Video Soft Colour Segmentation

by

Johanna Barbier, B.Sc.Eng.

A Dissertation

Presented to the University of Dublin, Trinity College

in partial fulfilment of the requirements for the degree of

Master of Science in Computer Science

(Computer Graphics & Vision Technologies)

Supervisor: Aljosa Smolic

Co-supervisors: Mairéad Grogan & Pierre Matysiak

September 2018

Declaration

I, the undersigned, declare that this work has not previously been submitted as an exercise for a degree at this, or any other University, and that unless otherwise stated, is my own work.

Johanna Barbier

August 29, 2018

Permission to Lend and/or Copy

I, the undersigned, agree that Trinity College Library may lend or copy this thesis upon request.

Johanna Barbier

August 29, 2018

Acknowledgments

Firstly, I would like to thank my co-supervisors Mairéad Grogan & Pierre Matysiak for their support, their feedback and for the essential weekly meetings we had during these five months. Similarly, I would like to acknowledge my supervisor Aljosa Smolic for his valuable guidance and assistance on how to evaluate the designed method.

Secondly, I want to thank my family for their strong support and continuous encouragement throughout my education. I particularly express my gratitude to my mother for proof-reading this dissertation in its final stage.

Finally, I would like to thank my friends in this master and all those I have met during my time in Dublin. I enjoyed it all, from the long studious nights to the great boxing and basketball training, thanks a lot!

Johanna Barbier

University of Dublin, Trinity College
September 2018

Video Soft Colour Segmentation

Johanna Barbier, Master of Science in Computer Science

University of Dublin, Trinity College, 2018

Supervisor: Aljosa Smolic

This dissertation explores the extension of a state-of-the-art image-based soft colour segmentation into the video domain. The objective is to design an automatic video soft colour segmentation that decomposes an input video into a set of colour layers with transparency channels. The output layers have to be spatially and temporally coherent. To answer this objective, the implemented method is divided into two parts: a layer initialisation technique using optical flow to provide temporal information, and a colour model estimation to correctly represent the current frame colour distributions. This method is evaluated on the Blender Sintel animated movie in terms of temporal smoothness, reconstruction error and time performance. While the layer initialisation technique yields unexpectedly poor results, it uncovers a transparency diffusion behaviour of the original image-based segmentation. As for the colour model estimation, fixing the colour model on detected video cuts produces temporally coherent output layers and speeds up the video segmentation process.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Nowadays, video editing software allows various colour editing such as saturation, gamma, contrast or brightness corrections. These widely used tools enable creative content over the full sequence of a video. However many of these edits are limited to the global image scope and does not allow local modifications.

Video soft colour segmentation is the process of segmenting a video into temporally consistent layers of different colours and opacities. These layers usually consist of fully opaque and fully transparent regions, as well as pixels with alpha values between 0 and 1 wherever multiple layers overlap. On each layer, the represented colours should be homogeneous, i.e. the variance of colours should stay small. Finally, when summed back together, the output layers need to accurately reconstruct the input video. When these requirements are fulfilled, it is then possible to apply per-layer modifications on the resulting layers. Thus it allows an input video to be edited locally by changing regions of similar colours in a post-processing step.

This process should enable further control over the video manipulation and allow the creation of more complex visual effects. For instance, a layer containing the hair colour could be contrasted more compared to the rest of the video elements, or the sky colour changed. In the case of an animated movie, this video segmentation and post-

processing step does not require access to the full scene's initial objects and lighting. The output video could directly be processed into video soft colour layers and then edited. To sum up, this video segmentation could be used as a post-processing step to give more freedom for video editing.

1.2 Objectives

The objectives of this thesis are to automatically extract soft colour segments from a video. These soft colour segments, also known as layers in traditional editing software, should be temporally consistent throughout the video. They should also reconstruct the original video when summed back together. This is a necessary condition to allow flawless video editing. Finally we want this video segmentation to be as fast as possible.

To satisfy these objectives, we work on the extension of the state-of-the-art paper "Unmixing-Based Soft Color Segmentation for Image Manipulation" published by Aksoy et al. in 2017 [1]. They present an image-based soft colour segmentation algorithm that efficiently decomposes an image into colour layers with transparency channels. This soft colour segmentation allows different segments of the image to be processed independently. For instance, the user can modify a layer colour or replace it by a background image (see fig. 1.1).

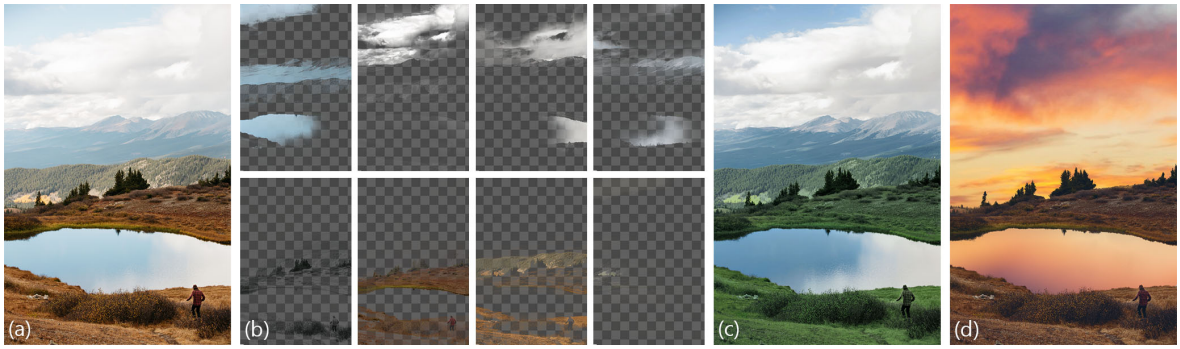

Figure 1.1: Input image (a) decomposed in soft segments (b) that can be used for image editing (c) or compositing (d).

The aforementioned paper is the foundation of our research and its method is described in section 2.1.2. Our objective is to extend this existing soft colour seg-

mentation for images to a video soft colour segmentation. The goal is to preserve the layers' colours and temporal information between frames in order to create a fast video segmentation. To this end, we design a method which is twofold: how to use optical flow information to ensure temporal smoothness in the output layers on the one hand; and how to properly set the colour model throughout the video on the other hand.

Eventually this research could be integrated into an editing software for clients from movie and visual effects industries. It could allow video editing such as colour editing: changing the sky colour to red in the first frame of the video and have it automatically spread to the remaining frames.

1.3 Plan of the Dissertation

To start with, chapter 2 reviews the state of the art in both image and video soft colour segmentation. In particular, the first section details the unmixing-based paper by Aksoy et al. [1] which is the foundation of this dissertation work. Then the video segmentation section reviews the recent competitive methods proposed for video processing and especially those using optical flow.

The design of the proposed video soft colour segmentation method is detailed in chapter 3 together with its implementation. This chapter also includes preliminary work carried out on the image-based soft colour segmentation initial code. Our method can be divided into two parts: the layer initialisation technique using optical flow on the one side and the colour model estimation on the other side.

Results for each step of the proposed method are given in chapter 4 for sequences extracted from the Blender Sintel animated movie. Each section starts by describing the evaluation process as well as the chosen metrics. The output layers are evaluated based on their temporal smoothness and on the overall time performance while the colour model estimation is evaluated depending on its time performance and how well the output layers reconstruct the original frame.

The conclusion in chapter 5 summarises the main contributions and gives possible avenues for future work.

Chapter 2

State of the Art

This chapter presents state-of-the-art works regarding image soft colour segmentation first and video soft colour segmentation second.

2.1 Image Segmentation

2.1.1 Definition

In the field of image processing, image segmentation is the process of separating an input image into different parts and is usually done at the pixel level. Each of these parts corresponds to regions or objects that have similar features or characteristics. Even though this is a fairly easy task for a human brain, intelligently recognising objects of regions is a complex task for a computer and this need has lead to the development of a broad range of methods on the segmentation of images.

Following this discussion, *colour* segmentation is the process of segmenting an input image into regions of *distinct colours*. Compositing the sub-images of each region should output the original image.

