeiros *frames* da sequência Akiyo e um raio de 70 pontos. As metodologias do cálculo de erro e ambientação destes testes serão detalhadas no Capítulo 5.

A Figura 4.5 mostra um gráfico da razão entre o tempo de comparação sem a limitação de área e com a limitação, demonstrando que, em média, a aplicação de limitação faz a comparação ficar 33 vezes mais rápida. Já a Figura 4.6 mostra os erros de designação de cor com e sem a área de busca, como pode-se verificar, em todos os casos o percentual de erros foi menor ao utilizar a limitação, isso se deve devido ao fato de se evitar comparações com regiões distantes.

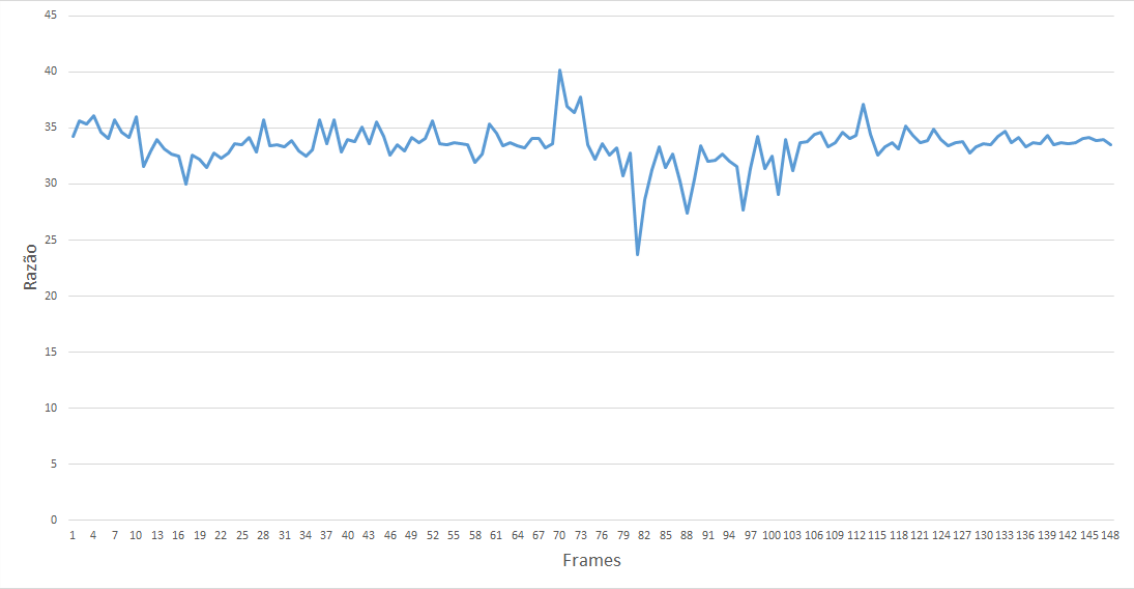


Figura 4.5: Razão entre o tempo de comparação sem a aplicação da limitação e com a aplicação da limitação.

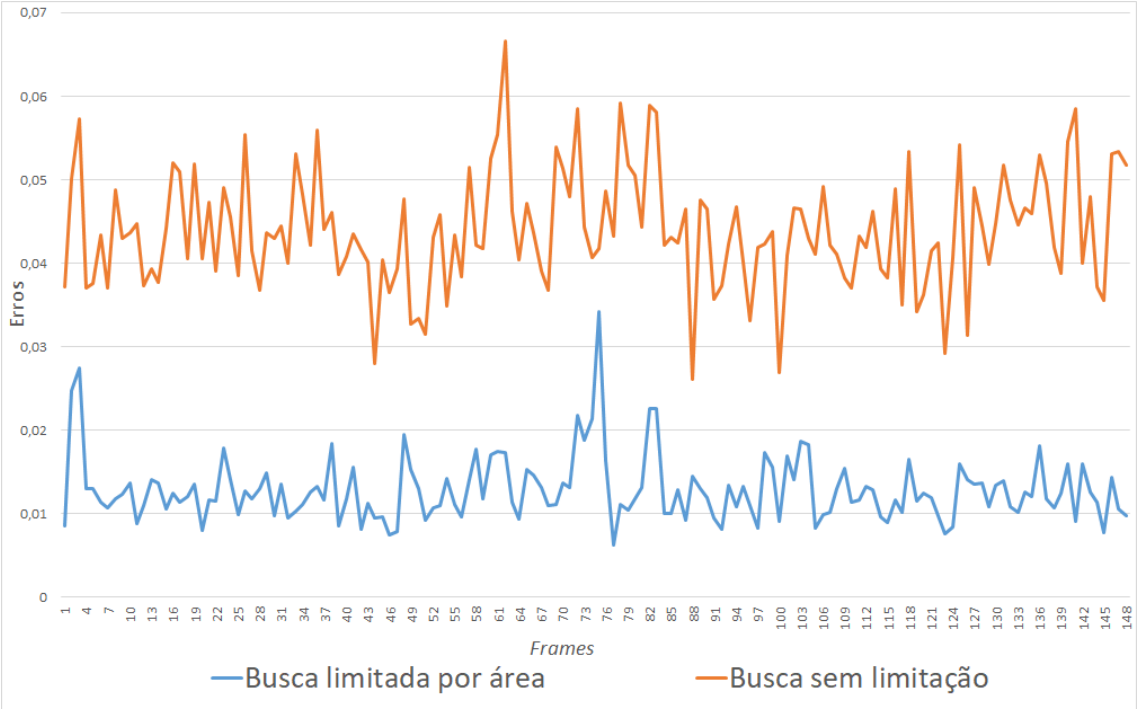


Figura 4.6: Percentual de erros de designação de cor com e sem a limitação da área de busca.

Importante ressaltar a influência do valor do raio de busca nos resultados e desempenho. Um valor alto irá impactar no desempenho e também poderá aumentar os erros, em contrapartida, um valor muito baixo limita a generalização levando a possíveis resultados viciados que poderá afetar o resultado final. Assim, na hora de seleção desse valor vários pontos devem ser levados em conta, mas principalmente a taxa de quadros da sequência. Uma sequência com uma taxa de quadros baixa tende a ter mais mudanças entre um *frame* e outro do que uma sequência de taxa de quadros alta.

Comparação

Na subseção anterior foi tratado o problema da seleção das regiões candidatas as quais serão comparadas com a região atual. Essa comparação visa encontrar, dentre as regiões dentro da área de busca, a região de maior similaridade com a atual para então transferir a informação de cor. Neste trabalho, a similaridade é calculada de acordo com as distâncias entre os vetores de características de cada região. A forma de cálculo de distância usada foi a denominada *Cityblock* (Perlibakas, 2004), uma vez que a comparação é feita entre vetores unidimensionais, essa medida se mostra suficiente para medir a similaridade, bem como apresenta um cálculo mais simples que demanda menos tempo de processamento.

A Equação 4.6 demonstra como é realizado tal cálculo, onde a e b, neste contexto, representam dois vetores de características, com dimensão n.

$$D(a,b) = \sum_{j=1}^{n} |a_j - b_j| \tag{4.6}$$

Como citado anteriormente, neste trabalho foi utilizado três tipos diferentes de características, entretanto cada uma delas carrega informações com relevância diferentes. Um descritor de textura, por exemplo, para o problema atual tende a ter uma relevância maior do que a informação de intensidade. Além disso, cada tipo de vetor de características possui faixa de valores diferentes, logo se faz necessário reescalar tais valores.

Desta forma, deve-se reescalar e ponderar os vetores de características dando pesos para cada um de acordo com a sua relevância para o problema. Com relação ao valor dos pesos, no Capitulo 5 foram realizados vários testes que apontam alguns valores com melhor desempenho.

Seja $\mathbf{r}$ a região atual a ser colorizado, e $\mathbf{u}$ uma das regiões candidatas provenientes de S_{i-1}. O objetivo é encontrar a região $\mathbf{u}$ que apresente o menor valor $FD_{\mathbf{r}}(\mathbf{u})$, sendo que este último representa a soma ponderada de todas as distâncias entre os vetores de características de $\mathbf{r}$ e $\mathbf{u}$.

As Equações 4.7 e 4.8 demonstram o processo de reescala dos vetores de característica. Por fim, a Equação 4.9 fornece o valor final do processo de soma ponderada, onde W_k é um valor real do peso para uma característica k e $D_k(\mathbf{r},\mathbf{u})$ a distância entre $\mathbf{r}$ e $\mathbf{u}$ para o vetor de característica de k, onde a soma de todos os pesos W_k deve ser igual a 1.

$$T_k(\mathbf{r}) = \sum_{\mathbf{z} \in U(i,\mathbf{r},rad)} D_k(\mathbf{r},\mathbf{z}) \tag{4.7}$$

$$\Lambda_{\mathbf{r},k}(\mathbf{u}) = \left(\frac{D_k(\mathbf{r},\mathbf{u})}{T_k(\mathbf{r})} \right) W_k \tag{4.8}$$

$$FD_{\mathbf{r}}(\mathbf{u}) = \sum_{k} \Lambda_{\mathbf{r},k}(\mathbf{u}) \tag{4.9}$$

O valor $FD_{\mathbf{r}}(\mathbf{u})$, obtido pelo processo citado anteriormente, é o valor final, para efeitos de comparação, de uma determinada região candidata, logo, deve-se escolher, dentre as candidatas, a região com menor valor. Uma vez que o cálculo é baseado nas distâncias dos vetores, esta região será considerada a mais similar à região atual.

4.6 Transferência e propagação de cores

Na Seção anterior foi apresentada a forma de seleção da região mais similar à região atual **r**, a partir desta região iremos extrair as informações de cores.

Neste trabalho foi escolhido o modelo de cor **CIE L*a*b** (ISO 11664-4:2008, 1976), ou simplesmente modelo L.A.B, que separa a imagem em três informações, sendo **L** a camada de luminância ou intensidade, **a** a intensidade da luz de verde ao vermelho e **b** para a luz azul à amarela. Importante destacar que apesar da escolha deste modelo, outros poderiam ser utilizados, desde que separem as informações de crominância das informações de intensidade.

Como uma imagem em tons de cinza possui somente a camada de intensidade, a colorização poderia ser feita por meio do preenchimento das informações das camadas **a** e **b** do modelo L.A.B. Entretanto, conforme Kimmel (1998); Yatziv e Sapiro (2006) as mudanças graduais de intensidade de uma imagem geralmente indicam transições de crominância, em imagens naturais.

Dessa forma, o preenchimento uniforme e direto do valor de crominância obtido no processo de comparação para toda a área da região acabaria por gerar um resultado artificial em vários casos. A fim de evitar esse problema e gerar um resultado mais natural, o método propõe a utilização do algoritmo descrito em Levin et al. (2004).

Como já comentado no Capítulo2, o algoritmo realiza a colorização por meio de um processo de otimização, no qual devem ser fornecidos "rabiscos"como a informação de cor inicial. Para esse método, em vez de aplicar diretamente a cor, são gerados pequenos pontos coloridos, de apenas um *pixel* cada, que contém a cor obtida no processo de comparação. Após isso, o algoritmo citado é aplicado e o mesmo realiza a distribuição das cores para o restante da imagem com transições graduais de crominância. A Figura 4.7 mostra o resultado da geração da imagem com os pontos coloridos que serviram de entrada para o algoritmo proposto em Levin et al. (2004).

A Figura 4.6 demonstra a diferença entre uma imagem colorida diretamente, na qual apenas foram transferidas as informações de cor diretamente, e outra colorida por meio do algoritmo de otimização. Podemos perceber que na Figura 4.8(a) a cor dos lábios e dos monitores ao fundo não ficam naturais, principalmente devido à mudança brusca de cor. Já na Figura 4.8(b) podemos perceber uma transição mais natural das cores.

Figura 4.7: Pontos coloridos gerados para a utilização do algoritmo de colorização.

(a) (b)

Figura 4.8: Comparação de um *frame* da sequência Akiyo colorida via aplicação direta de cor e via o algoritmo de (Levin et al., 2004).

A saída do algoritmo de colorização é demonstrada na Figura 4.9. Essa imagem representa o resultado final que ira compor o conjunto colorido de *frames*.

Figura 4.9: Imagem colorizada após aplicação do algoritmo.

4.7 Conferência do usuário

Após a colorização do *frame*, o mesmo deverá ser submetido à avaliação do usuário. Mesmo com taxas de acerto boas, é possível haver alguns erros na hora de designar as cores para uma região, bem como há a possibilidade de oclusão e de aparição de novos objetos na cena e como não há informação prévia da cor desses novos objetos há a possibilidade que a cor aplicada seja a errada.

A avaliação do usuário é a oportunidade do mesmo conferir e interferir no resultado que foi gerado automaticamente, corrigindo ou até mesmo aplicando novas cores aos objetos, ou então simplesmente aprovar o resultado.

Para este trabalho, propõe-se a criação de uma aplicação de interface gráfica, no qual o usuário verá o resultado da colorização para o *frame* atual e poderá interagir selecionando uma cor da paleta, bem como também poderá escolher uma cor já existente na imagem. Após isso ele poderá rabiscar livremente sobre a imagem, as regiões por onde o rabisco do usuário passou receberão a cor selecionada. Quando estiver satisfeito com o resultado o usuário solicitará o processamento do *frame* seguinte, ou, caso já tenha processado, passará para a análise do próximo. A Figura 4.10 mostra a tela dessa aplicação. Nela

foi selecionado um *frame* onde ocorreram alguns erros na designação das cores. A figura também mostra o uso da ferramenta através de um traço vermelho, aplicando a cor selecionada na região. A Figura 4.11 mostra o antes e o depois da intervenção realizada pelo usuário. Note que na imagem da esquerda houve um pequeno erro na fase automática onde foi atribuída a cor vermelha a uma parte do rosto. Após a correção a cor do rosto volta ao normal.

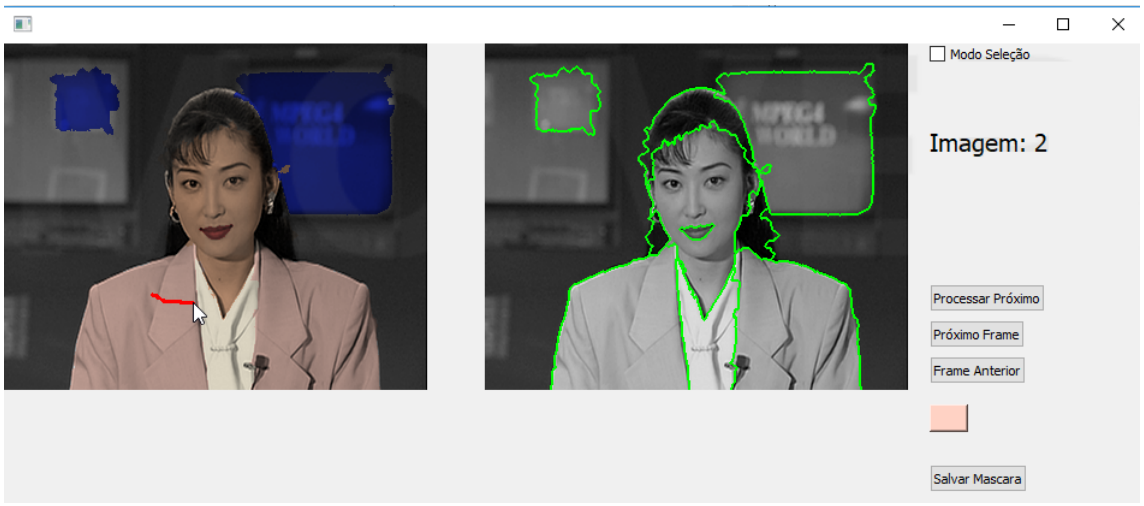


Figura 4.10: Tela da ferramenta para o processo de conferência do usuário.

Figura 4.11: Antes e depois do processo de correção manual pelo usuário.

5

Experimentos

Neste capítulo são descritos os experimentos realizados com o método proposto, bem como as métricas usadas para realizar a avaliação.

5.1 Ambiente de execução dos experimentos

Todos os experimentos foram executados em um computador de mesa com: Processador Intel i7-6700 3.40Ghz, 8GB de memória RAM e sistema operacional Windows 10 (64-Bits). A implementação foi feita em linguagem Python 2.7 com o uso das bibliotecas Numpy (Walt et al., 2011), OpenCV (Bradski e Kaehler, 2000), SciPy (Jones et al., 2014), Cython (Behnel et al., 2011) e Mamba (Beucher e Beucher, 2011). A parte de otimização do algoritmo foi implementada em linguagem C++ e integrada ao Python.

Para criação da segmentação hierárquica foi utilizada a *Max-tree* montada sobre o negativo do gradiente da imagem. A partir de então foram aplicados os filtros de extinção por área de forma a preservar apenas a quantidade desejada de mínimos regionais. Para as operações com a *Max-tree* foi utilizada a Tool-Kit SIamxt (Souza et al., 2017). A tool-box apresentou desempenho excelente sendo mais rápido do que outros métodos de extinção implementados puramente em Python, sendo possível realizar a montagem da árvore, aplicação do filtro e recuperação da imagem em tempo de médio de 15 milissegundos.

5.2 Parâmetros dos algoritmos

O seguintes parâmetros foram utilizados para a realização dos experimentos:

1. Algoritmo de superpixel: para a escolha do número de regiões buscou-se um equilíbrio de forma a conseguir uma quantidade de regiões suficientes para separar os objetos e ao mesmo tempo regiões grandes suficientes para conseguir se extrair informações relevantes de textura e intensidade. Portanto, tendo em vista as dimensões das imagens utilizadas, foi escolhida a quantidade de 1500 regiões.

2. Área de Extinção: foi definido como parâmetro a preservação de 950 mínimos regionais de acordo com a sua área. Da mesma forma que no superpixel, deve-se levar em consideração o tamanho da imagem na hora da seleção desse quantitativo.

3. Filtros de Gabor: para a extração de descritores de textura utilizando Gabor, foi utilizado 40 Filtros de Gabor com orientações variando de 0 a $7\pi/8$ e cinco escalas exponenciais $exp(i \times \pi) : i = \{0, 1, 2, 3, 4\}$, buscou-se dimensionar o tamanho do banco de descritores para se obter um bom poder descritivo e um tempo de processamento aceitável;

4. LBP: para o LBP foi utilizada uma vizinhança de 8 pontos com raio 1 e o histograma foi gerado somente para os pontos que estejam dentro da região desejada;

5. Raio de busca: foi escolhido um raio de tamanho igual a 25% da largura da imagem. Como explicado no Capítulo 4, o tamanho escolhido deve possuir relação com a taxa de quadros da sequência. Para as imagens utilizadas neste experimento, 25% se mostrou suficiente.

6. Pesos: foram usados os seguintes pesos para ponderar os descritores: 0,25 para Intensidade, 0,30 para LBP e 0,45 para Gabor. Esses valores foram obtidos baseado nos experimentos descritos na Seção 5.4.

5.3 Sequências de imagens utilizadas

Para a realização dos experimentos, foram utilizadas três sequências de imagens, cada uma composta de 150 *frames*. As sequencias são intituladas *Foreman*, *Akiyo* e *Carphone*. A Tabela 5.1 lista as dimensões de cada sequência. Tais sequências são originalmente coloridas, no entanto, para os experimentos foram removidas as informações de cor trabalhando unicamente com as suas versões em escala de cinza. Foi necessário utilizar uma sequência originalmente colorida para ser possível usar uma das métricas de avaliação. A Figura 5.1 demonstra o primeiro quadro de cada uma das sequências utilizadas.

Tabela 5.1: Dados das sequências utilizadas.

Nome	Dimensões
Foreman	352x280
Akiyo	352x288
Carphone	176x144

Figura 5.1: Primeiro *frame* de cada sequência, da esquerda para direita a sequência de *Foreman*, Akiyo e Carphone.

5.4 Escolha do Parâmetro de Extinção por Área

Diversos experimentos foram realizados para determinar qual melhor o parâmetro de extinção a ser aplicado no presente problema. Foram selecionados três tipos de parâmetros de extinção para comparação, são eles: os por área, por altura e por volume. Foi realizado um teste aplicando os três parâmetros em todos os *frames* das três sequências, variando-se também os números de mínimos a serem preservados, uma vez que os parâmetros de extinção podem apresentar resultados melhores com quantidades de mínimos diferentes.

O resultado dessa segmentação foi rotulado baseado em um Padrão Ouro e então foi realizado o processo de propagação a partir do qual foi possível se calcular o percentual de erros. A Figura 5.2 demonstra o resultado desses experimentos onde cada ponto corresponde a média de todos os *frames* das três sequências para aquele determinado parâmetro e quantitativo de mínimos. A Tabela Tabela 5.2 resume os dados demonstrando a média geral dos erros para cada parâmetro. Como pode ser observado, o parâmetro por área foi ligeiramente melhor que o de extinção por volume. Utilizamos este experimento também para definir uma quantidade adequada de mínimos regionais a serem preservados. Analisando os dados coletados na Figura 5.2 entendemos que 950 mínimos seria adequado para o presente caso, por gerar um baixo número de erros, bem como demanda menos tempo de processamento.

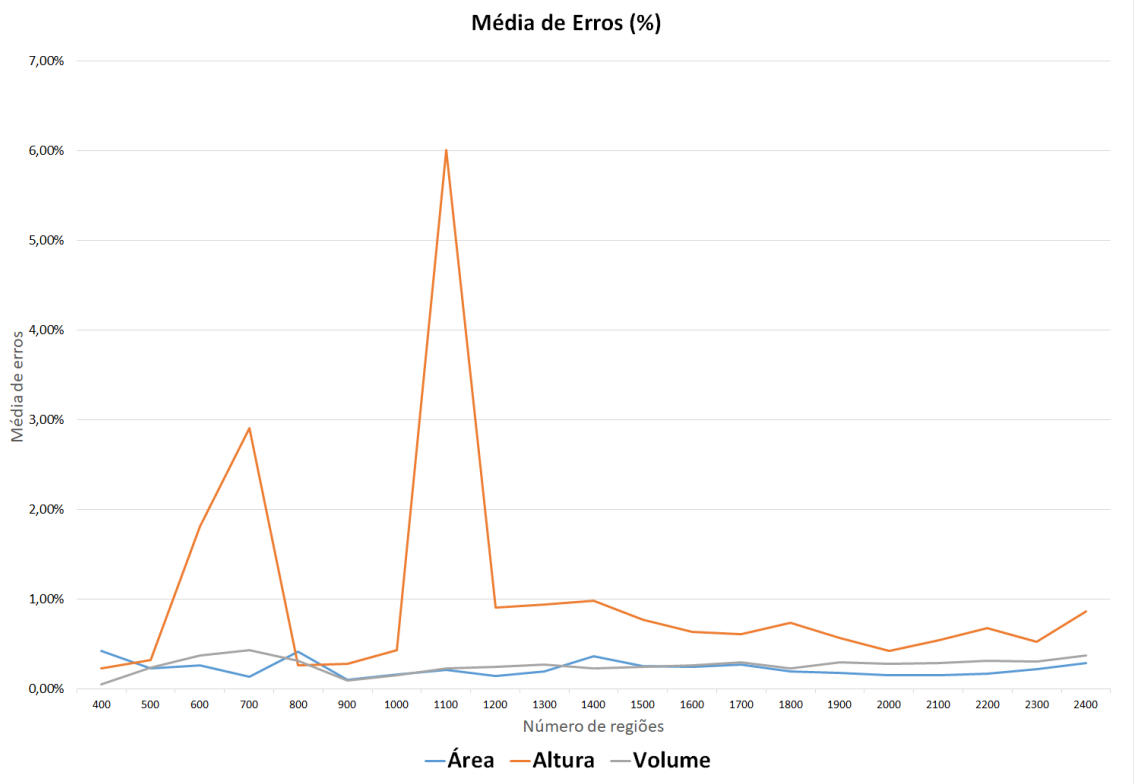


Figura 5.2: Experimentos com parâmetros de extinção.

Parâmetro	**Erro**
Área	0,23%
Altura	1,02%
Volume	0,26%

Tabela 5.2: Resumo dos experimentos com parâmetros de extinção.

5.5 Pesos dos vetores de características

Para determinarmos uma boa parametrização dos pesos usados para ponderar os vetores de características, vários experimentos foram realizados. Em um primeiro momento, foram realizados experimentos individualmente com cada tipo de características, identificando o grau de erro de classificação de cada uma. Para determinar o grau de erro, foram criados *frames* rotulados manualmente contendo as mesmas regiões do *frame* a ser analisado o erro. Consideramos esse *frame* rotulado com um Padrão Ouro, que seria o valor ideal. Após o processo de comparação de características foi verificado, para cada região, se a mesma foi classificada corretamente. Após isto foi somado o percentual de erro de

todas as regiões designadas incorretamente, em que o percentual de erro de uma região é proporcional à quantidade de pontos presentes nela.

A Figura 5.3 demonstra os erros de classificação para a sequência Foreman com a utilização individual do LBP, de Gabor, intensidade e com o uso dos três, de forma ponderada. A Figura 5.4 demonstra as mesmas métricas, porém para a sequência Akiyo e a Figura 5.5, para sequência Carphone.

Pode-se observar que todas as sequências as características tem comportamento similar, sendo que a intensidade demonstra o maior erro, o que provavelmente se deve pela falta de informações de textura. O LBP mantém um percentual de erro abaixo da intensidade e Gabor apresenta uma taxa consideravelmente mais baixa que LBP e intensidade. Já a junção das três características apresenta o nível mais baixo, quase sempre ficando abaixo dos erros com Gabor, apenas em alguns casos ficando acima, porém influenciada pelo erro das outras características. Para a utilização das três característica em conjunto as mesmas foram ponderadas, sendo que os pesos foram definidos de acordo com o percentual de acerto de cada característica. A Tabela 5.3 mostra a média de erro de cada característica para as três sequências. Ao fim, mostra o erro com a utilização das três características em conjunto, já ponderadas, de acordo com os valores obtidos e descritos na Subseção 5.2.

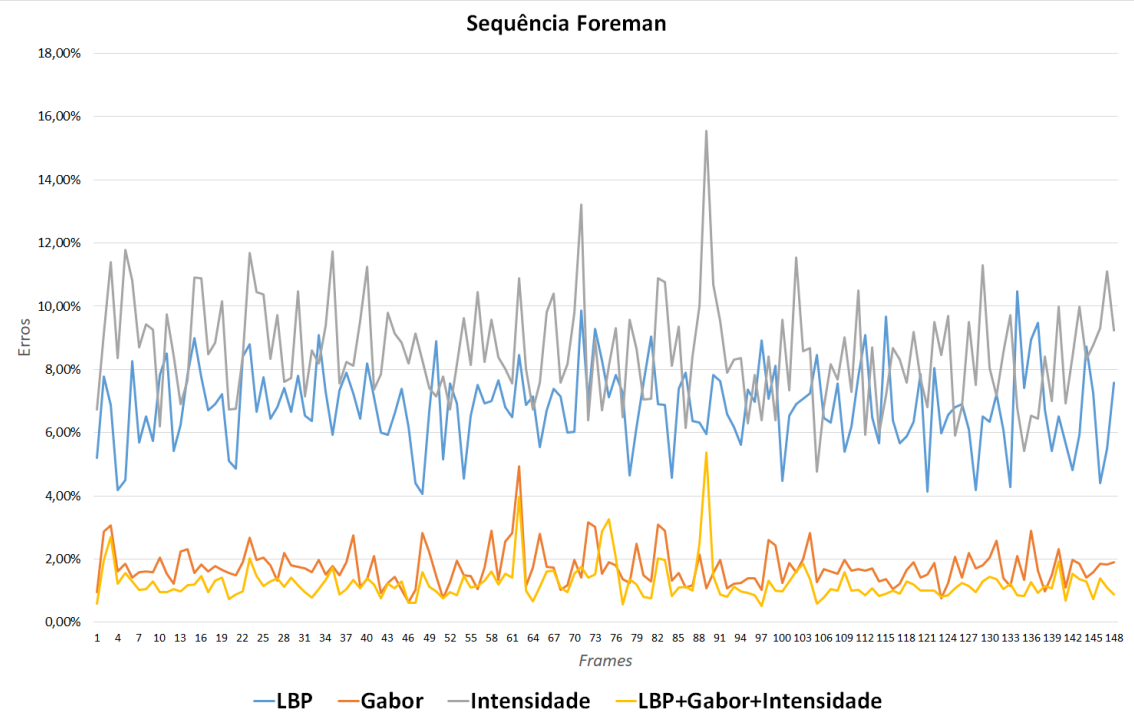


Figura 5.3: Erros de classificação de cada característica na sequência Foreman

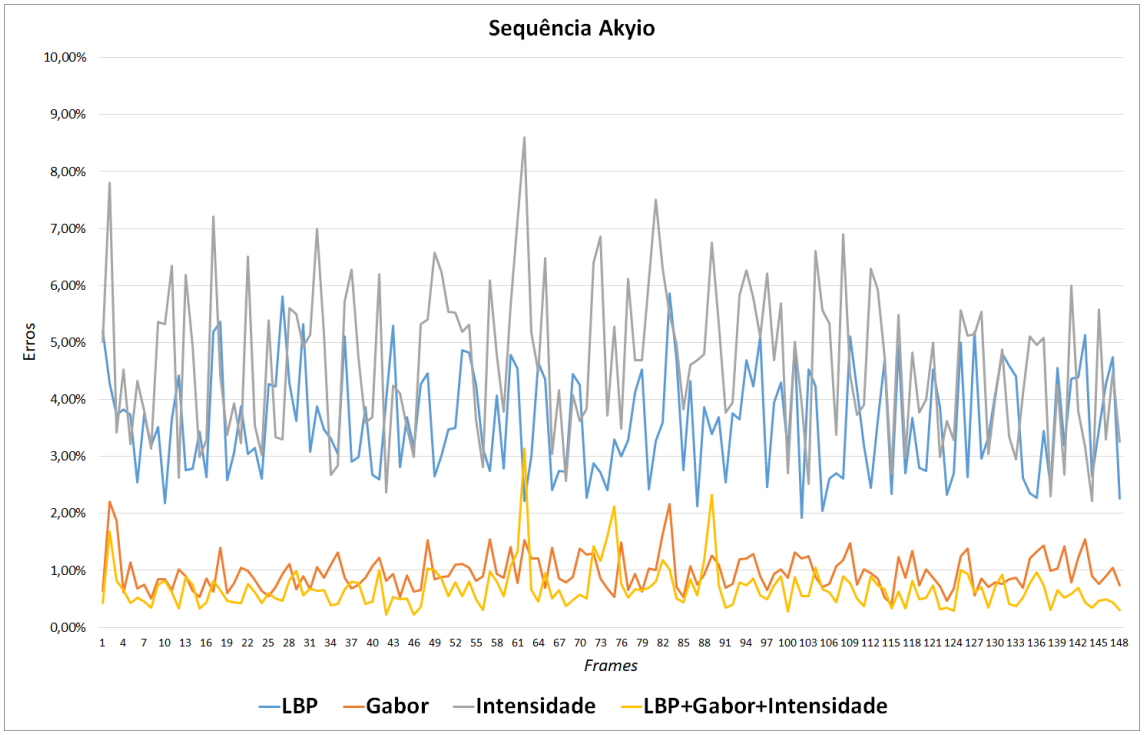


Figura 5.4: Erros de classificação de cada característica na sequência Akiyo

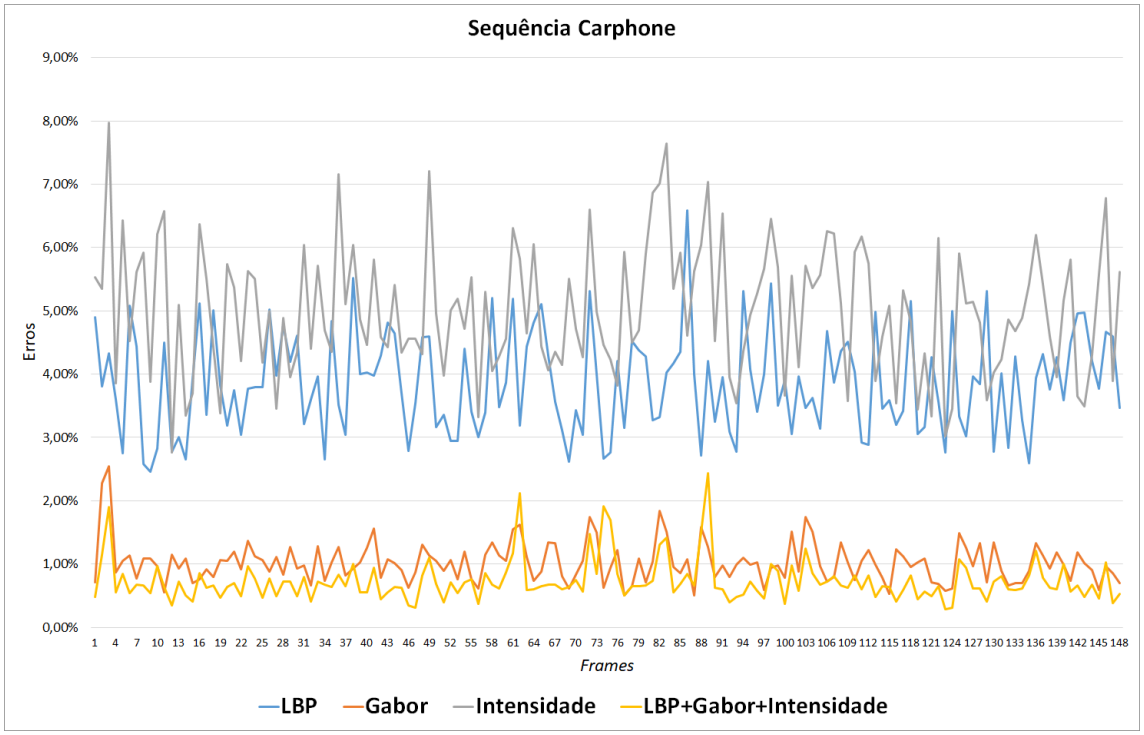


Figura 5.5: Erros de classificação de cada característica na sequência Carphone

Tabela 5.3: Média de erro do uso individual das características.