While the traditional *hard* segmentation exclusively sets every pixel as present or absent from a region, *soft* segmentation allows a pixel to be part of several regions by varying its opacity component or probability of belonging to a certain region. In the

literature *soft* segmentation can also refer to any method computing both a label as well as a confidence value for each point in the feature space. Under that definition even the K-means clustering algorithm [2] can be described as a soft segmentation method.

In this thesis, *soft segmentation* is defined as the decomposition of an image into a set of layers with alpha channels and homogeneous colours, such as in fig. 2.1. In other words every pixel can appear in different colour regions with different opacities as long as the sum of its colour value weighted with its opacity equals to the original pixel colour and an opacity of 1. This report uses interchangeably the following terms: alpha from RGBA (Red-Green-Blue-Alpha) colour representation, transparency and opacity. Soft colour segments are analogous to colour layers with alpha channels that have been commonly utilised in modern image manipulation software. These layers usually consist of fully opaque and fully transparent regions, as well as pixels with alpha values between 0 and 1 wherever multiple layers overlap.

Figure 2.1: Sintel alley2 input image and its corresponding output layers.

The following subsections first describe the soft colour segmentation technique presented by Aksoy et al. in 2017 [1] and then discuss how it compares to other state-of-the-art methods.

2.1.2 Unmixing-based Soft Colour Segmentation

Published in 2017, the paper "Unmixing-based Soft Color Segmentation for Image Manipulation" [1] proposes a competitive method producing high-quality colour segments. Therefore it was chosen as the foundation for our video soft colour segmentation. Given an input image, the presented technique starts by computing a set of prevalent colour

distributions known as the Colour Model (CM). The original image is then segmented in several layers, each layer being associated with a colour distribution from the CM. Figure 2.2 shows an example of an input image, its corresponding CM, as well as the layers after segmentation.

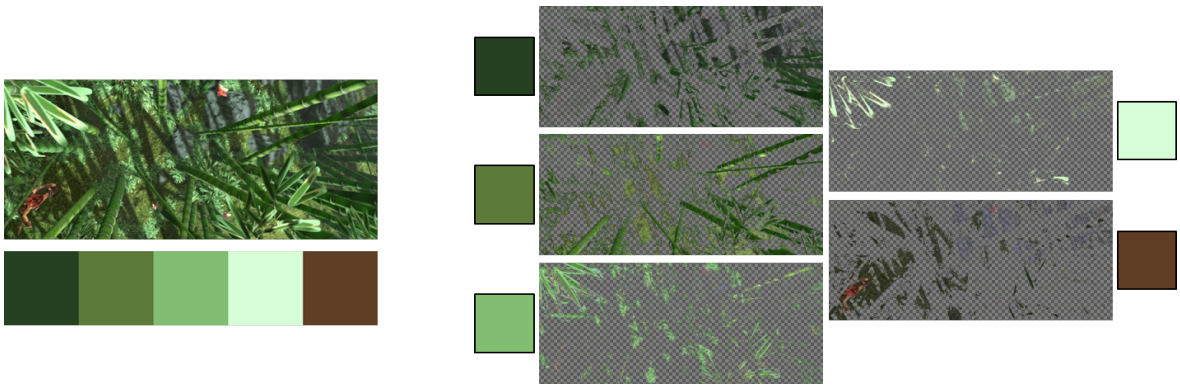

Figure 2.2: Left: Input image and associated Colour Model means. Right: Output layers corresponding to the colour distributions.

This subsection first explains the colour model estimation method and then describes the soft colour segmentation technique. The colour segmentation can be divided into three subsequent steps described below: Colour Unmixing, Matte Regularisation and Colour Refinement. During this segmentation process, a set of constraints needs to be satisfied. References from other related works are also given.

Colour Model Estimation

The colour model gives an approximate representation of the colours in an input image. It is made up of N normal distributions $\mathcal{N}(\mu_i, \sigma_i)_{i=1,..N}$. Each normal distribution is associated with a layer, with μ_i determining the mean colour of the i^{th} layer, and σ_i the variance of colours in this i^{th} layer. The size of the CM and the colour distributions are automatically estimated using the input image. The method consists in dividing the RGB colour space into 10*10*10 bins. Every pixel from the input image is then going to vote for its own bin depending on how well it is already represented by the current -possibly incomplete- colour model. This is done by calculating a representation score r^p for each pixel p as follows:

$$r^p = min_i(\{\mathcal{D}_i(\mathbf{u}_i), \forall i\} \cup \{\mathcal{F}_{i,j}, \forall i, \forall j \neq i\}), \tag{2.1}$$

where $\mathcal{D}_i(\mathbf{u}_i)$ is the Mahalanobis distance between $\mathbf{u}_i$, the RGB colour of the pixel p in the i^{th} layer, and the current i^{th} layer colour distribution $\mathcal{N}(\mu_i, \sigma_i)$. $\mathcal{F}_{i,j}$ is a simplified version of the colour unmixing function (equation 2.3) in the case of a two-colour mixture. The first term of this equation represents how well the current pixel fits a single colour distribution from the colour model, while the second term shows how well the pixel can be represented as a mixture of two colour distributions in the colour model.

After computing the representation score of every pixel, the RGB bin with the most under-represented pixels is selected. For every pixel in this bin, the algorithm looks back at its 20x20 pixel neighbourhood in the input image. The next seed pixel is the one surrounded by the most other pixels from this same under-represented bin. Then the colour mean μ_i and covariance σ_i in that 20x20 neighbourhood of pixels from the selected bin are computed. The resulting colour distribution $\mathcal{N}(\mu_i, \sigma_i)$ is added to the colour model. This process is re-iterated until the majority of pixels in the input frame are correctly represented.

Constraints

A crucial property of soft colour segmentation is for the resulting layers to sum up to the original image. This allows the editing of individual layers without degrading the final output. In mathematical terms, we want every pixel to satisfy the colour, alpha and box constraints as follows:

$$\sum_i \alpha_i \mathbf{u}_i = \mathbf{c}, \qquad \sum_i \alpha_i = 1, \qquad \textit{and} \qquad \alpha_i, \mathbf{u}_i \in [0, 1], \tag{2.2}$$

where, for the current pixel, α_i denotes the opacity value for the i^{th} layer, u_i the layer colour in RGB for the i^{th} layer and c the original colour of the pixel in the input image.

In equation (2.2), the *colour* constraint states that we should obtain the original

colour c of a pixel when we mix the underlying colours u_i using the corresponding alpha values α_i; the *alpha* constraint states that the alpha values α_i should sum up to unity assuming the original image is fully opaque; and the *box* constraint limits the range of possible alpha and colour values.

Colour Unmixing

In this first step, [1] proposes a novel energy formulation "Sparse Colour Unmixing" (SCU) which upgrades the energy function introduced by Aksoy et al. for green-screen keying in 2016 [3]. SCU decomposes the input image into soft compact layers of homogeneous colours by unmixing the colour of each pixel independently. It produces compact colour segments by favouring fully opaque or transparent pixels. This compactness is the principal upgrade compared to green-screen keying. To this end, each layer is associated with a colour distribution from the pre-computed colour model.

The sparse energy function to be minimised for each pixel is expressed as follows:

$$\mathcal{F}_s = \sum_i \alpha_i \mathcal{D}_i(\mathbf{u}_i) + s\left(\frac{\sum_i \alpha_i}{\sum_i \alpha_i^2} - 1\right), \tag{2.3}$$

where α_i is the opacity value of the pixel in the i^{th} layer, $\mathcal{D}_i(\mathbf{u}_i)$ is the Mahalanobis distance between $\mathbf{u}_i$, the RGB colour of the pixel in the i^{th} layer, and the current i^{th} layer colour distribution $\mathcal{N}(\mu_i, \sigma_i)$. μ_i is the mean and σ_i the variance of the colour distribution $\mathcal{N}$. s is the sparsity weight and is set to 10 in the implementation at this step.