Característica	Média de erros
LBP	4,76%
Gabor	1,24%
Intensidade	6,08%
LBP+Gabor+Intensidade	0,89%

Como visto na tabela Tabela 5.3 os descritores de textura tiveram uma menor taxa de erros, o que já era de se esperar, uma vez que a intensidade possui dados menos relevantes para efeitos de comparação.

5.6 Métricas de avaliação (Benchmark)

Conforme Bharati et al. (2004) não existe na literatura um método absoluto para medição da desempenho de um algoritmo de colorização, sendo que se acredita que a melhor avaliação ainda é pela observação humana. Entretanto, devido a falta de tal método de avaliação, vários trabalhos utilizam a medida *Peak Signal to Noise Ration* (PSNR), que fornece uma forma relativa de medir o desempenho de colorização.

Neste trabalho foram utilizadas três métricas para avaliação a fim de criar um *benchmark* para avaliação da colorização. Tal *benchmark* foi inspirado no *benchmark* proposto no trabalho de Flores et al. (2008) que avalia métodos assistidos de segmentação. As métricas selecionadas para o presente trabalho são: (a) tempo de processamento e número de interferências do usuário por *frame*, (b) erro de segmentação e (c) *Color Peak Signal to Noise Ratio* (CPSNR). Nas subseções seguintes são detalhadas cada uma das métricas.

5.6.1 Tempo de Processamento e Número de interferências por Frame

Como o objetivo principal deste trabalho é reduzir o tempo e a quantidade de trabalho envolvidos na colorização de sequências de imagens, uma forma de avaliar é justamente quantificar o quanto de fato se está economizando de tempo e trabalho comparado com um método manual.

O método aqui denominado "manual"consiste na aplicação de um método de segmentação assistida, como o Watershed, pelo usuário para se definir as regiões de interesse e, após isso, a seleção de cor para cada região da imagem.

Para esse métodos, foi analisado o tempo que leva, em média, para se colorir cada *frame*, bem como a quantidade de correções que o usuário teve de realizar para que o resultado ficasse natural. Para o método proposto, foi considerado o tempo que o método leva para realizar a etapa automática mais o tempo que o usuário leva para analisar e corrigir o *frame*. Como interferência, foi considerado o total de alterações de cores, realizadas pelo usuário, na imagem colorida automaticamente.

Para o método manual, levou-se em conta o tempo que usuário leva para segmentar a imagem e definir as cores, como sendo o tempo total. A interferência foi calculada como o total de marcadores adicionados, alterados e removidos pelo usuário para alcançar a segmentação desejada.

5.6.2 Erros de segmentação

O método de colorização proposto pode ser definido como um subproblema de segmentação de imagens, por isso, uma medição da qualidade da segmentação pode ser usada como referência para avaliar o desempenho do método.

Importante ressaltar, no entanto, que como o método aplica um algoritmo para realizar a distribuição das cores, muitas vezes um pequeno erro na segmentação pode

não necessariamente gerar um erro perceptível no resultado final. De toda forma, o erro de segmentação ainda serve como referência para analisar a robustez do método.

Para computar o valor do erro da segmentação foi utilizada uma versão modificada da métrica proposta em por Flores e de Alencar Lotufo (2010). Foi necessário realizar uma modificação pois a métrica original leva em consideração que há somente dois objetos resultantes da segmentação, sendo fundo e objeto. Entretanto, no presente método podemos ter diversos objetos dependendo da quantidade de cores e regiões da imagem.

Para efetuar esse cálculo precisamos de um padrão de referência, também denominado Padrão Ouro. Esse padrão representa o que seria uma segmentação boa para um usuário, capaz de particionar completamente os objetos desejados. No caso presente, foi realizada uma segmentação manual de todos os *frames* das três sequências, definindo classes para cada região.

Como o presente método utiliza segmentação automática, o que tende a gerar uma super segmentação. Para ser possível efetuar a comparação é necessário agrupar regiões adjacentes, com a mesma informação de crominância, para gerar grandes regiões de apenas uma cor.

Seja Z_i o conjunto de $\mathbf{z}_m$ regiões, de tal forma que $\mathbf{z}_m$ seja a união de regiões de S_i. Uma região $\mathbf{r} \in S_i$ pertence a $\mathbf{z}_m \in Z_i$ se existir um $\mathbf{r}_2 \in Z_i$ de tal forma que $\mathbf{r}$ e $\mathbf{r}_2$ são adjacentes e $C_i(x) = C_i(y), x \in \mathbf{r}, y \in \mathbf{r}_2$.

Seja m_v uma máscara de segmentação binária para a região v da segmentação Z_i, onde 1 indica que aquele ponto pertence a região e 0 caso contrário. Seja g_v máscara de segmentação do Padrão Ouro para a mesma região v. Sendo ψ a diferença simétrica entre m_v e g_v, dada pela Equação 5.1

$$\psi(m_v, g_v) = \begin{cases} 1 & \text{se } |m_v - g_v| = 1, \\ 0 & \text{o contrário} \end{cases} \tag{5.1}$$

Computa-se o erro de segmentação $SE(Z_i)$ para a segmentação Z_i como a média do erro de todas as regiões. As Equações 5.2 e 5.3 demonstram o cálculo, onde $\#(m_v)$ denota o número de pontos com valor 1 para a máscara binária m_v. O valor do erro de cada região é dividido por dois para evitar a soma duplicada do erro, pois um erro em uma região também equivale a um erro na região adjacente.

$$ER(m_v) = \frac{\#(\psi(m_v, g_v))}{\#(m_v) + \#(g_v)} \tag{5.2}$$

$$SE(Z_i) = \frac{1}{n}\sum_{v=1}^{n}\frac{ER(m_v)}{2} \tag{5.3}$$

5.6.3 Color Peak Signal to Noise Ratio

O CPSNR é uma adaptação do PSNR para imagens coloridas utilizando os canais de cor Vermelha, Verde e Azul, muito utilizado para avaliação de técnicas de processamento de vídeo e imagens(Bharati et al., 2004; Pang et al., 2014). O objetivo dessa métrica é calcular a distorção dos canais de cores entre duas imagens.

Dadas duas imagens coloridas de 8-bits, O_1 e O_2 de dimensões $H \times W$, o CPSNR é calculado conforme as Equações 5.4 e 5.5, sendo que quanto maior o CPSNR, menor a distorção.

$$CPSNR = 10\log_{10}\frac{255^2}{MSE} \tag{5.4}$$

$$MSE = \frac{1}{3HW}\sum_{\Omega\in(R,G,B)}\sum_{i=1}^{H}\sum_{j=1}^{W}(O_1^{(\Omega)}(i,j) - O_2^{(\Omega)}(i,j)) \tag{5.5}$$

5.7 Resultados

Nesta seção são mostrados os resultados dos experimentos usando as três sequências de imagens já citadas. Os resultados foram individualizados por sequência, sendo exibido o resultado visual, bem como os valores das métricas aplicadas.

5.7.1 Sequência Foreman

Para a sequência *Foreman*, o resultado do processo de colorização pode ser visto na Figura 5.6 para segmentação hierárquica e na Figura 5.7 para superpixel, sendo que as figuras mostram somente metade dos 150 *frames* para uma melhor visualização. Os gráficos demonstrados nas figuras: Figura 5.8, Figura 5.9 e Figura 5.11 mostram os valores do tempo de colorização, quantidade de interferência do usuário e CPSNR respectivamente, tanto para o método proposto quanto para o manual. A Figura 5.10 mostra os percentuais de erros de segmentação para cada *frame*.

Pode-se observar que o tempo de colorização por *frame* para o método proposto é muito inferior ao método manual. Já entre as duas implementações do método proposto,

verifica-se que a utilização de segmentação hierárquica foi ligeiramente mais rápida que a utilização de superpixel. Como a diferença foi baixa não é possível afirmar que o método com superpixel é mais lento, até porque tal diferença pode ser consequência, por exemplo, da forma que o algoritmo foi implementado.

Na quantidade de interferência, o método proposto mostrou valores bem menores do que o método manual. Isso se deve pelo fato de, neste método, o usuário tem que apenas corrigir pequenas imperfeições em vez de realizar toda a colorização. Com relação às implementações do método, ambas apresentaram resultados muito similares.

No erro de segmentação, entre as duas implementações do método proposto, pode-se observar que há uma diferença clara entre a implementação com segmentação hierárquica e a com superpixel, sendo que a primeira apresentou um percentual de erro inferior ao superpixel em quase todos os casos, pois provavelmente a segmentação hierárquica conseguiu separar melhor os objetos da cena.

Os valores de CPSNR confirmam as métricas anteriores, uma vez que o método manual apresentou uma distorção maior (menor valor), possivelmente por ter ocorrido uma transferência direta de cor. Já na comparação entre as implementações do método proposto, a segmentação hierárquica apresentou resultado melhor. Novamente acredita-se que seja devido a uma melhor capacidade de separar os objetos do que a utilização com superpixel.

O resultado visual também é satisfatório, para ambas implementações, no qual detalhes como a gola da camisa, a pele e o capacete estão devidamente coloridos com uma tonalidade natural.

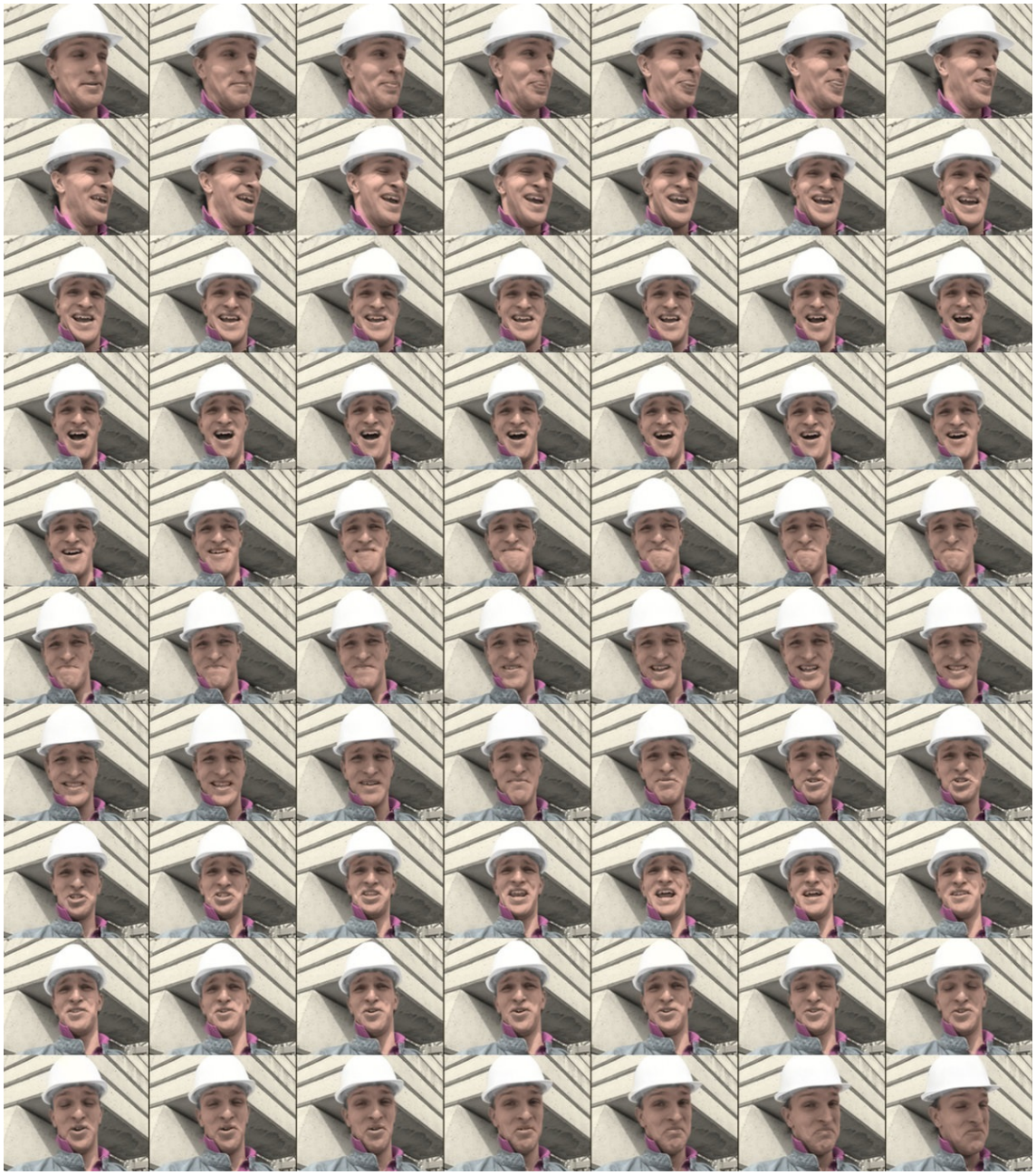


Figura 5.6: Resultado da colorização para sequência *Foreman*, segmentação hierárquica.

Figura 5.7: Resultado da colorização para sequência *Foreman*, segmentação por superpixel.

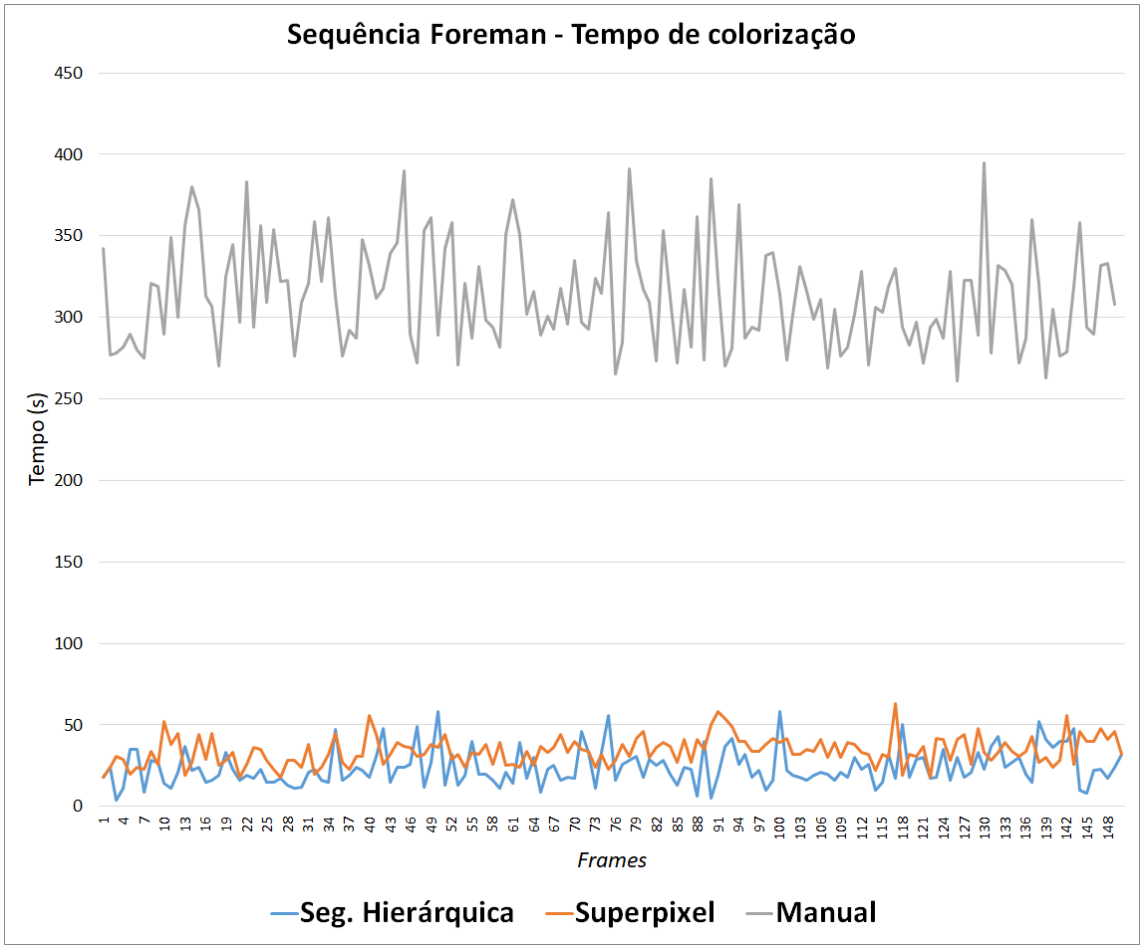


Figura 5.8: Tempo de colorização por *frame* para sequência *Foreman*.

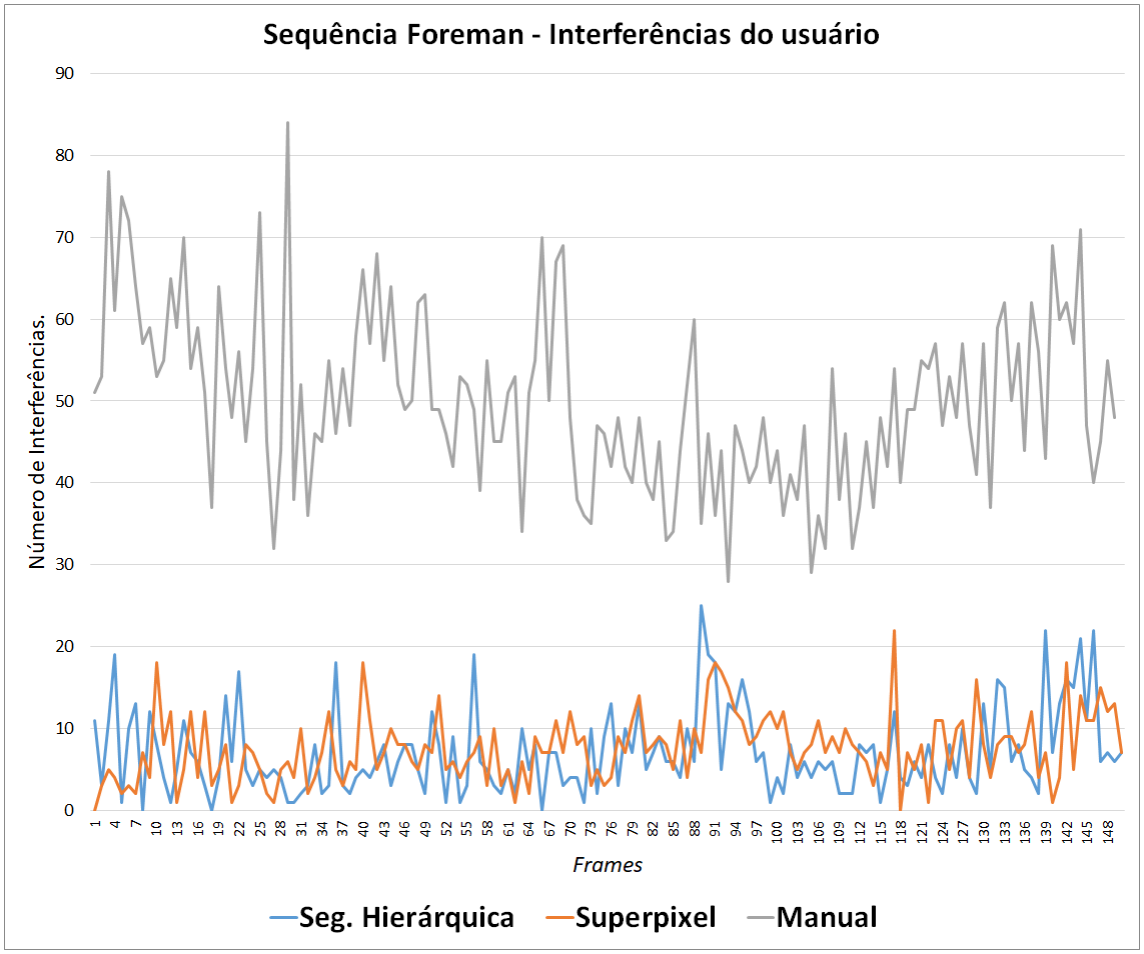


Figura 5.9: Interferências do usuário por *frame* para sequência *Foreman*.

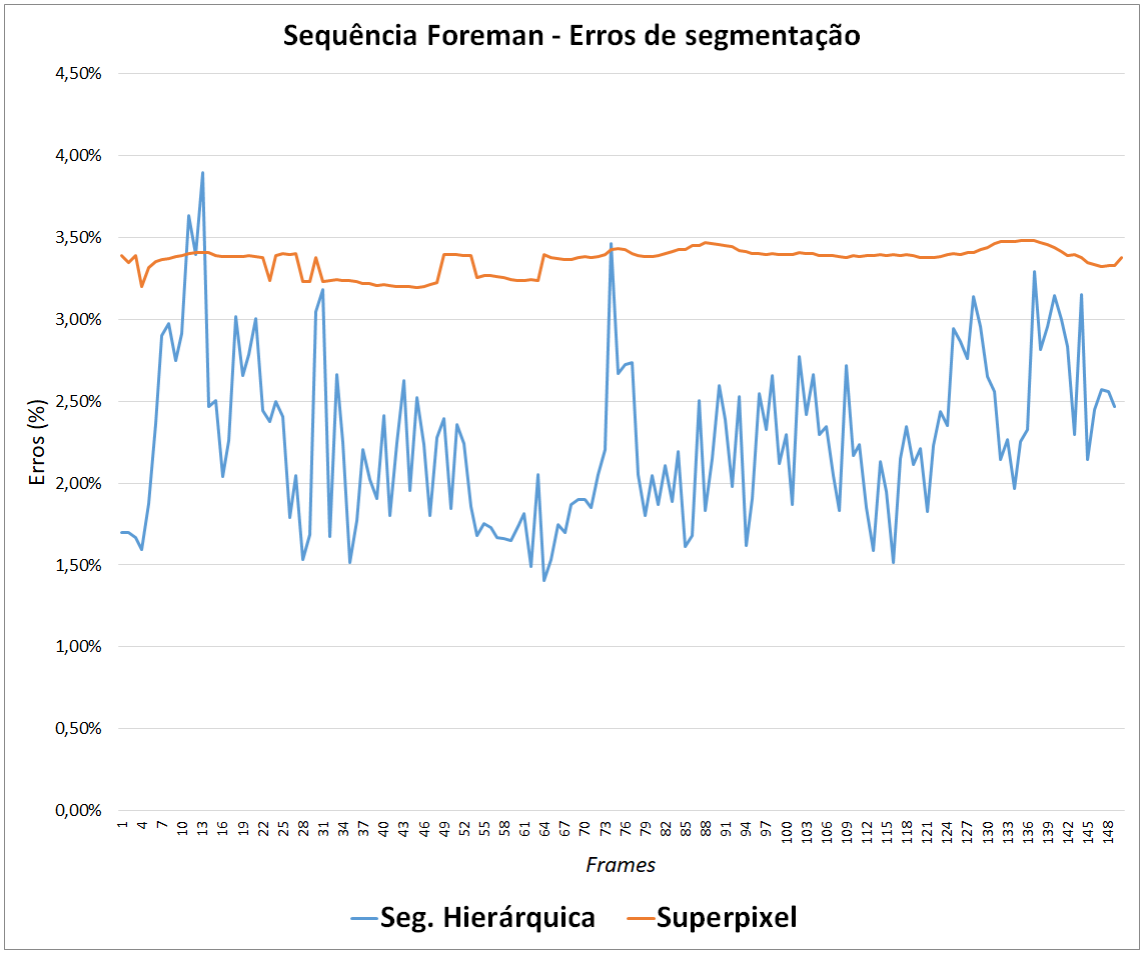


Figura 5.10: Erro de segmentação por *frame* para sequência *Foreman*.

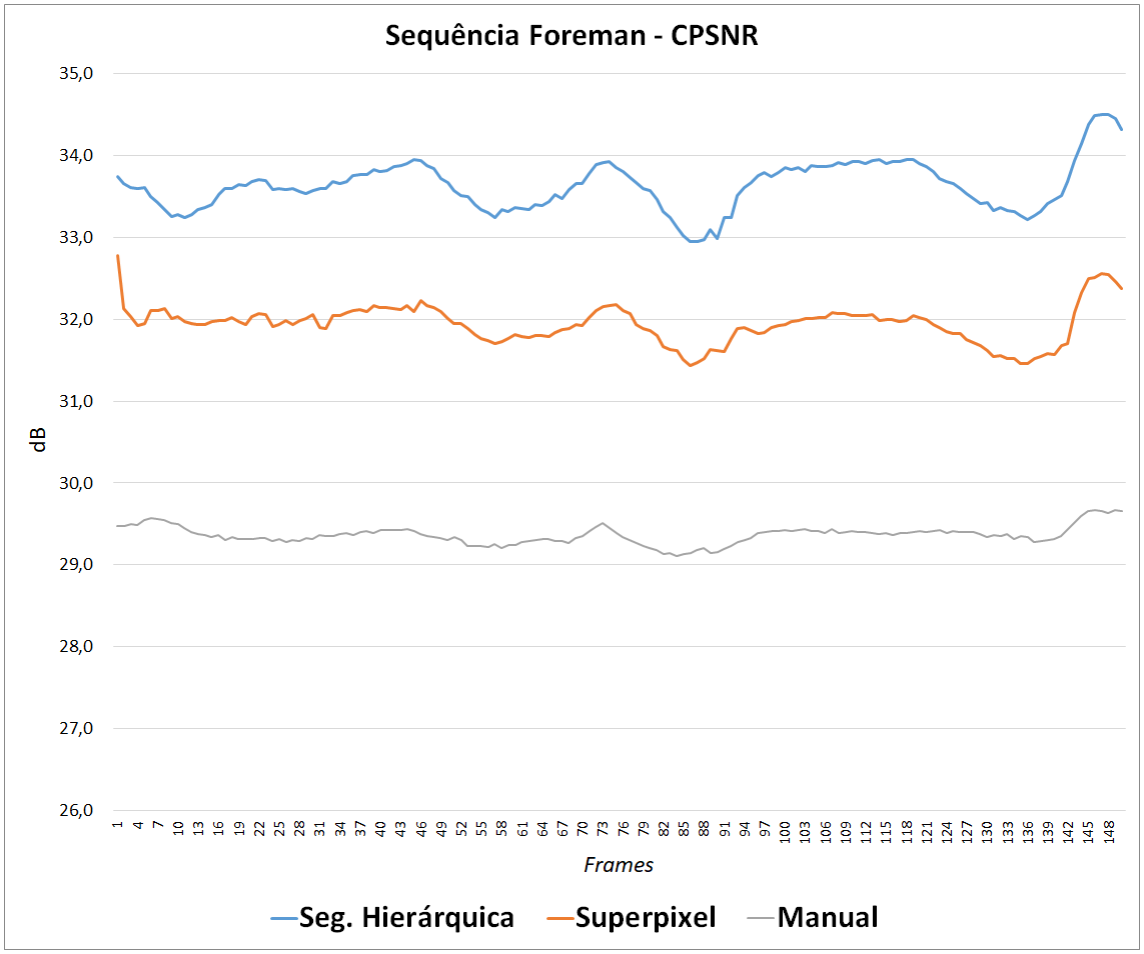


Figura 5.11: Valor do CPSNR por *frame* para sequência *Foreman*.

5.7.2 Sequência Akiyo

Para a sequência *Akiyo*, o resultado do processo de colorização pode ser visto na Figura 5.12 para segmentação hierárquica e na Figura 5.13 para superpixel. Novamente foram selecionados somente metade dos 150 *frames* para uma melhor visualização. Os gráficos de Figura 5.14, Figura 5.15 e Figura 5.17 mostram os valores do tempo de colorização, quantidade de interferência do usuário e CPSNR respectivamente, tanto para o método proposto quanto para o manual. A Figura 5.16 mostra os percentuais de erros de segmentação para cada *frame*.

Na presente sequência podemos ver comportamento similar ao observado para a sequência Foreman. O tempo da segmentação manual foi inferior se comparada com a sequência de foreman, devido a menor complexidade da cena. Mesmo assim a diferença de tempo entre o método proposto e o manual foi considerável. Entre as implementações do método proposto ambas apresentaram tempos similares.

Na quantidade de interferência, novamente vemos que o método proposto mostrou valores bem menores do que o método manual. Entre as implementações do método proposto, ambas apresentaram resultados similares entre si. Cabe destacar que o quantitativo geral de interferência foi no geral menor em relação à sequência de Foreman e isso também se deve pela menor complexidade da cena.

No erro de segmentação, novamente a implementação por superpixel apresentou quantidades de erros maiores, porém é possível perceber que nesta cena a diferença do erro é menor e oscila mais do que para a sequência de Foreman.

Para a métrica de CPSNR, nesta sequência podemos perceber, também, uma diferença considerável entre o método proposto e o manual. Destaca-se que a diferença entre as implementações é bem inferior as observadas na sequência de Foreman.

O resultado visual também é satisfatório. Como ocorreu na sequência de Foreman, as cores dos monitores ao fundo, da pele, roupa e boca estão bem definidas e apresentam características naturais.