The minimisation of this $\mathcal{F}_s$ function is used to estimate the alpha values α_i and the RGB colour $\mathbf{u}_i$ while respecting the constraints using the *method of multipliers* introduced by Bertsekas in 1982 [4]. The used algorithm is well explained in section 3.1 of the green-screen keying paper [3]. The Mahalanobis distance penalises RGB colour that are too far from the current layer mean colour and thus imposes homogeneous colour layers. The first term of this equation is similar to the energy function found in green-screen keying [3]. However the second term called the sparsity term is a new one and enforces compact layers, i.e. layers with opacity values being either 0 or 1. This prevents pixels to be present in too many layers with small alpha values which

was happening with the green-screen keying method.

Matte Regularisation

Afterwards the resulting layers have their opacity channels spatially filtered in order to provide spatial coherency, hence smoothness. Indeed as the colour unmixing minimisation is done at the pixel level independently, spatial coherency needs to be taken into account at this point. The pixel-based minimisation alone may result in sudden changes in opacities that do not quite agree with the underlying image textures, as shown in fig. 2.3(b). Hence, spatial regularisation of the opacity channels is necessary for ensuring smooth layers as in fig. 2.3(c).

This matte regularisation is usually done using the matting Laplacian introduced by Levin et al. [2008] [5] combined with the linear system solved by Gastal and Oliveira [2010] [6] to include the spatially non-coherent alpha values. Even though it regularises matte very effectively, this method is computationally and memory expensive which is why the guided filter proposed by He et al. [2013] [7] is used instead. This guided filter is an edge-aware filtering technique that uses a guide image to extract the edge characteristics and then filters a second image using the edge information from the guide image. The edges are extracted using the texture information of the guide image.

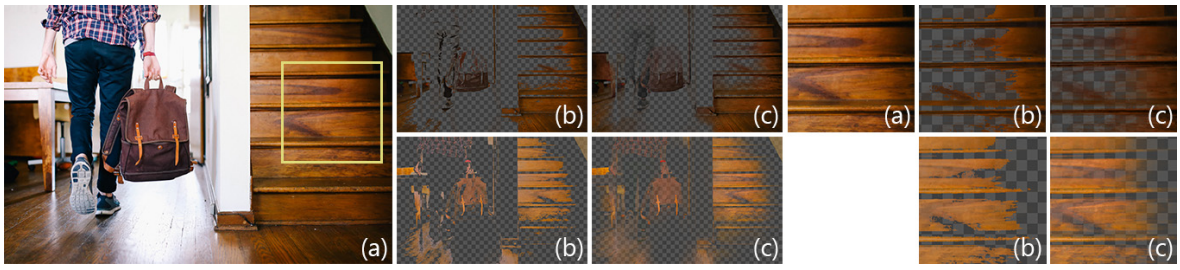

Figure 2.3: Two layers corresponding to the dark (top) and light (bottom) wood colour in the original image (a) are shown before (b) and after (c) matte regularisation and colour refinement.

He et al. demonstrates that satisfactory alpha mattes are obtained when the guided filter is applied with the original image as the guide image. As these alpha mattes are already properly minimised through the sparse colour unmixing step, the guided filter provides a sufficient quality approximation of the matting Laplacian behaviour.

However, the filtered opacity channels might no longer add up to 1 with this method, therefore Aksoy et al. normalise the sum of alpha values for each pixel after filtering to get rid of small deviations from the alpha constraint. This normalisation alters the RGB colours which no longer respect the colour constraint.

Colour Refinement

The colour refinement aims at correcting the layers' artefacts potentially created in the matte regularisation step. The previous step only filtered the alpha values and this step refines the RGB colour values while taking into account the regularised alpha. In order to do so, the energy function (2.3) is minimised once again with the following modified alpha constraint:

$$\sum_i (\alpha_i - \hat{\alpha}_i)^2 = 0, \tag{2.4}$$

which ensures that the output alpha α_i is close to the previously regularised alpha $\hat{\alpha}_i$ for the i^{th} layer. This step does not require the sparsity term of the energy function (2.3) as the regularised alpha values have to be retained and need no modification; consequently s is set to 0 in this step. Eventually the unmixed colours at all layers are re-estimated so that they satisfy the colour constraint while retaining spatial coherency of the alpha channel.

Summary

Aksoy et al. [1] starts by automatically estimating the size and colour distributions of the colour model from the input image. Each of this distribution is associated with a layer for the soft segmentation algorithm. The soft segmentation process starts by minimising the SCU energy function (2.3) for every pixel independently. Afterwards the resulting layers have their opacity channels spatially filtered in order to provide spatial coherency. Finally the energy minimisation is run once again but with an augmented alpha constraint to ensure that the basic colour, alpha and box constraints potentially overridden in the previous regularisation step are correctly satisfied. Hence, resulting soft segments satisfy the fundamental colour, alpha, and box constraints, as

well as the matte sparsity and spatial coherency requirements for high-quality soft-segmentation. Furthermore, as the two energy minimisation steps are performed for each pixel independently, this algorithm is also highly scalable and parallelisable. For a full description, please refer to the original paper [1].

2.1.3 Related Works

This subsection discusses other state-of-the-art methods in soft segmentation, and how the unmixing-based soft colour segmentation by Aksoy et al. in 2017 compares to them. They can be categorised into unmixing-based and affinity-based methods.

Unmixing-based methods such as [8], [3], [9] and the one presented previously [1] process the input colour of each pixel given a statistical representation for the layers in order to turn it into a set of RGB colours with corresponding opacities.

In Tai et al. [2007] [8], the problem formulation and constraints are similar to [1], but the energy function is alternatively optimised between the colour values, alpha values and colour model. Though converging to a good optimal solution, the alternation process tends to have layers oscillating between two solutions as the iterations progress. Eventually this interdependence can lead to unnatural gradient in layer alphas.

Tan et al. [2016] [9] use a different approach by fixing the layer colours beforehand and optimising only the opacity values. They start by extracting the colour model from the convex hull formed by the input image pixels in RGB space. The output layer colours are fixed to this predetermined colour model and then optimised only for their alpha values. They also construct layers using an overlay representation where the sum of alpha values for each pixel can be greater than one. This method requires the user to initially input both the number of layers and their ordering. Additionally, layers can not have colours close to the centre of the RGB cube as the colour model is extracted from the boundaries of the convex hull. Therefore, the colour model can include imaginary colours absent from the input image which, in turn, can lead to incorrect image reconstruction.

The green-screen keying method by Aksoy et al. [2016] [3] proposes an energy function which is shared with the first term of the sparse colour unmixing equation

(2.3). They introduce alpha sparseness by estimating a per-frame local colour model, which locally determines a subset of the global colour model. The first estimation of this local colour model is done with a Random Markov Field optimisation that later has to be manually refined by the user to correct artefacts. This method achieves satisfactory results for green-screen keying but requires user input. It also lacks a measure for spatial coherency which results in abrupt transitions between layers.

Affinity-based methods such as [10], [11], [12] aim to use local or non-local pixel proximities to propagate a set of given labels to the rest of the image. All the following methods arise from matting estimation. Image matting deals with the estimation of the alpha matte at each pixel, i.e. the contribution of the foreground and background layers at each pixel.

Levin et al. [2008] [10] technique extracts layers with varying opacity values but without colour. The soft segments are automatically computed from eigenvectors corresponding to the smallest eigenvalues of a suitably defined Laplacian matrix. The Laplacian matting matrix is introduced in a previous paper by the same authors in early 2008 [5]. It is used to evaluate the quality of a matte by focusing on the alpha channels without explicitly estimating the foreground and background colours. This approach extends traditional spectral *hard* segmentation techniques to the extraction of *soft* matting components. Their primary aim is to extract spatially connected soft segments, rather than layers of homogeneous colours. However, their formulation requires user input and fails to keep the alpha values between 0 and 1.

Singaraju et al. [2011] [11] introduce a method for estimating alpha mattes for multiple image layers ($N > 2$), extending the traditional foreground-background segmentation. They estimate alpha mattes two layers at a time and eventually connect these sub-problems together. However, their closed-form approach can not satisfy both non-negative alpha values and the alpha values summing up to one at the same time. Similarly to the previous method by Levin et al., only the alpha values are estimated and an additional step is required to obtain the layer colours. Singaraju et al. estimate these colours that best fit the pre-computed alpha values using the energy function introduced in [5]. Unfortunately, the colour constraint is not necessarily satisfied with that formulation.