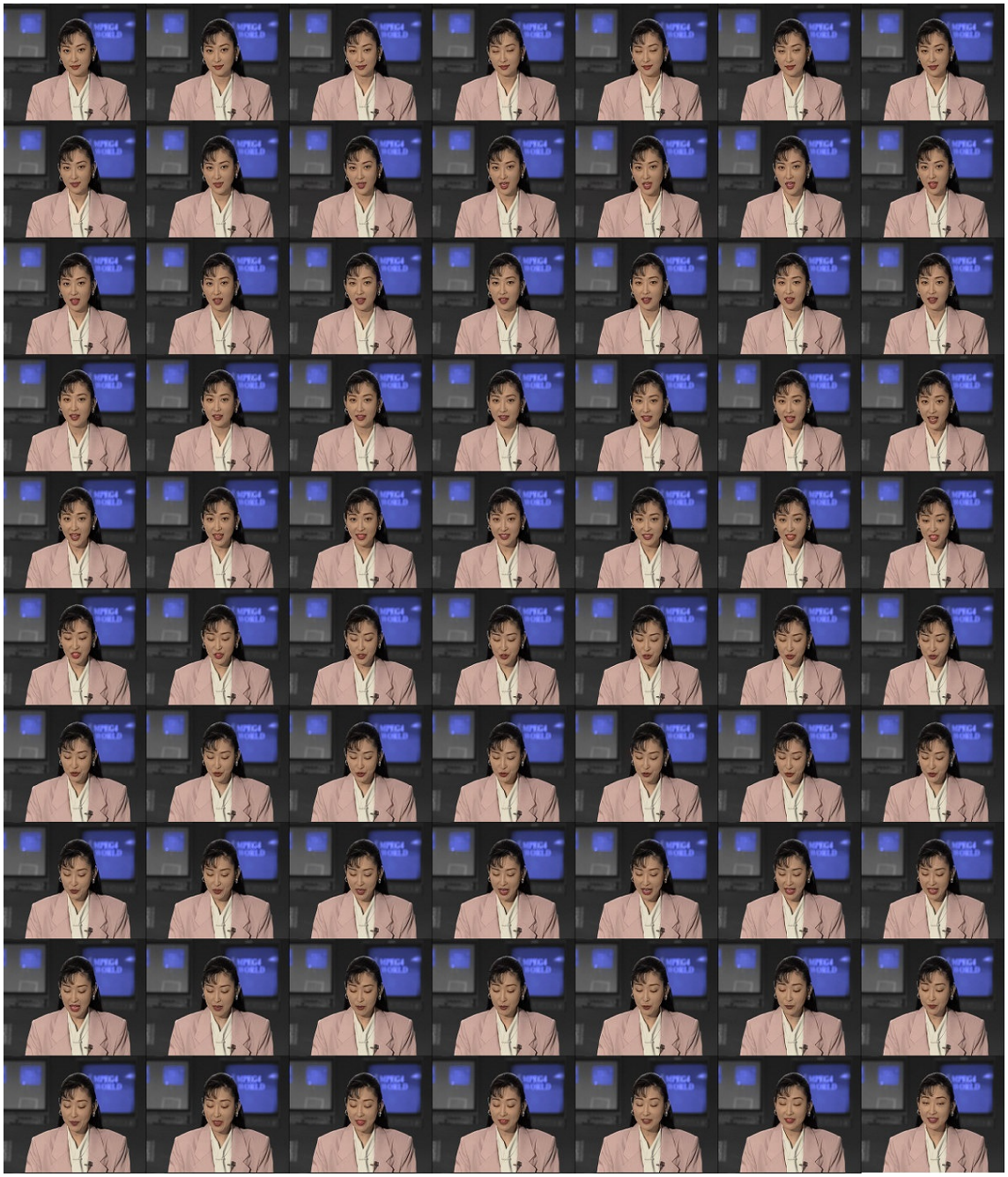


Figura 5.12: Resultado da colorização para sequência *Akiyo*, segmentação hierárquica.

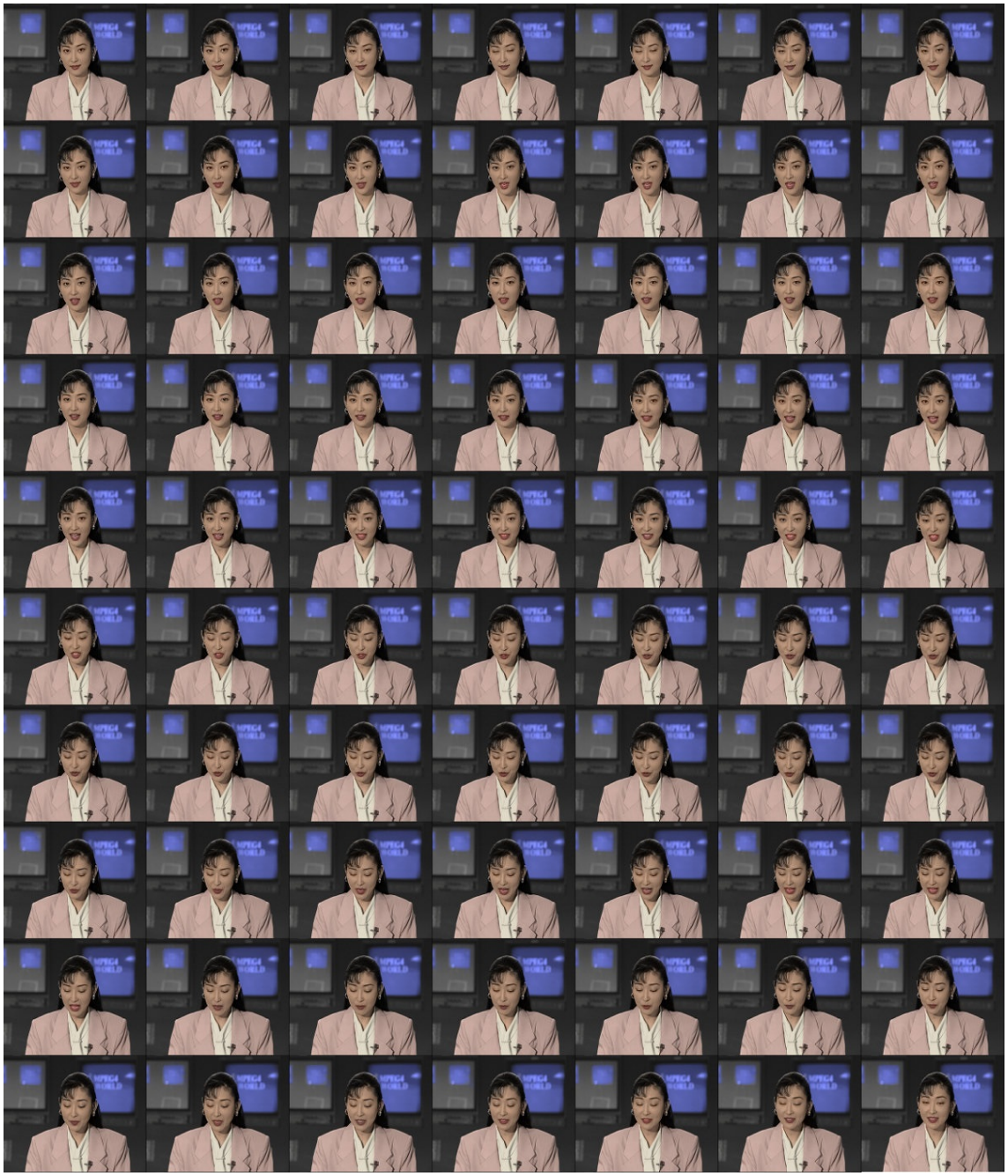


Figura 5.13: Resultado da colorização para sequência *Akiyo*, segmentação por superpixel.

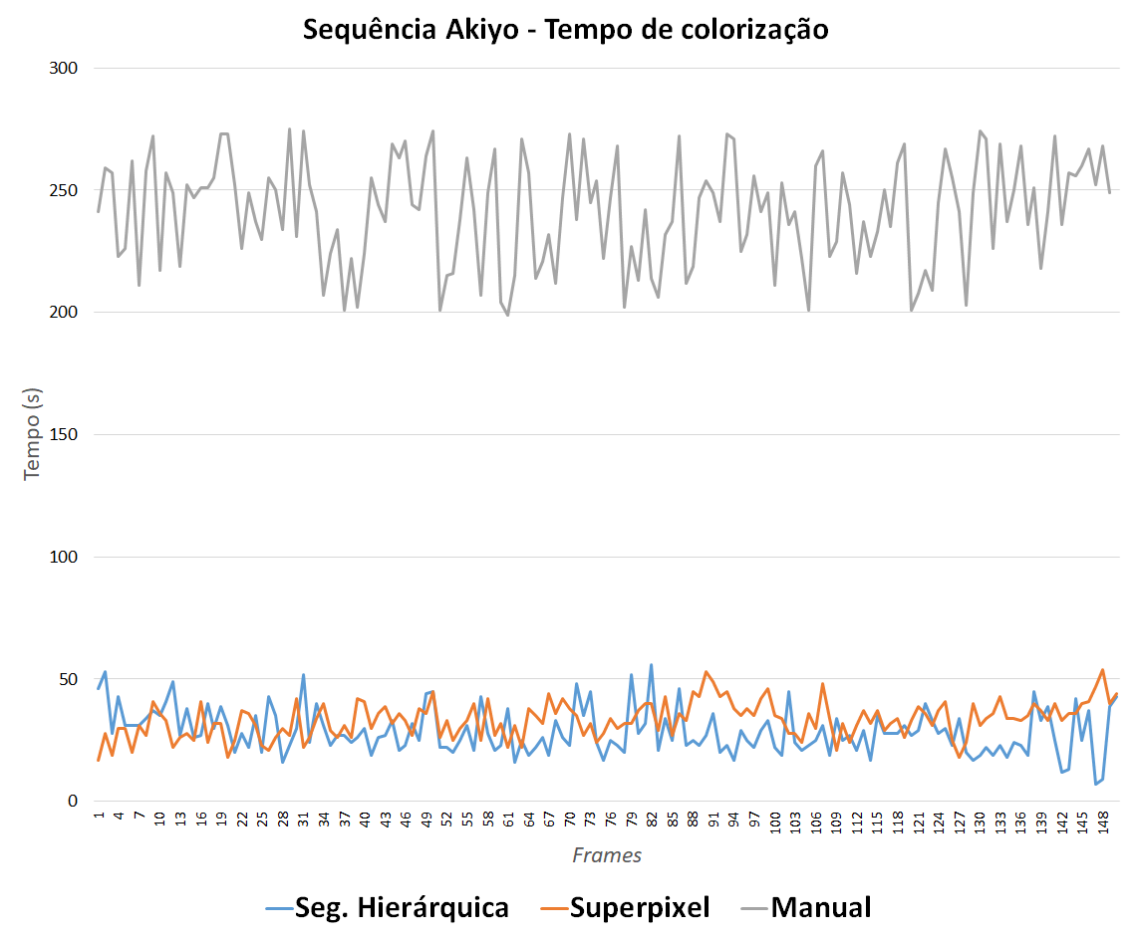


Figura 5.14: Tempo de colorização por *frame* para sequência *Akiyo*.

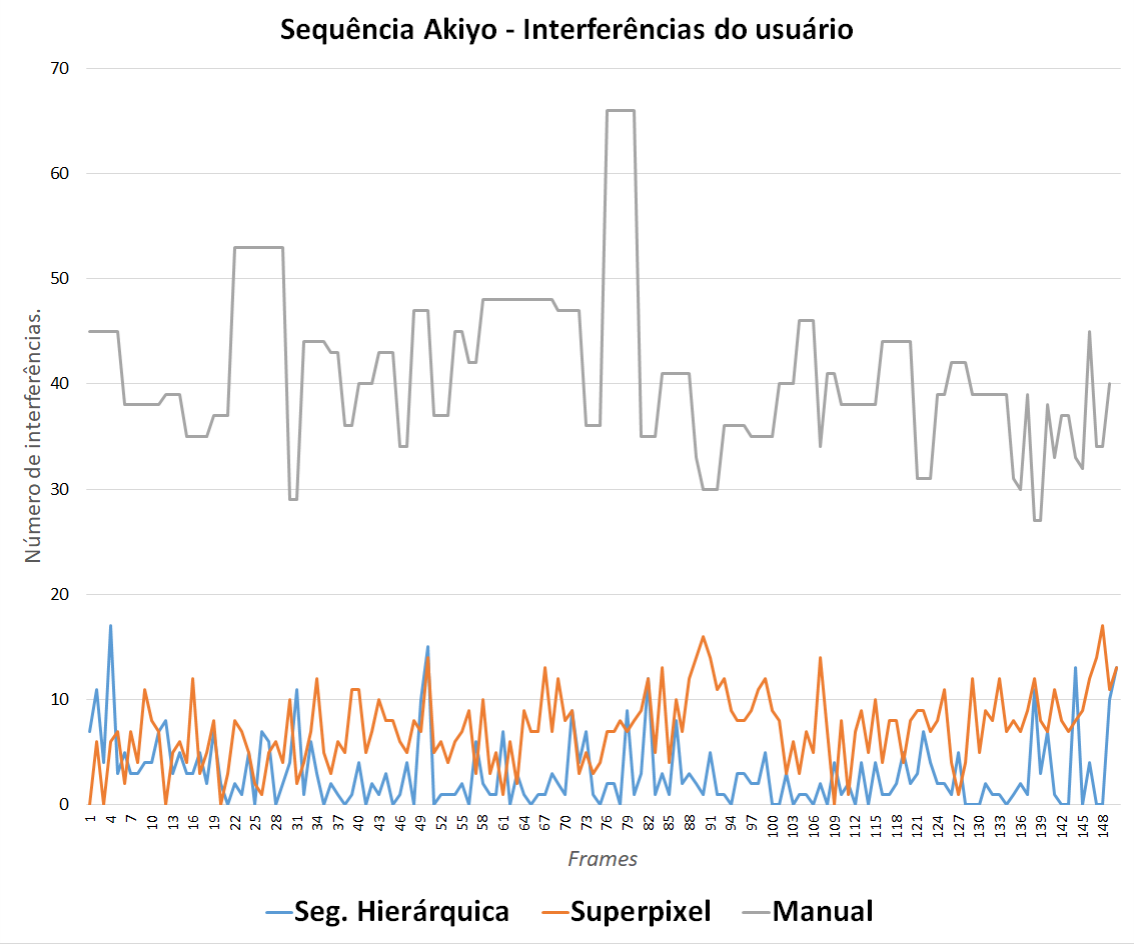


Figura 5.15: Interferências do usuário por *frame* para sequência *Akiyo*.

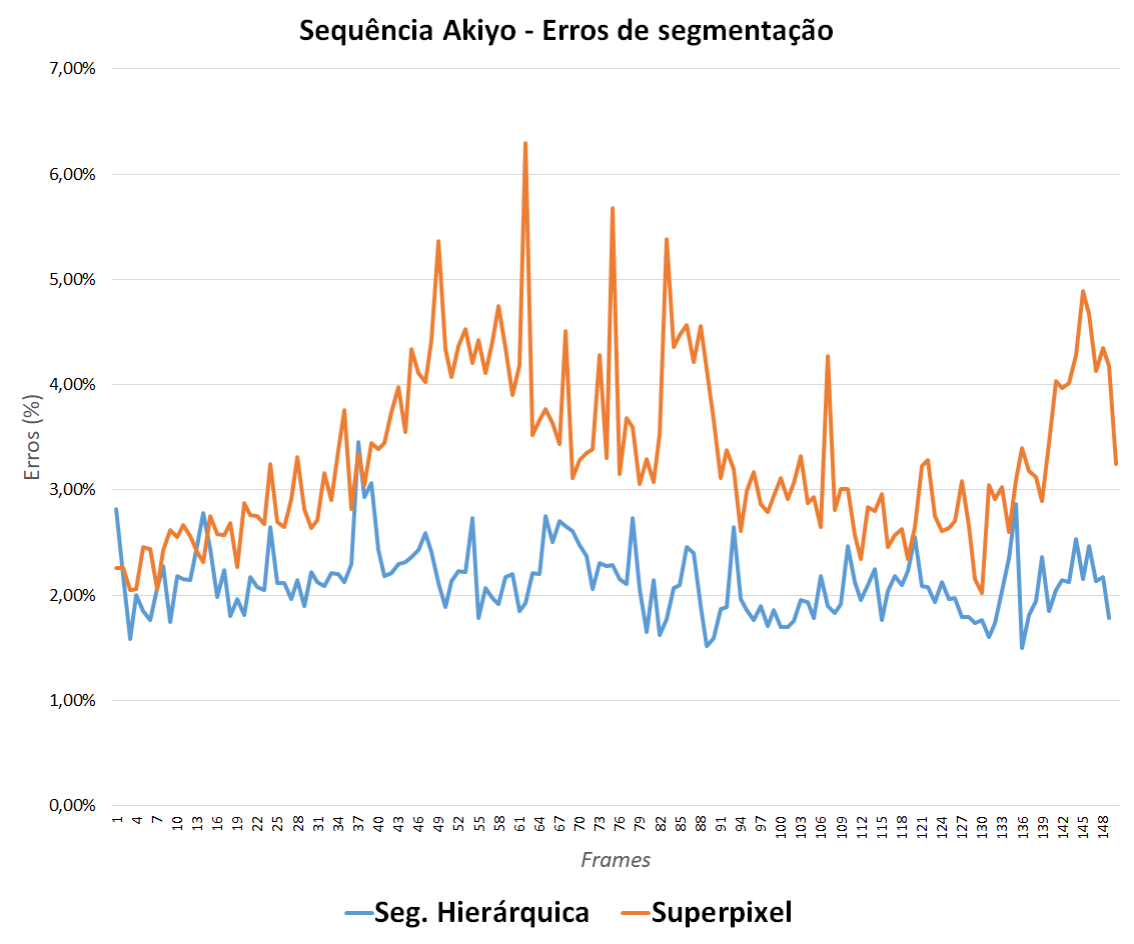


Figura 5.16: Erro de segmentação por *frame* para sequência *Akiyo*.