The KNN matting method by Chen et al. [2013] [12] extracts layers by using K Nearest Neighbours (KNN) to match non-local neighbourhoods. Unlike Singaraju work, their estimated alphas naturally satisfy the alpha constraint of equation (2.2). Given the alpha value, their closed-form solution can be elegantly generalised to extract multiple image layers ($N > 2$). Following their non-local alpha computation, KNN matting solves a sparse linear system to estimate the layer colours independently. This method generalises well to any colour, feature space, dimension, any number ($N > 2$) of alphas and layers at each pixel. In fact, this non-local approach produces higher-quality soft layers when compared to its local counterparts [10] and [11]. However it is highly computationally intensive and disturbing block-shaped artefacts can appear on the layers due to KNN hard constraints around the seed pixels (see fig. 2.4).

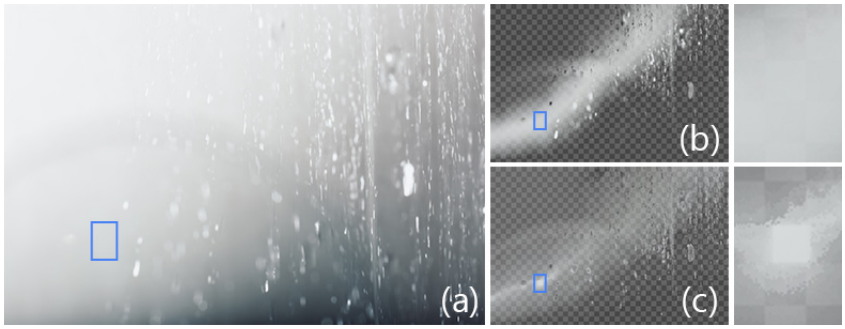

Figure 2.4: A layer computed by Aksoy et al. 2017 algorithm (b) and KNN Matting (c). KNNs hard constraints on alpha values may cause artefacts as shown in the inset.

To summarise, this section shows an in-depth discussion of state-of-the-art soft segmentation methods with a focus on their differences and shortcomings when compared to the "Unmixing-based Soft Color Segmentation for Image Manipulation" one. Throughout this research, we see that the technique presented by Aksoy et al. in 2017, explained in-depth in section 2.1.2, has several advantages in terms of the quality of the opacity channels, the colour content of individual layers and the absence of user input. Indeed, it satisfies the colour, alpha and box constraints; it does not require a predetermined layer ordering or other kind of user input; it provides smooth transitions between layers; and it is scalable to high-resolution images and parallelisable thanks to its per-pixel formulation. In contrast, the other analysed techniques fail in at least one of these aspects. Consequently, this is the soft colour segmentation technique used as foundation of our video soft segmentation.

2.2 Video Segmentation

The previous section presents image-based soft segmentation state-of-the-art methods and details the one selected for our video soft colour segmentation. It is now time to review the state-of-the-art techniques that can help extend this work to the temporal domain by particularly ensuring temporal consistency in the results. The great majority of papers referring to video segmentation deals with *hard* segmentation which targets the extraction of semantically meaningful regions. Video segmentation aims at grouping pixels into spatio-temporal regions that exhibit coherence in both appearance and motion. Such a segmentation is useful for several higher-level vision tasks such as activity recognition, object tracking, content-based retrieval, and visual editing.

Most high-quality video segmentation methods present some forms of motion tracking or optical flow to enforce temporally coherent segmentation. Temporal consistency is usually acquired through the use of optical flow, hence we first detail optical flow computation methods, then recent temporal filtering techniques utilising optical flow, and finally related works on video segmentation.

2.2.1 Optical Flow

Optical flow (OF) is the pattern of apparent motion of image objects between two consecutive frames caused by the movement of objects or cameras. It is the visible motion of an object in an image, and the apparent flow of pixels in an image. It can be represented as a 2D vector field where each vector is a displacement vector showing the movement of pixels from the first frame to the second. Real motion may or may not give rise to optical flow. For example, a sphere uniformly illuminated and rotating about the axis parallel to the image plane would not render optical flow even though real motion exists. Conversely, a stationary sphere with a moving light would render optical flow without real motion.

The first breakthrough papers enabling optical flow computation are differential methods published in 1981 by Lucas and Kanade [13] on one side, and Horn and Schunck [14] on the other side. Both methods propose valid assumptions to formulate solvable equations; they share the brightness constancy assumption which states that

image intensities remain constant under their displacement. In addition, Lucas-Kanade technique assumes that neighbouring pixels have similar motions and computes OF using a local approach; while Horn-Schunk considers a global approach by expecting smoothness in the flow over the whole image and minimises a corresponding energy function.

Lucas-Kanade method is simple and widely used. It resists well to image noise compared to point-wise method as it combines information from several nearby pixels. On the negative side, it cannot provide flow information inside uniform regions of the image as it is purely a local method. Horn-Schunk technique contrasts with Lucas-Kanade one as it produces a higher density flow field. Indeed more pixels are represented because the flow information inside homogeneous regions of the image are filled in from the motion boundaries. On the downside, Horn-Schunk technique is more sensitive to noise than local methods such as Lucas-Kanade one.

Obtaining a reliable optical flow for real-world videos remains a challenging problem caused mainly by the recurring complex cases of motion discontinuities, large displacements and occlusions. Since Horn-Schunk in 1981 who were the first to formulate optical flow estimation as an energy minimisation, energy-based approaches have become increasingly popular and many effective ones have emerged to improve the performance with complex motions [15][16]. Even though these methods are combined with a coarse-to-fine scheme, they still often fail to estimate large displacements caused by fast motions. Sun et al. [17] offer a great analysis of the current practices in optical flow and the principles behind their enhancement. They point out that the majority of today's methods strongly resembles the original formulation of Horn and Schunk. Indeed, they combine a data term that assumes constancy of some image properties with a spatial term that models how the flow is expected to vary across the image. An objective function combining these two terms is then optimised.

Improvements come from additional techniques. The classical methods all perform beyond the original Horn-Schunk algorithm by using a spatial pyramid to cope with fast motion. This spatial hierarchical matching is investigated in recent papers such as those by Kim et al. [2013] [18], Hur et al. [2015] [19] and Hu et al. [2016] [20]. It consists in building a pyramid of an input image in different resolutions; and then using the matching correspondences from higher levels of the pyramid as a guidance

for the matching process on lower levels in a coarse-to-fine scheme.

Furthermore, some state-of-the-art methods are inspired by the Nearest Neighbour Field (NNF) algorithms. NNF computation aims at finding one or more nearest visually similar neighbours for every patch in a given image against another image. Being highly computational intensive, several methods have advanced the efficiency of computing NNF since the seminal work called PatchMatch by Barnes et al. in 2009 [21] to the improved methods by He et al. in 2012 [22] and Korman et al. in 2016 [23]. The key reason for this computational time enhancement lies in random search and propagation between neighbours.

In 2016, Hu et al. [20] introduce CPM (Coarse-to-fine PatchMatch) matching method that combines an efficient random search strategy using NNF (PatchMatch) with a coarse-to-fine scheme of hierarchical architecture to handle optical flow with large displacements. They introduce a fast propagation technique between each pyramid level. The proposed CPM algorithm starts with constructing the pyramids and processing them from top to bottom. On each level, the initial matching correspondences are propagated with random search after a fixed number of time, and the results of each level are used as an initialisation of the next lower level. The results after the propagation on the finest level are their matching results. Their work assumes that the visual similarity between two image regions is the most important clue for large optical flow estimation.

To summarise, state-of-the-art OF methods are mainly energy-based methods resembling the original Horn-Schunk technique. They can be local or non local; and additionally use Nearest Neighbour Field for random search, or a spatial pyramid allowing coarse-to-fine propagation. One of the best performing technique is the CPM matching method by Hu et al. [2016] which combines an efficient random search with the coarse-to-fine scheme on a sparse image grid structure. The evaluation on modern datasets such as the MPI-Sintel Database [24] shows that CPM is capable of providing highly accurate optical flow for large displacements and is more efficient than state-of-the-art methods with similar quality.

2.2.2 Temporal Filtering

In this section, we describe recent state-of-the-art papers in the field of temporal filtering. Within the scope of video segmentation, spatial filtering improves smoothness at the frame level and temporal filtering enhances the temporal consistency between two successive frames. It should enforce that colour segments stay temporally similar within the same video shot. In general, temporal filtering uses classic image-filtering methods extended to the temporal axis t. Therefore this section starts with a quick description of common image-filtering methods.

Initially, the most widely used filtering operation attenuates or removes noise in a specific signal (image). In order to correctly restore the signal (image), the methods either require to model the signal from an initial belief or to learn the underlying characteristics of the signal from the given data. Using the latter has become very popular notably with the rise of patch-based techniques [25] [26] [27]. Patch-based methods look for similarities and affinities at both the local and non-local scopes. They measure affinities between a given data point, patch or region of interest and others in the given image. These similarities are then used to filter contributions from more similar data values with higher weights, and to properly discount filtering between data points that are less similar. Other filtering techniques include the efficient Mean Shift algorithm [28] from pattern recognition and general regression problems such as least-squares regression [29] from machine learning. For a recent description of general filtering methods in image applications, please refer to Milanfar exhaustive review in 2013 [30].

Edge-aware (EA) filtering methods generate useful local flexibility to the given data by following the frame spatial information. One of the first established EA methods is the bilateral filter (BF) [25] which is easy to construct and simple to compute. Since then, a great diversity of EA filters [31] [32] [33] [34] [35] has been developed for diverse applications such as colourisation, High Dynamic Range (HDR) tone mapping, detail editing and noise reduction. The Weighted Least Squares (WLS) filter [31] is a celebrated method producing high quality results which suppresses artefacts by penalising the distance between the original and filtered images; on the downside it has a high computation cost. Methods such as the Guided Filter (GF) [7], the Permeability

Filter (PF) [36] and the Domain Transform (DT) [32] show a good trade-off between quality and complexity. GF and DT, however, still suffer from halo-artefacts, which is not the case of PF. PF is designed to mimic similar behaviour as the high-quality WLS filter - but with significantly lower computational complexity.

Temporal consistency is one of the main challenges in video processing as the frame-by-frame brute force approaches of computing image-based methods often produce incorrect visual artefacts [34]. Spatio-temporal EA filters [33] [34] perform well in removing high-frequency temporal artefacts such as noise and flickering. On the other side, some algorithms such as described in [37] and [38] can correct low-frequency defects, but their large optimisation formulations provoke a significantly increased complexity and necessitate pre-computed accurate optical flow. Even if limited for handling low-frequency noises, spatio-temporal EA is an efficient trade-off for introducing temporal consistency.

Lang et al. [33] produce temporal consistency with their spatio-temporal EA filter by filtering along motion paths using an iterative approach that simultaneously uses and estimates per-pixel optical flow vectors. Their method can be used to create temporally consistent results for optical flow, disparity estimation or scribble propagation among others. The state-of-the-art method by Schaffner et al. in 2017 [39] enhances Lang et al. as a mean to yield accurate optical flow using sparse correspondences and interpolation. They define a temporal filtering extension requiring smaller memory and bandwidth access to reduce computation and memory resource. This is achieved by filtering only over the past video frames and using the Permeability Filter (PF) [36] kernel, originally introduced to filter disparity data, and successfully extended to the temporal domain to filter HDR data in [34]. The PF can be implemented with linear complexity.

To sum up, this temporal filtering section shows that edge-aware filtering methods can efficiently introduce temporal consistency in a video. In particular, the recent method by Schaffner et al. illustrates the convenience of the edge-aware permeability filter to spatially and temporally refine optical flow. This permeability filter outputs results similar to the high-quality WLS filter while significantly simplifying the computational complexity.

2.2.3 Related Works

A first brute force approach to video segmentation is to apply image segmentation techniques to video frames without considering temporal coherency [40] [41]. While easily scalable and potentially fast, these methods lack temporal information from neighbouring frames, which usually provokes apparent flickering across frames. More appropriate methods include the temporal aspect in their segmentation. For instance, a paper by Wang in 1998 [42] presents an unsupervised video segmentation technique divided in two phases: an initial segmentation followed by temporal tracking. Temporal tracking is performed after the first frame of a video sequence has been segmented into moving objects. The objective of temporal tracking is to segment the subsequent frames of the video sequence and to establish a correspondence of moving objects between frames.

Spatio-temporal video segmentation techniques can generally be separated depending on whether they simply use past frames or whether future frames are used as well. Causal methods apply Kalman filtering to aggregate data over time and only consider past data such as in [43] and [44]. Spatio-temporal techniques utilising past and future frames typically treat the video as a 3D space-time-volume [45] and segment this volume thanks to a variant of the mean shift algorithm for segmentation [46] [47]. As for tracking-based video segmentation methods, they generally define segments at frame-level and use motion, colour and spatial cues to force temporal coherence [48] [49]. In addition, Brendel and Todorovic [50] use contour cues to allow splitting and merging of segments to boost the tracking performance.

Another type of video segmentation can be witnessed in the hierarchical graph-based video segmentation method introduced by Grundmann in 2010 [51]. This method generalises Felzenszwalb and Huttenlochers [52] graph-based image segmentation technique. Grundmann et al. begin by over-segmenting a volumetric video graph into space-time regions grouped by appearance. Then they construct a region graph over the obtained segmentation and iteratively repeat this process over multiple levels to create a tree of spatio-temporal segments. They further improve segmentation quality by using dense optical flow to guide temporal connections in the initial graph.

Finally a higher-level approach uses a structure-from-motion (SFM) algorithm to

estimate the 3D-scene structure [53], which is then segmented leading to a corresponding 2D segmentation. It is a model-based approach as it starts by using a robust structure-from-motion algorithm to identify multiple objects and recover initial 3D-shape models. Afterwards, these models are used to identify and track the objects over multiple frames of the video sequence. Though interesting for video segmentation, SFM has a very high complexity inducing a long computation time.

All the methods discussed above are video *hard* segmentation. Very little research has been done in the field of video *soft* segmentation. We can cite the method by Bai et al. [2009] [54], which proposes an interactive framework for soft segmentation and matting of natural images and videos. The technique presented is mainly based on computation of weighted geodesic distances to user-provided scribbles, from which the whole data is automatically segmented. The paper displays diverse applications including extraction of moving foreground from dynamic background in video and natural and 3D medical images.

The framework is introduced for still images and extended to video segmentation and matting. For every pixel in the image, they compute the geodesic distance to every user's scribbles. The geodesic distance is the smallest integral of a weight function over all possible paths from the scribbles to the current pixel. The weights are here selected as the gradient of the likelihood that a pixel belongs to the foreground. This likelihood is based on a specific feature that can be chosen depending on the image / video content and on the particular application: a classic choice is pixel colour but for instance frequency response of Gabor filters can be used for texture images and optical flow can be used for image sequences of moving objects.

This method by Bai et al. necessitates a heavy user intervention through scribble drawing to label the region of interest and in a later stage to refine the segmentation. It can be justified as foreground and background interpretation of objects of interest can differ from one person to the other. This method is limited as it only depends on the pixel value distributions. As such, the algorithm performs poorly when these distributions overlap too much. In principle, this could be solved by tedious additional user interactions.

To summarise this section on video segmentation, several methods exist for video

hard segmentation such as efficient spatio-temporal segmentation, hierarchical graph-based segmentation or structure-from-motion segmentation. However very little research has been done on video *soft* segmentation yet. A matting example creating soft segments for the foreground and background of a video was introduced by Bai et al. in 2009. It yields correct results but necessitates heavy user interaction through scribbles and has difficulties to segment the video in regions where distributions overlap too much.

The next chapter 3 describes the method we propose for video soft colour segmentation that combines several state-of-the-art background techniques.

Chapter 3

Design and Implementation

This chapter first presents the design of the proposed method in section 3.1. Then section 3.2 explains how the method is technically implemented.