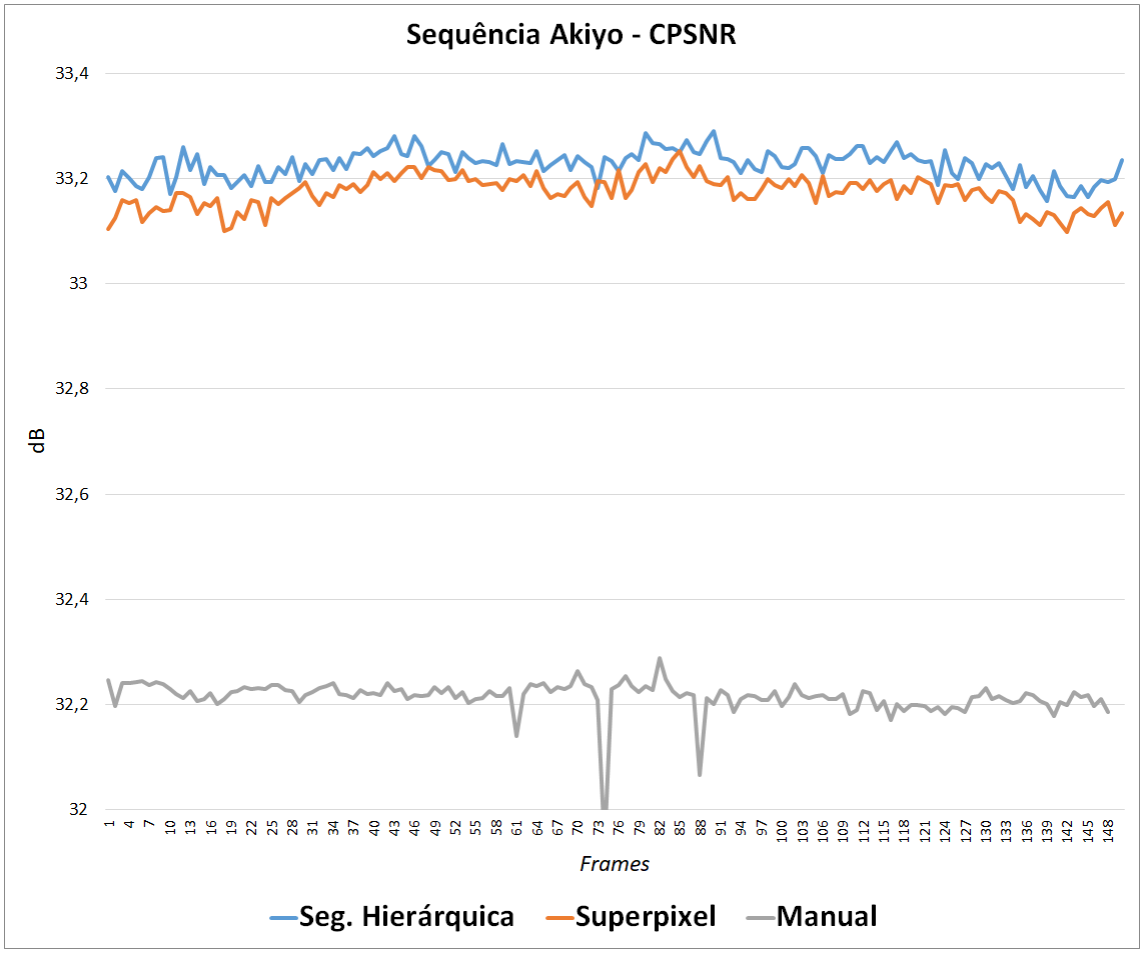


Figura 5.17: Valor do CPSNR por *frame* para sequência *Akiyo*.

5.7.3 Sequência Carphone

Para a sequência *Carphone*, o resultado do processo de colorização pode ser visto na Figura 5.18 para segmentação hierárquica e na Figura 5.19 para superpixel. E novamente foram selecionados somente metade dos 150 *frames* para uma melhor visualização. Os gráficos de Figura 5.20, Figura 5.21 e Figura 5.23 mostram os valores do tempo de colorização, quantidade de interferência do usuário e CPSNR respectivamente, tanto para o método proposto quanto para o manual. A Figura 5.22 mostra os percentuais de erros de segmentação para cada *frame*.

Na sequência Carphone, vemos comportamento similar as das sequências Foreman e Akiyo. A diferença de tempo entre o método proposto e o manual foi novamente considerável. Entre as implementações do método proposto ambas apresentaram tempos similares, com a segmentação por superpixel sendo ligeiramente melhor, na média..

Na quantidade de interferência, novamente, o método proposto mostrou valores menores do que o método manual. Entre as implementações do método proposto, a implementação usando segmentação hierárquica foi ligeiramente melhor, para esta cena.

Para o erro de segmentação ambas implementações apresentaram quantitativos similares, pode-se destacar que a segmentação por superpixel apresentou uma oscilação maior, em casos particulares ultrapassando os 3,50% de erro.

Para a métrica de CPSNR, obtivemos resultados similares aos da sequência de Foreman. O resultado do método manual foi inferior ao do método proposto, porém houve uma diferença considerável entre as implementações do método proposto. A implementação via segmentação hierárquica apresentou um resultado superior em relação a implementação com superpixel.

Visualmente o resultado é satisfatório. Detalhes da roupa, do carro e da vegetação ao fundo estão devidamente coloridos e com uma aparência natural.

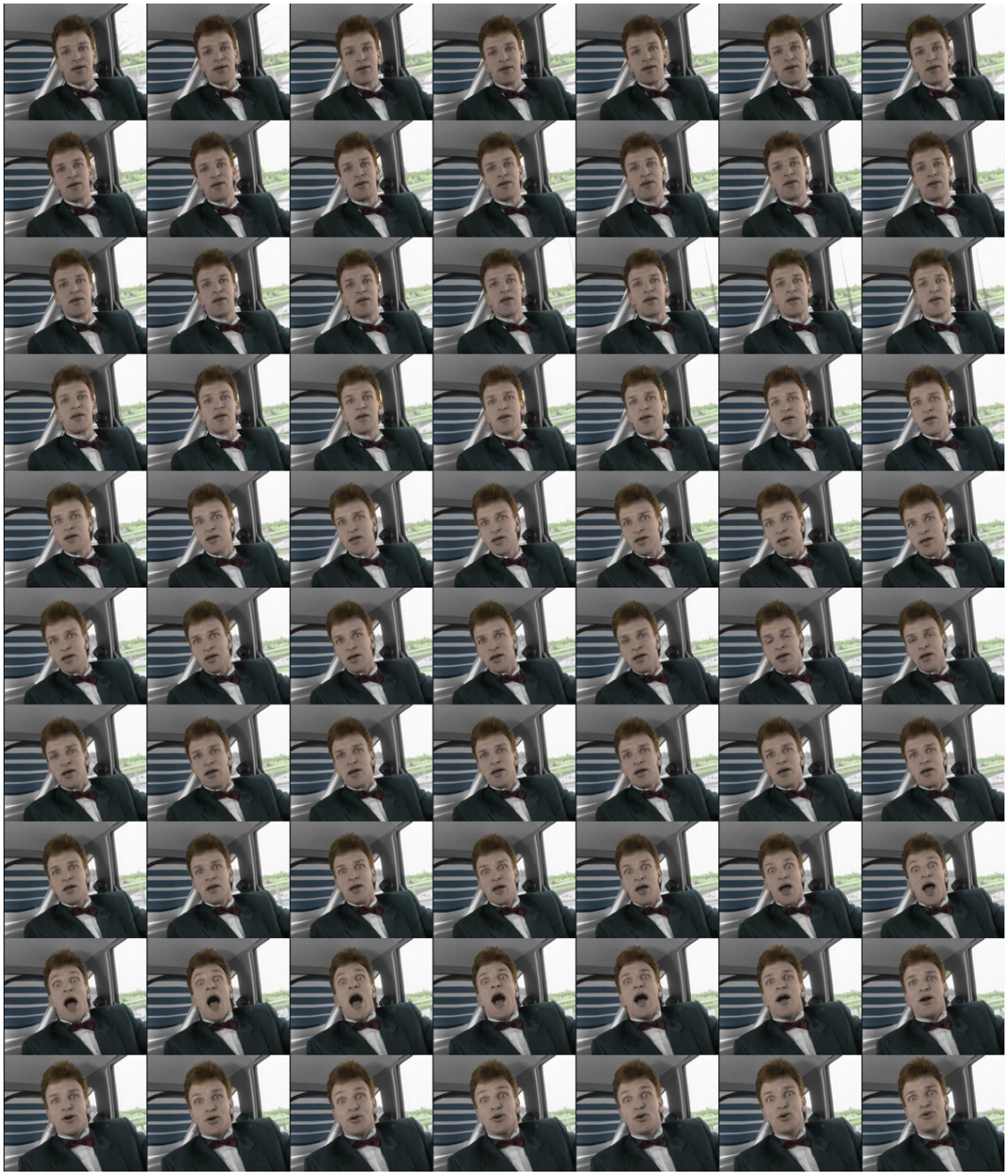


Figura 5.18: Resultado da colorização para sequência *Carphone*, segmentação hierárquica.

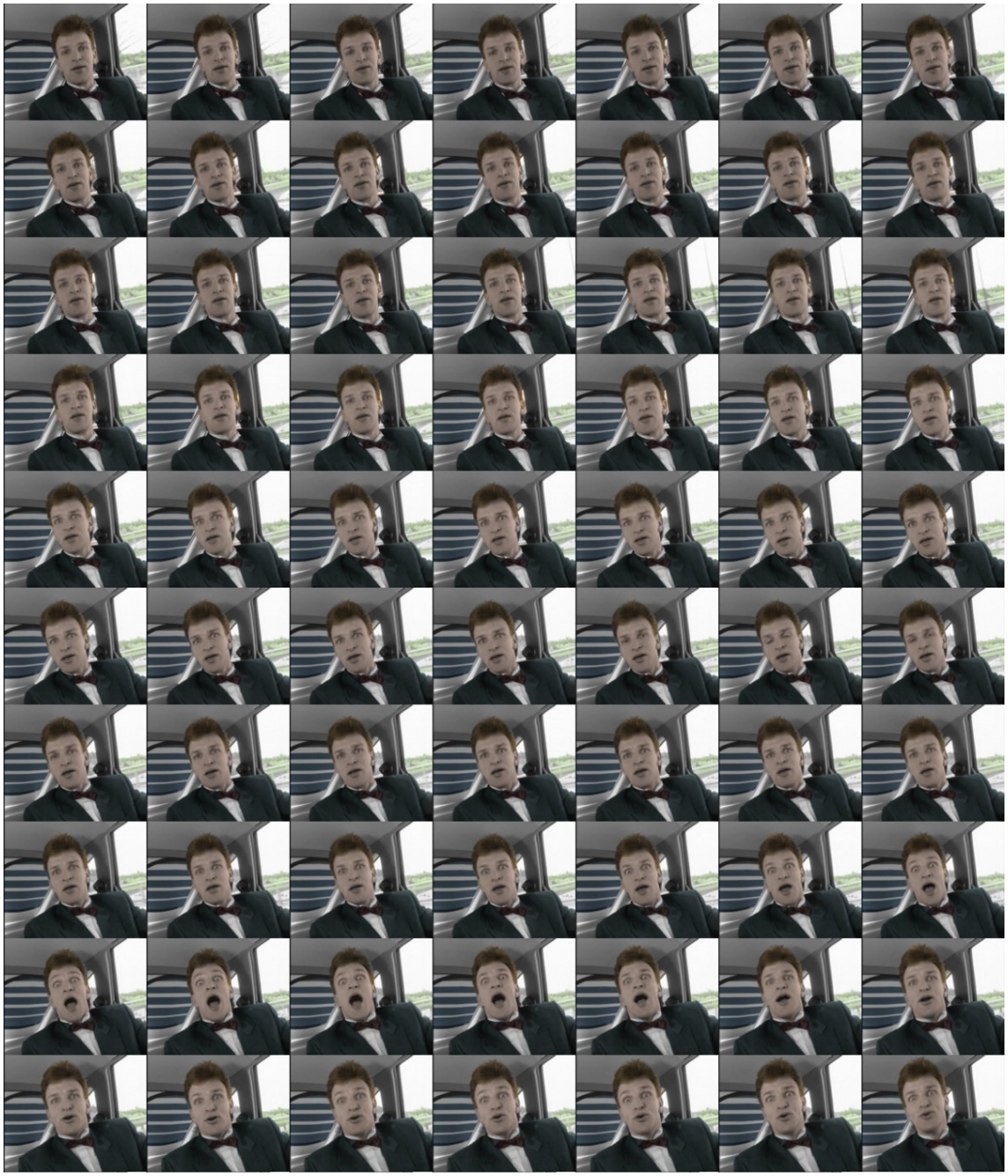


Figura 5.19: Resultado da colorização para sequência *Carphone*, segmentação por superpixel.

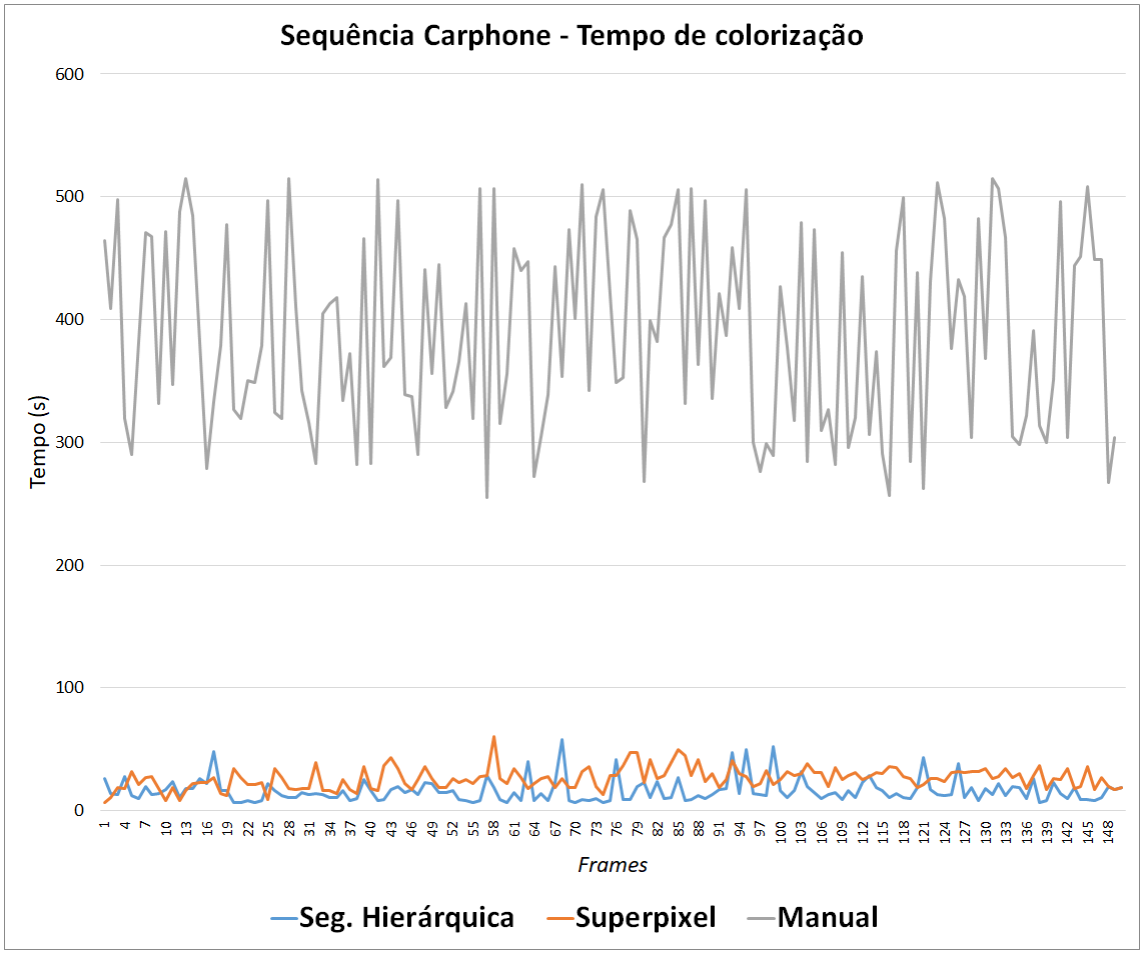


Figura 5.20: Tempo de colorização por *frame* para sequência *Carphone*.