3.1 Design

Our objective is to design a video soft colour segmentation which automatically produces high-quality layers. These layers have to be temporally consistent. The computation time needed for each individual frame should also be reduced in comparison to the independent image-based segmentation. Two aspects are analysed to achieve these objectives: how to use motion information through optical flow to improve the video layer results, and how to properly set the colour model throughout the video.

3.1.1 Layer Initialisation using Optical Flow

This section introduces a layer initialisation technique using Optical Flow (OF). We select this technique because we assume that, as optical flow is a measure of motion information, it can be used to propagate the layer colour and alpha values from frame N to frame $N+1$. These propagated layers can then be employed to initialise the colour unmixing energy described in eq. (2.3) before this energy is minimised to refine the layer estimates. Initialising the image segmentation following the optical flow should

enforce temporal consistency. In addition, using a more accurate initialisation of the layers in the unmixing energy should also speed up the convergence of the algorithm.

The whole process is shown in fig. 3.1. To begin with, an input frame N is segmented following the image-based soft colour segmentation presented in section 2.1.2. Afterwards, the output colour layers of frame N are shifted following the OF from frame N to frame $N+1$. This flow can be either computed along the video segmentation or loaded from given pre-computed flow files at the start of the program. Finally the shifted layers are used to initialise the colour layers of the subsequent frame before minimising the energy function (equation 2.3).

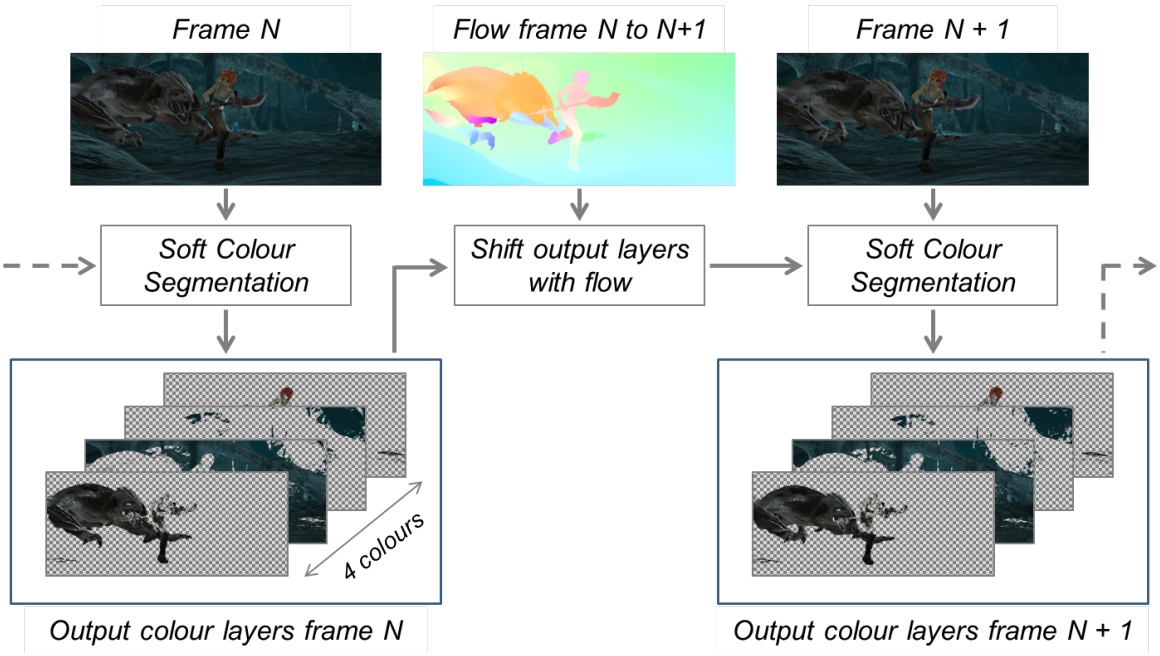

Figure 3.1: Overview of the layer initialisation technique. An input image N is first segmented. The resulting layers are shifted following the optical flow. Finally these shifted layers are used to initialise the next frame segmentation.

3.1.2 Colour Model Estimation

The Colour Model (CM) represents the principal set of colour distributions of an input image. Our method uses the colour model computation presented in the image-based

soft segmentation paper by Aksoy et al. [1]. The whole computation technique is described in section 2.1.2. In order to extend this image-based estimation to video, the method examines how and when a new colour model should be computed. Especially, how to detect the key frames on which the colour model has to be reset. The chosen technique has to correctly handle video sequences with cuts. A video cut is defined as the separation between two different video shots. In general, the frames before and after a cut vary greatly in terms of colours and thus their colour model can be very different.

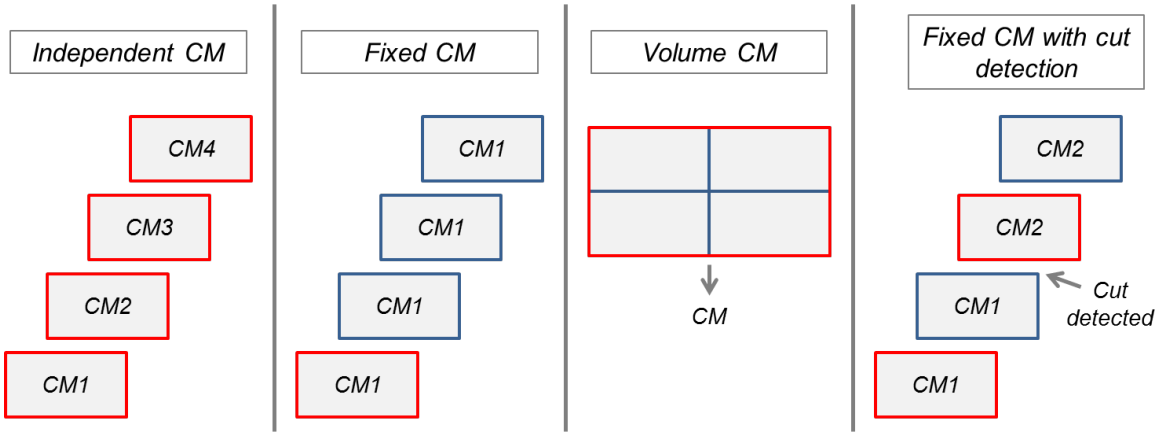

Figure 3.2: Different colour model estimation approaches on four successive frames.

The proposed approaches for the colour model estimation summed up in fig. 3.2 are:

1. Independent CM: The first brute force approach consists of computing the CM for each frame independently and then applying the soft segmentation on each frame with its associated CM. The quality of this technique mainly relies on the stability of the CM computation, it requires similar frames to output the same number and set of colour distributions so that the video segmentation per layer stays temporally coherent.

2. Fixed CM: A simple approach to keep the same CM for the full video is to calculate the colour model on the first frame and apply it on all subsequent frames. However, if the video contains a cut, one can expect difficulty when

converging to a correct segmentation, particularly if the new shot displays colours far from the currently fixed CM.

3. Volume CM: A separate way involves stacking all frames from a video together and estimating the CM from this larger image representing the video volume before using it individually on each frame. Hence the CM represents all the main video colours and stays the same throughout the video. The efficiency of this technique could decrease as the number of frames increases.

4. Fixed CM with cut detection: Finally this last way fixes the CM until a cut is detected in the video. Every time a cut happens, the layers are reset, the CM is recomputed and then remains the same until the next cut is identified. The cut detection is done by calculating the RGB histogram distances between two successive frames. When this distance is higher than a certain threshold, a cut is found or at least considered as such. Indeed in some cases, the introduction of a large object in the scene can be identified as a cut and would also require to reset the colour model.

3.2 Implementation

This section describes the implementation of the designed video soft colour segmentation method.

3.2.1 Computer Work Environment

The majority of this dissertation work, including the evaluation, was done on a personal computer with the following CPU: Intel core i7-3630qm, 2.4GHz; and GPU: Nvidia GeForce 660M, 2GB. Therefore, the computation time results presented in chapter 4 are performed on a medium-performance machine and would likely be faster on a more recent and powerful computer.

Our code was implemented on a Linux environment, precisely Ubuntu 16.04. The code platform is Visual Studio Code associated with an internal GDB debugging. The

code is shared on the web-based Git-repository manager GitLab (private link: https://gitlab.scss.tcd.ie/V-SENSE/extended_soft_segmentation). Some Linux-libraries such as "boost" or the system "make directory" are used, hence the current implementation is not directly portable to Windows.

Throughout this dissertation, the open content movie "Sintel" copyright ©Blender Foundation www.sintel.org produced by Blender Institute [55] is used to present and test our method. MPI Sintel Dataset [24] contains open-source ground truth optical flow of the "Sintel" animated movie which are employed for the evaluation of the method.

3.2.2 Preliminary Work

Before beginning the extension of the available image-based soft segmentation code, some changes were required. Indeed, upon inspection, it became clear that several steps needed to be optimised and refined so that the code would accurately reflect the algorithm proposed by Aksoy et al. [1]. The initial code was an incomplete version as the Colour Refinement step was missing from it. The Colour Refinement step explained in section 2.1.2 comes just after the Matte Regularisation step which spatially filters the alpha values. After integrating the current state of the code in our framework without the Colour Refinement, the computed layers were showing reconstruction issues. Indeed, some pixels were not converging correctly, which in turn highly increased the reconstruction error. The reconstruction error represents how far the sum of every layer is from the original image. Having a low reconstruction error is a necessary condition to enable correct image editing. The initial code produced layers with alpha values not adding up 1.0, some of them being as low as 0.5. The following section explains how we handled this convergence issue.

Initial Code Fixes and Modifications

Figure 3.3 (b,d) shows the opacity error in the reconstructed image as well as the pixels having an opacity lower than 0.8 or higher than 1.1. Every pixel needs to have their layer opacity values adding up to a value close to 1.0 in order to enable correct image

editing. The initial image is a painting by J. M. W. Turner made in 1842 and named "The Blue Rigi, Sunrise".

Upon inspection of the original code, we found that when computing the gradient of the minimisation energy function (equation 2.3) to find the direction towards the minimum, only the blue channel was taken into account at the expense of the green, red and alpha channels. This wrong gradient explains why most pixels were not converging to a correct unmixing. Using an accurate gradient already leads to significantly better output layers, as can be seen on fig. 3.3 (c,e).

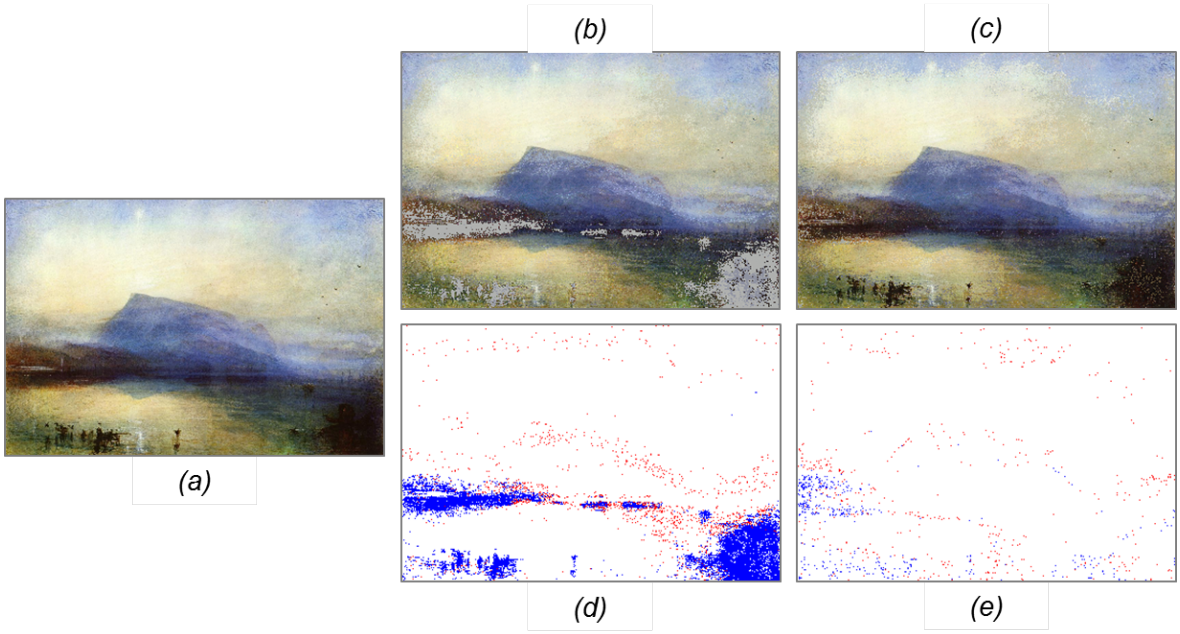

Figure 3.3: Initial image (a), image reconstruction before (b) and after (c) code corrections. Their respective reconstruction error images (d) and (e) show pixels with opacity values lower than 0.8 (blue) or higher than 1.1 (red).

Nonetheless, the reconstruction remained imperfect, and we focused on the minimisation process to understand why some pixels were still converging incorrectly. In order to do so, we tweaked several minimisation parameters such as the maximum number of iterations per pixel before stopping the minimisation, the tolerance value under which the minimisation stops and whether the method is reaching this maximum number of iterations or reaching the tolerance threshold. Eventually this study revealed that the cost function taking a value below this tolerance threshold is not proof of good

convergence. Indeed the best results are achieved with a high maximum number of iterations and a very low and unreachable tolerance. The more iterations, the better.

This demonstrates that the tolerance is not playing its part. The original tolerance value represents the distance between the previous and current pixel unmixing values; it is equal to the step size between the previous and current pixel unmixing values. The issue is that this step size is systematically reduced at each iteration and that the direction in which we step is periodically reset. Therefore if the current direction does not point towards the minimum, the step size would still get smaller even though the pixel unmixing colour does not improve. As such, the step size is not a good indication of convergence. In other words, the step can decrease towards zero even if the energy minimisation itself is not improving.

Instead of comparing the distance between two subsequent unmixing values of the same pixel which equals the step size, we choose to compare the distance of the current pixel colour with the original image pixel colour as follows:

$$dist = \left\| \sum_i \hat{\alpha}_i \hat{\mathbf{u}}_i - \mathbf{c} \right\|_{L2}, \tag{3.1}$$

where $\hat{\mathbf{u}}_i$ and $\hat{\alpha}_i$ are respectively the estimated colour and alpha values of the i^{th} layer at the current step of the optimisation, and $\mathbf{c}$ is the original image pixel colour.

The tolerance threshold is set to 4.0 in our current implementation. Eventually after changing the tolerance to the RGB distance rather than the step size, the reconstruction error finally goes down and the output layers are perfectly adding up to the original image. The table 3.1 displays the reconstruction error after each correction step. It is clear that after the different corrections, the reconstruction error is substantially lower, which is required to continue towards the video extension.

	Initial Code	After code fixes	After code modifications
Reconstruction Error	0.070638	0.044447	0.000538

Table 3.1: Output layers reconstruction error of the image “The Blue Rigi, Sunrise” after code fixes and modifications.

Video Utility

Thereafter, the corrected and modified image-based code is extended to include a video framework. A class to handle all video operations is introduced. It can load frames directly from an input folder or from an input video file. An output folder is automatically created at run-time to organise all the different outputs from the video soft segmentation. Some utility functions to save all output layers as videos are also included. However mainstream video formats do not allow transparency channel visualisation such as a chess pattern, thus we decide to fill the transparent values with a white background before saving the output layers.

3.2.3 Layer Initialisation using Optical Flow

In order to initialise the layers using Optical Flow (OF), we first need to compute the flow. This is done by integrating the Permeability Filter (PF) [36] to spatially and temporally obtain a refined Optical Flow (OF) following the video motion path similar to Schaffner et al. [2017] [39]. The initial raw OFs are computed with the Coarse-to-fine PatchMatch (CPM) method by Hu et al. [2016] [20]. This recent method has proven to output high quality OF. CPM combines an efficient random search strategy using Nearest Neighbour Field (NNF and PatchMatch) with a coarse-to-fine scheme of hierarchical architecture to handle optical flow with large displacements. As with other OF estimations, the biggest failing cases appear for images containing too many flat surfaces or texture-less zones. In those cases it becomes difficult to detect feature points or similar patches between images, which is the basis for any OF computation. Please refer to section 2.2 for the full state of the art on optical flow and temporal filtering.