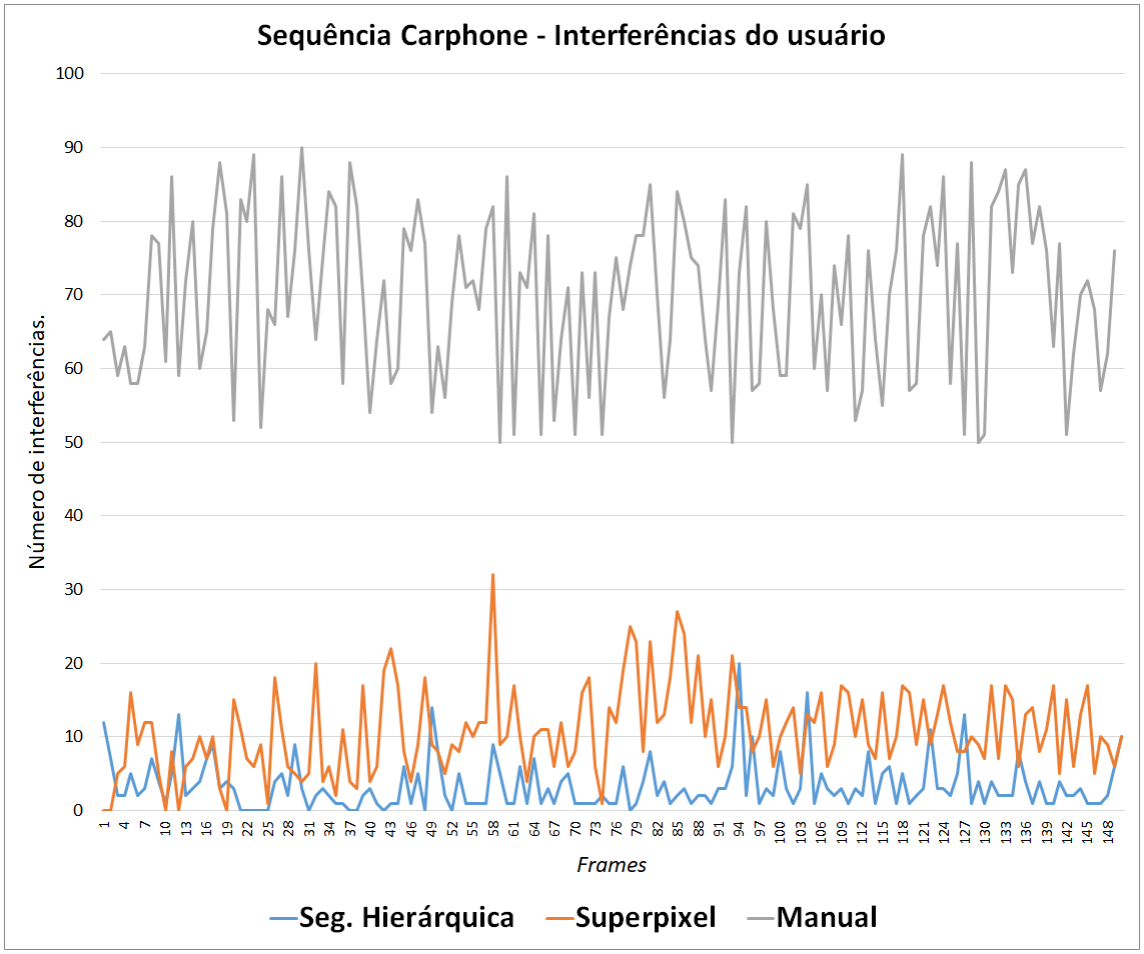


Figura 5.21: Interferências do usuário por *frame* para sequência *Carphone*.

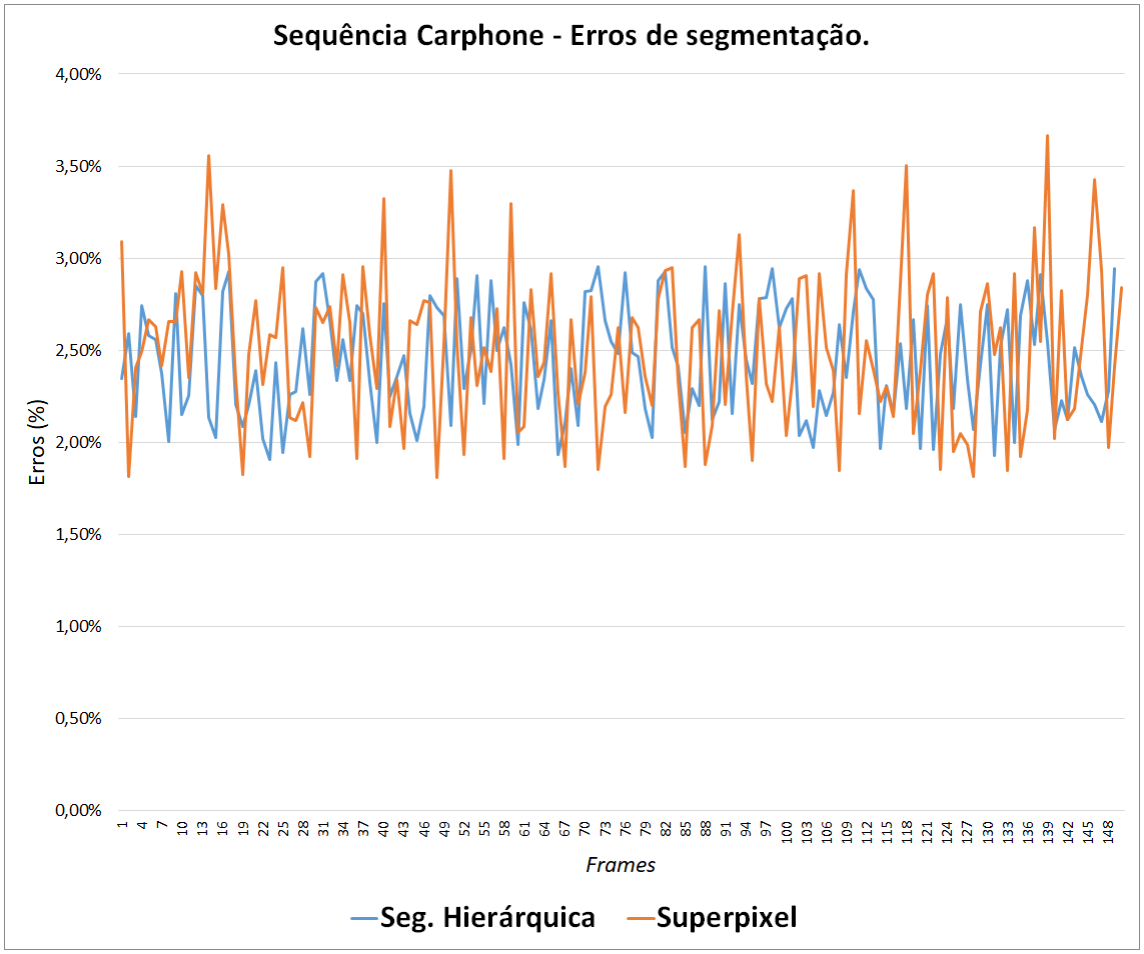


Figura 5.22: Erro de segmentação por *frame* para sequência *Carphone*.

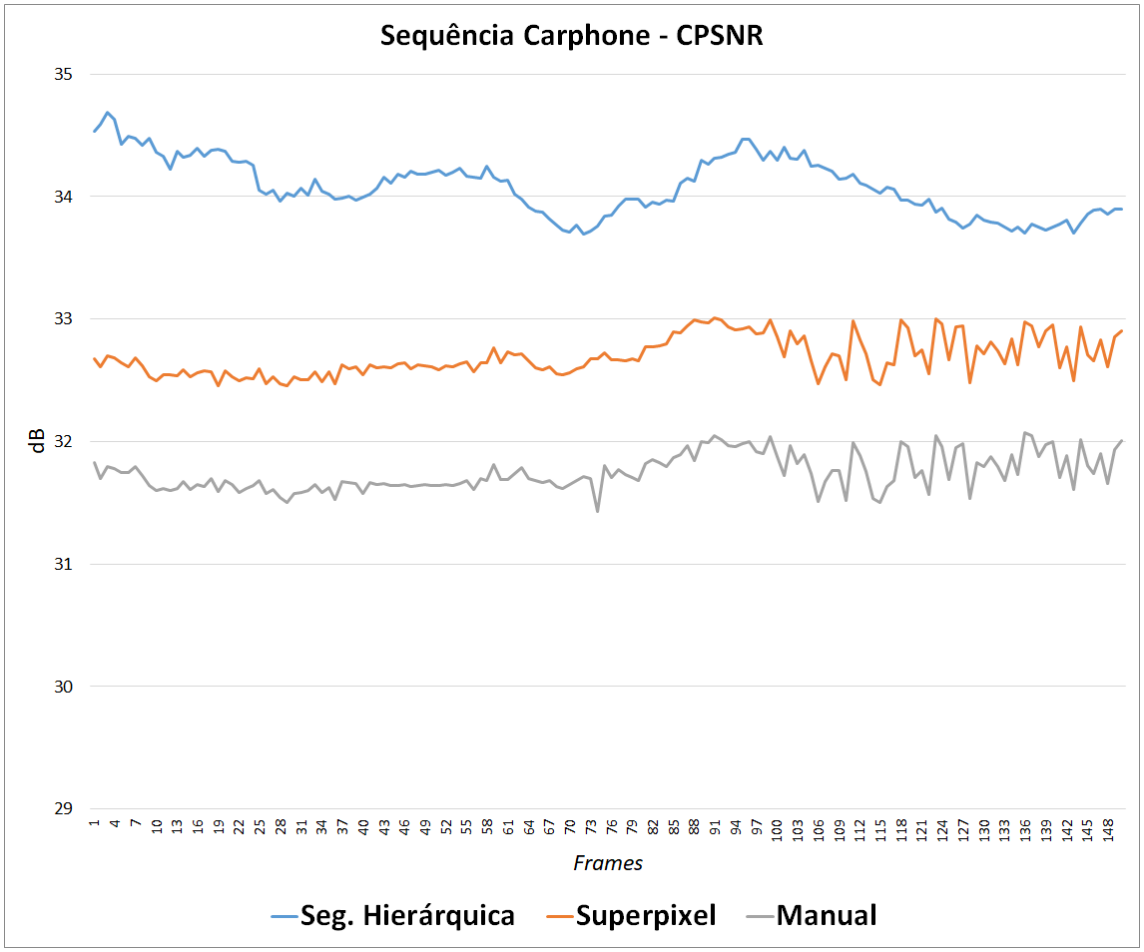


Figura 5.23: Valor do CPSNR por *frame* para sequência *Carphone*.

5.7.4 Resumo dos experimentos

A Tabela 5.4 mostra a média para cada métrica dos valores obtidos nos experimentos utilizando o método manual. A Tabela 5.5 mostra os resultados utilizando o método proposto com o uso do algoritmo de superpixel, A Tabela 5.6 demonstra o resultado do método proposto com a utilização de segmentação hierárquica. Como se pode observar, em todos os casos o tempo médio necessário para realizar a colorização foi inferior ao método manual, pois o usuário só precisou corrigir a cor de algumas regiões por *frame* não sendo necessário repetir toda a segmentação. O volume de interferências também se mostrou expressivamente menor no método proposto, devido a pouca necessidade de correção em cada *frame*. O CPSNR foi superior para o método proposto, demonstrando que o resultado visual da colorização foi superior ao método manual, havendo mais similaridade com a imagem original. Isso se deve principalmente ao fato de ser usado um algoritmo de otimização para realizar a distribuição das cores, fazendo de forma suave, enquanto no

manual ocorre um contraste grande de crominância entre as regiões o que muitas vezes não ocorre em imagens naturais.

Comparando-se os dois métodos de segmentação, a segmentação hierárquica apresentou resultados ligeiramente melhores do que a segmentação utilizando o algoritmo de superpixel. A segmentação hierárquica aparenta conseguir realizar um melhor particionamento dos objetos da imagem, provavelmente por causa da utilização do algoritmo de Watershed, que leva em consideração a informação topográfica da imagem para segmentar, em vez de partir de uma grade pré-definida, como no superpixel.

Tabela 5.4: Média dos valores obtidos nos experimentos com as três sequências utilizando o método manual.

	Método manual		
	Tempo	Interferências	CPSNR
Foreman	317,00 (s)	49,92	29,36 (dB)
Akiyo	241,29 (s)	41,08	32,21 (dB)
Carphone	390,00 (s)	70,01	31,75 (dB)

Tabela 5.5: Média dos valores obtidos nos experimentos com as três sequências utilizando o método proposto com segmentação por superpixel.

	Método proposto - superpixel			
	Tempo	Interferências	CPSNR	Erro de segmentação
Foreman	34,03 (s)	7,60	31,93 (dB)	3,34%
Akiyo	33,43 (s)	7,22	33,17 (dB)	3,31%
Carphone	26,16 (s)	10,77	32,69 (dB)	2,50%

Tabela 5.6: Média dos valores obtidos nos experimentos com as três sequências utilizando o método proposto com segmentação hierárquica via Watershed.

	Método proposto - hierárquica			
	Tempo	Interferências	CPSNR	Erro de segmentação
Foreman	23,95 (s)	7,00	33,63 (dB)	2,26%
Akiyo	28,68 (s)	3,01	33,22 (dB)	2,12%
Carphone	16,28 (s)	3,39	34,08 (dB)	2,44%

5.8 Comparação com Métodos do Estado da Arte

Nesta seção é apresentada a comparação do método proposto com outros métodos existentes na literatura, especialmente com métodos mais recentes que representam o estado da arte das técnicas de colorização de imagem e vídeos digitais.

5.8.1 Métodos Selecionados

Foram selecionados cinco métodos da literatura que utilizam de diversas técnicas para realizar a colorização, sendo eles os trabalhos de: Levin et al. (2004), Yatziv e Sapiro (2006), Gupta et al. (2017), Larsson et al. (2016) e Zhang et al. (2017).

Os experimentos foram realizados no mesmo ambiente já mencionado na Seção 5.1. Para os métodos propostos por Levin et al. (2004) e Yatziv e Sapiro (2006) foram realizadas implementações utilizando a linguagem Python. Para os demais métodos foram utilizados códigos fornecidos pelos próprios autores.

Como cada método utiliza diferentes formas para realizar a colorização, foram tomados alguns cuidados para tornar a comparação mais justa possível.

Para os métodos propostos em Levin et al. (2004); Yatziv e Sapiro (2006), foi utilizada a mesma paleta de cor utilizada no método deste trabalho, a fim de manter as cores mais próximas possíveis para efeitos de comparação. Foi computado como o número de interferências o total de vezes que o usuário adicionou e removeu rabiscos da imagem. Como sugerido pelos autores, foi utilizado um algoritmo de fluxo ótico para propagar esses rabiscos para os *frames* seguintes.

Para o método proposto por Gupta et al. (2012), como o mesmo utiliza uma imagem de referência para colorir, não é possível aplicar a métrica de interferências do usuário. Para manter uma comparação justa, para cada sequência, foi utilizada como imagem de referência o primeiro *frame* já previamente colorido . Da mesma forma que ocorre no método anterior, no método proposto por Larsson et al. (2016) não é possível computar as interferências do usuário pois o método é totalmente automático. Da mesma forma, o método não permite sugerir ou definir uma paleta de cores.

Os resultados numéricos dos experimentos podem ser vistos nas Tabelas 5.7, 5.8 e 5.9. As figuras 5.24, 5.25, 5.26, 5.27, 5.28, 5.29 e 5.30 mostram o resultado visual de cada uma das técnicas para o *frame* 100 de cada uma das sequências. Importante salientar que nenhuma interferência foi feita no *frame* 100. Esse *frame* foi escolhido porque os métodos tiveram que processar diversos *frames* para chegar ao resultado apresentado.

Como se pode observar, apesar de alguns métodos conseguirem colorizar sem interferência do usuário o resultado visual se mostra ruim e artificial, o que também reflete em um baixo valor de CPSNR. Como se pode ver ver na Tabela 5.8, onde o método automático proposto por Larsson et al. (2016) teve baixo desempenho. Métodos como os propostos por Zhang et al. (2017) e Gupta et al. (2017) apesar de apresentarem um resultado visual razoável demandam muito tempo ou interferências do usuário em comparação com o método aqui proposto.

Tabela 5.7: Tempo médio por *frame* (s).

	Sequências		
Método	Akiyo	Foreman	Carphone
Método Proposto (Seg. hierárquica)	28,68	23,95	16,28
Método Proposto (Seg. superpixel)	33,43	34,03	26,16
Levin et al., 2004	67,47	63,7	45,54
Yatziv e Sapiro, 2006	154,74	137,78	102,45
Larsson et al., 2016	7,3	7,5	7,3
Zhang et al., 2017	275	291	254
Gupta et al., 2017	183	195	180

Tabela 5.8: CPSNR médio (dB).

	Sequências		
Método	Akiyo	Foreman	Carphone
Método Proposto (Seg. hierárquica)	33,22	33,63	34,08
Método Proposto (Seg. superpixel)	33,17	31,93	32,69
Levin et al., 2004	32,61	32,17	31,18
Yatziv e Sapiro, 2006	30,74	29,47	28,18
Larsson et al., 2016	29.28	30,41	29,58
Zhang et al., 2017	31,78	31,51	31,03
Gupta et al., 2017	32,46	31,63	31,71

Tabela 5.9: Média de interferências do usuário.

	Sequências		
Método	Akiyo	Foreman	Carphone
Método Proposto (Seg. hierárquica)	3,01	7,00	3,39
Método Proposto (Seg. superpixel)	7,22	7,61	10,77
Levin et al., 2004	11,78	12,72	8,89
Yatziv e Sapiro, 2006	17,97	16,87	13,20
Larsson et al., 2016	*	*	*
Zhang et al., 2017	46,94	51,54	37,14
Gupta et al., 2017	*	*	*

* não se aplica para o método.

Figura 5.24: Resultado da colorização pelo método proposto com segmentação hierárquica *frame* 100.

Figura 5.25: Resultado da colorização pelo método proposto com superpixel *frame* 100.

Figura 5.26: Resultado da colorização pelo método de Levin et al., 2004 *frame* 100.

Figura 5.27: Resultado da colorização pelo método de Yatziv e Sapiro, 2006 *frame* 100.

Figura 5.28: Resultado da colorização pelo método de Larsson et al., 2016 *frame* 100.

Figura 5.29: Resultado da colorização pelo método de Zhang et al., 2017 *frame* 100.

Figura 5.30: Resultado da colorização pelo método de Gupta et al., 2017 *frame* 100.

6

Conclusão

Este trabalho propôs um método assistido para colorização de sequências de imagens em tons de cinza, de forma a reduzir o tempo e quantidade de trabalho necessário.

O método proposto utilizou descritores de textura e extração de característica da intensidade de cada *frame* para conseguir transferir cores de um *frame* para o próximo. Também é dada ao usuário, por meio de uma ferramenta, a possibilidade de corrigir eventuais erros na designação das cores. Foi utilizado um algoritmo de otimização para efetuar a distribuição das cores de forma a criar um resultado mais natural.

Os experimentos realizados mostraram que o método conseguiu reduzir o tempo necessário para colorir cada *frame* de minutos para apenas alguns segundos. A quantidade de esforço exigido do usuário também foi reduzida drasticamente, sendo que, graças à taxa de acerto, é necessária a correção de apenas algumas regiões para alcançar o resultado desejado.

Em experimentos realizados com métodos do estado da arte, presentes na literatura, constatou-se que o método proposto conseguiu alcançar resultados, tanto quantitativos como qualitativos, superiores aos métodos comparados, conseguindo conciliar um resultado visual bom, com taxas de interferências do usuário e tempo por *frame* baixas, de acordo com o *benchmark* utilizado que abrange várias métricas. Ao se realizar uma comparação entre as variações do método proposto com a utilização de segmentação por superpixel e de segmentação hierárquica com o uso de Watershed, pode-se verificar que a que utiliza segmentação hierárquica apresentou resultados ligeiramente melhores do que a com utilização de superpixel para segmentação.

6.1 Contribuições Principais

Uma das contribuições principais deste trabalho é a proposta de um método para a colorização de sequências de imagens em escala de cinza. Esse método promoveu redução no tempo necessário para a realização de tal tarefa, além de ter diminuído o esforço do usuário bem como apresentou resultados melhores que vários métodos encontrados na literatura. Além disso, o método pode servir de referência para outros pesquisadores na busca de soluções do problema de colorização de sequência de imagens ou ainda servir de base para implementação de ferramentas em outras linguagens.

Outra contribuição principal é a proposta de um *benchmark* para avaliação de métodos interativos de colorização de sequências de imagens. Futuramente pesquisadores poderão usar as métricas propostas para realizar comparações quantitativas e qualitativas entre os métodos de colorização.

6.2 Dificuldades encontradas

Apesar de ser um problema bem conhecido não há na literatura métricas consolidadas para uma avaliação quantitativa dos resultados da colorização. A maioria dos trabalhos utilizam a comparação visual do leitor para avaliar o resultado, o que acarreta em uma avaliação subjetiva, ou então se baseiam na distorção em relação à uma imagem já colorida. Essa falta de métricas motivou a criação do benchmark supracitado. Também houve dificuldade para encontrar sequências com Padrão Ouro para serem utilizadas como referência, uma vez que a maioria dos existentes se baseiam em separar objeto de fundo. Com isso, foi necessária a criação de padrões manuais o que demandou uma quantidade enorme de esforço e tempo.

6.3 Trabalhos futuros

Como trabalhos futuros, pode-se destacar a possibilidade do estudo e uso de outros descritores de textura que possam eventualmente apresentar mais discriminação, bem como outros tipos de características. Também pode-se estudar outros tipos de segmentação automática, para se conseguir regiões mais relevantes que consigam dividir de forma mais eficiente a imagem e, por conseguinte, reduzir o custo computacional e melhorar os resultados, bem como experimentos com outros algoritmos de segmentação e outros parâmetros de controle de super-segmentação.

Sugere-se também a construção de mais sequências de imagens rotuladas pelos objetos e cores que servirão como Padrão Ouro para sequências.

REFERÊNCIAS

Achanta, R.; Shaji, A.; Smith, K.; Lucchi, A.; Fua, P.; Süsstrunk, S. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE transactions on pattern analysis and machine intelligence*, v. 34, n. 11, p. 2274–2282, 2012.

Ahonen, T.; Hadid, A.; Pietikainen, M. Face description with local binary patterns: Application to face recognition. *IEEE transactions on pattern analysis and machine intelligence*, v. 28, n. 12, p. 2037–2041, 2006.

A Trip to the Moon, Direção: Georges Méliès. França: Star Film Company, 1902. 1 Filme (18 min,), mono. .

Ball, J. A. The technicolor process of three-color cinematography. *Journal of the Society of Motion Picture Engineers*, v. 25, n. 2, p. 127–138, 1935.

Behnel, S.; Bradshaw, R.; Citro, C.; Dalcin, L.; Seljebotn, D. S.; Smith, K. Cython: The best of both worlds. *Computing in Science & Engineering*, v. 13, n. 2, p. 31–39, 2011.

Van den Bergh, M.; Boix, X.; Roig, G.; de Capitani, B.; Van Gool, L. Seeds: Superpixels extracted via energy-driven sampling. In: *European conference on computer vision*, Springer, 2012, p. 13–26.

Beucher, N.; Beucher, S. Hierarchical queues: general description and implementation in mamba image library. 2011.

Beucher, S. Use of watersheds in contour detection. In: *Proceedings of the International Workshop on Image Processing*, CCETT, 1979.

Beucher, S.; Meyer, F. The morphological approach to segmentation: the watershed transformation. *Optical Engineering-New York-Marcel Dekker Incorporated-*, v. 34, p. 472–479, 1992.

Bharati, M. H.; Liu, J. J.; MacGregor, J. F. Image texture analysis: methods and comparisons. *Chemometrics and intelligent laboratory systems*, v. 72, n. 1, p. 57–71, 2004.

Bradski, G.; Kaehler, A. Opencv. *Dr. Dobb's journal of software tools*, 2000.

Bugeau, A.; Ta, V.-T.; Papadakis, N. Variational exemplar-based image colorization. *IEEE Transactions on Image Processing*, v. 23, n. 1, p. 298–307, 2014.

Cheng, Z.; Yang, Q.; Sheng, B. Deep colorization. In: *Proceedings of the IEEE International Conference on Computer Vision*, 2015, p. 415–423.

Daugman, J. G. Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters. *JOSA A*, v. 2, n. 7, p. 1160–1169, 1985.

Digabel, H.; Lantuéjoul, C. Iterative algorithms. *European Symp. Quantitative Analusis of Microstrctures in Material Sciente, Biology and Medicine*, v. 2, n. 2, p. 85–99, 1977.

Dougherty, E. *Mathematical morphology in image processing* CRC press, p. 433–481, 1992.

Duval, G.; Wemaere, S. *La couleur retrouvée du voyage dans la lune.* Fondation Groupama GAN pour le cinéma, 2011.

Flores, F. C.; de Alencar Lotufo, R. A new hierarchical decomposition applied to object segmentation in image sequences: the uniform decomposition. In: *Graphics, Patterns and Images (SIBGRAPI), 2010 23rd SIBGRAPI Conference on*, IEEE, 2010, p. 156–163.

Flores, F. C.; de Alencar Lotufo, R.; et al. Benchmark for quantitative evaluation of assisted object segmentation methods to image sequences. In: *XXI Brazilian Symposium on Computer Graphics and Image Processing*, IEEE, 2008, p. 95–102.

Gonzalez, R. C.; Woods, R. E. Image processing. *Digital image processing*, v. 2, 2007.

Gupta, R. K.; Chia, A. Y.-S.; Rajan, D.; Ng, E. S.; Zhiyong, H. Image colorization using similar images. In: *Proceedings of the 20th ACM international conference on Multimedia*, ACM, 2012, p. 369–378.

Gupta, R. K.; Chia, A. Y.-S.; Rajan, D.; Zhiyong, H. A learning-based approach for automatic image and video colorization. *arXiv preprint arXiv:1704.04610*, 2017.

Iizuka, S.; Simo-Serra, E.; Ishikawa, H. Let there be color!: joint end-to-end learning of global and local image priors for automatic image colorization with simultaneous classification. *ACM Transactions on Graphics (TOG)*, v. 35, n. 4, p. 110, 2016.

ISO 11664-4:2008 *COLORIMETRY — PART 4: CIE 1976 L*A*B* COLOUR SPACE*. Standard, International Commission on Illumination, Geneva, CH, 1976.

Jones, E.; Oliphant, T.; Peterson, P. {SciPy}: open source scientific tools for {Python}. 2014.

Kimmel, R. A natural norm for color processing. In: *Asian Conference on Computer Vision*, Springer, 1998, p. 88–95.

Larsson, G.; Maire, M.; Shakhnarovich, G. Learning representations for automatic colorization. In: *European Conference on Computer Vision*, Springer, 2016, p. 577–593.

Levin, A.; Lischinski, D.; Weiss, Y. Colorization using optimization. In: *ACM Transactions on Graphics (ToG)*, ACM, 2004, p. 689–694.

Li, Z.; Chen, J. Superpixel segmentation using linear spectral clustering. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, p. 1356–1363.

Luan, Q.; Wen, F.; Cohen-Or, D.; Liang, L.; Xu, Y.-Q.; Shum, H.-Y. Natural image colorization. In: *Proceedings of the 18th Eurographics conference on Rendering Techniques*, Eurographics Association, 2007, p. 309–320.

Mäenpää, T. *The local binary pattern approach to texture analysis: extensions and applications*. Oulun yliopisto Oulu, 2003.

Manjunath, B. S.; Ma, W.-Y. Texture features for browsing and retrieval of image data. *IEEE Transactions on pattern analysis and machine intelligence*, v. 18, n. 8, p. 837–842, 1996.

Meyer, F. Hierarchies of partitions and morphological segmentation. In: *International Conference on Scale-Space Theories in Computer Vision*, Springer, 2001, p. 161–182.

Ojala, T.; Pietikainen, M.; Maenpaa, T. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on pattern analysis and machine intelligence*, v. 24, n. 7, p. 971–987, 2002.

Pang, J.; Au, O. C.; Yamashita, Y.; Ling, Y.; Guo, Y.; Zeng, J. Self-similarity-based image colorization. In: *Image Processing (ICIP), 2014 IEEE International Conference on*, IEEE, 2014, p. 4687–4691.

Paul, S.; Bhattacharya, S.; Gupta, S. Spatiotemporal colorization of video using 3d steerable pyramids. *IEEE Transactions on Circuits and Systems for Video Technology*, v. 27, n. 8, p. 1605–1619, 2017.

Perlibakas, V. Distance measures for pca-based face recognition. *Pattern recognition letters*, v. 25, n. 6, p. 711–724, 2004.

Pierre, F.; Aujol, J.-F.; Bugeau, A.; Ta, V.-T. Interactive video colorization within a variational framework. *SIAM Journal on Imaging Sciences*, v. 10, n. 4, p. 2293–2325, 2017.

Qu, Y.; Wong, T.-T.; Heng, P.-A. Manga colorization. In: *ACM Transactions on Graphics (TOG)*, ACM, 2006, p. 1214–1220.

Silva, A. G.; de Alencar Lotufo, R. New extinction values from efficient construction and analysis of extended attribute component tree. In: *Computer Graphics and Image Processing, 2008. SIBGRAPI'08. XXI Brazilian Symposium on*, IEEE, 2008, p. 204–211.

Souza, R.; Rittner, L.; Machado, R.; Lotufo, R. A comparison between extinction filters and attribute filters. In: *International Symposium on Mathematical Morphology and Its Applications to Signal and Image Processing*, Springer, 2015, p. 63–74.

Souza, R.; Rittner, L.; Machado, R.; Lotufo, R. iamxt: Max-tree toolbox for image processing and analysis. *SoftwareX*, v. 6, p. 81–84, 2017.

Teng, S.; Shen, Y.; Zhao, Z.; Li, L.; Cao, M. An interactive framework for video colorization. In: *Image and Graphics (ICIG), 2013 Seventh International Conference on*, IEEE, 2013, p. 89–94.

Vachier, C. *Extraction de caractéristiques, segmentation d'image et morphologie mathématique*. Tese de Doutoramento, Ecole Nationale Supérieure des Mines de Paris, 1995.

VINCENT, L. Recent developments in morphological algorithms. *ACTA STEREOLOGICA*, v. 11, p. 521–521, 1992.

VINCENT, L.; SOILLE, P. Watersheds in digital spaces: an efficient algorithm based on immersion simulations. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, , n. 6, p. 583–598, 1991.

WALT, S. V. D.; COLBERT, S. C.; VAROQUAUX, G. The numpy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, v. 13, n. 2, p. 22–30, 2011.

XIA, S.; LIU, J.; FANG, Y.; YANG, W.; GUO, Z. Robust and automatic video colorization via multiframe reordering refinement. In: *Image Processing (ICIP), 2016 IEEE International Conference on*, IEEE, 2016, p. 4017–4021.

YANG, J.; LIU, L.; JIANG, T.; FAN, Y. A modified gabor filter design method for fingerprint image enhancement. *Pattern Recognition Letters*, v. 24, n. 12, p. 1805–1817, 2003.

YATZIV, L.; SAPIRO, G. Fast image and video colorization using chrominance blending. *IEEE transactions on image processing*, v. 15, n. 5, p. 1120–1129, 2006.

YUMIBE, J. *Moving color: Early film, mass culture, modernism.* Rutgers University Press, 2012.

ZHANG, R.; ISOLA, P.; EFROS, A. A. Colorful image colorization. In: *European Conference on Computer Vision*, Springer, 2016, p. 649–666.

ZHANG, R.; ZHU, J.-Y.; ISOLA, P.; GENG, X.; LIN, A. S.; YU, T.; EFROS, A. A. Real-time user-guided image colorization with learned deep priors. *arXiv preprint arXiv:1705.02999*, 2017.