The CPM and PF integration to our framework was complex due to the difficulty to handle the libraries with cmake. Ultimately, CPM was integrated as an external library while the PF was simply appended inside the current code. For a more steady computation time, we decide to compute the refined optical flows one at a time during run-time and not all at the beginning of the program.

Our video soft segmentation process begins by loading all the video frames from a

folder or from a video file. For a frame N, it then runs as follows: it starts by computing the frame histogram distance. If $N = 0$ or if a video cut is detected, the colour model is computed. Thereafter the process computes the spatio-temporal flow from frame N to frame $N + 1$ using the permeability filter. Afterwards the output layers from frame $N - 1$ are shifted by the refined flow. The image-based soft colour segmentation of Aksoy et al. [1] is initialised with those shifted layers. After the minimisation, all output elements are saved in the output directory. If it is the last frame, the output layers are also saved as videos.

3.2.4 Colour Model Estimation

Among the four techniques presented in the design section 3.1, the first three can easily be implemented by slightly modifying the initial colour model estimation of Aksoy et al. [1]. That's why this subsection focuses on the implementation of the fourth technique: fixed CM with cut detection, and especially how to identify the video cuts.

Video Cut Detection

A great amount of work has been done in the field of video cut detection. Classic approaches define a similarity measure between successive images. Two images being sufficiently dissimilar may indicate a cut. Gradual transitions are found by using cumulative difference measures and more sophisticated thresholding schemes. Based on the metrics used to reveal the difference between successive frames, the algorithms can be divided broadly into three categories: pixel, block-based and histogram comparisons. For an overview of these different categories, please refer to Koprinska survey [2001] [56].

Histogram comparisons are greatly time-efficient and reliable in detecting cuts, but two images with similar histograms may have completely different contents. However, the probability of this situation is rare in practice. Moreover, it is not a real drawback for our video soft colour segmentation. Indeed, as the colour distributions would be similar before and after the cut, it will still give coherent colour layers for both video shots. Therefore we select this algorithm for identifying video cuts. The final objective

of this fourth technique is to segment the video into cuts to properly fix the colour model within these cuts.

Histogram Comparisons

A histogram depicts the different colours present in an input image. To determine a histogram, a number of colour levels must be selected. Afterwards each pixel is sorted out in the range of intensity or level where it belongs. The output is an n-dimensional vector with the number of pixels present in each level.

Originally, grey-level histograms are used to detect video cuts, then their distance is computed as the sum of the absolute differences of each bin from frame $N + 1$ with each bin from frame N. One approach to increase segmentation accuracy is to consider the intensity distributions of the individual colour channels. This is the approach we implement. We are calculating the euclidean distances of the R, G and B colour histograms with level of range 1 from an intensity of 0 to 255. Hence our histograms contain 256 levels. The final histogram distance is the average of the R, G and B colour histogram distances. A cut is declared if this average distance between two successive frames is higher than a threshold T. In our implementation, T has been empirically set to 0.4.

Chapter 4

Results

This chapter examines the results of the method presented in chapter 3. Similarly to the method, the evaluation is twofold with the results of the layer initialisation in section 4.1, and those of the colour model estimation in section 4.2. Each part starts by describing the evaluation process including the metrics used.

4.1 Layer Initialisation using Optical Flow

4.1.1 Evaluation Process

Overview

The layer initialisation technique is evaluated on the sequences from the MPI Sintel Dataset [24]. This dataset comprises 23 video test sequences extracted from the animated movie Sintel copyright ©Blender Foundation www.sintel.org. We select this dataset as the associated ground truth optical flows are included. After a preliminary output analysis, we then evaluate the temporal smoothness and the time performance for each sequence with the three following techniques:

1. No flow: This is the reference technique where each frame is segmented independently with the image-based segmentation by Aksoy et al. [1]. The optical flow

is not used.

2. Initialisation with custom flow: The minimisation is initialised with the shifted output layers from the previous frame. The output layers are shifted using the refined optical flow computation technique by Schaffner et al. [39], where the optical flow is spatially and temporally filtered using the permeability filter along the video motion path.

3. Initialisation with ground truth flow: Instead of computing the optical flow during run-time, this technique loads the ground truth flows of the MPI dataset at the start of the program and utilises them to shift the output layers for initialisation. This ground truth technique serves as a comparison against the one with custom flow.

Each of these techniques uses the *Fixed CM* estimation, which means they have the colour model fixed on the first frame in order to properly compute the temporal smoothness on the output layers. All sequences from the MPI Sintel Dataset are video shots without cuts, therefore it justifies using the *Fixed CM* technique (see section 4.2 for the results of the colour model estimation). Moreover each approach is derived in two ways: with and without the matte regularisation step which spatially filters the output layers using the Guided Filter (GF). For the techniques initialising the layers with optical flow, the GF is applied before shifting the layers, thus the next frame is initialised by already spatially filtered layers.

To sum up, the Sintel test sequences are computed and tested in six different ways: No flow with and without GF, Initialisation with custom flow with and without GF, and finally Initialisation with ground truth flow with and without GF. The evaluation compares the independent output of each frame segmented using image-based soft segmentation method by Aksoy et al. [1] with the output frames from our video soft segmentation. The initialisation with the refined custom flow is also compared to the ground truth flow one.

Temporal Smoothness Metric

To evaluate the output layers temporal coherency, we define a smoothness metric similar to the one presented by Schaffner et al. [39]. To properly specify this metric, the trace tr_p of a pixel p has to be defined. The trace of a pixel p starting at position (p_x, p_y) in the first frame is the sequence of all its positions following the ground truth flow throughout the video (see fig. 4.1). The trace length can vary greatly as it stops as soon as the pixel is being occluded or disappears outside of the frame. Schaffner et al. find that for the Sintel sequences, this results in traces with an average length of 21 samples. In our case, the smoothness metric is employed on pixels in the output layers which additionally have a non-zero alpha channel. Thus the trace also stops when the next position is completely transparent, giving traces of smaller average length.

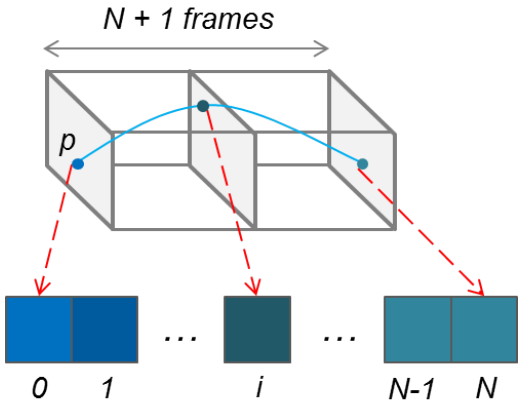

Figure 4.1: Visualisation of the trace of a pixel starting in p at frame 0 throughout a video volume of N + 1 frames and its corresponding vector of RGB values.

Following this trace, the RGB and alpha values at the different positions are stored and can be used to define the smoothness metric ψ. Indeed this trace vector represents the variance of the RGB colours or alpha values along the motion path and can serve to estimate temporal coherency for both RGB and alpha values.

Once the pixel trace $\mathbf{tr}_p$ of a pixel p is computed, its smoothness metric ψ can be calculated as follows:

$$\psi(\mathbf{tr}_p) = 0.5 \cdot acf(diff(\mathbf{tr}_p), 1) + 0.5 \tag{4.1}$$

The $diff$ operator differs depending on if $\mathbf{tr}_p$ contains RGB or alpha values. In the first instance, $diff$ calculates the 3D distances in RGB colour space of a pixel trace in each pair of subsequent frames. As for the alpha values, the $diff$ outputs the direct difference of the initial pixel trace values two-by-two. In both cases, the resulting vector is auto-correlated in lag 1. In the above equation, acf represents this auto-correlation function. The auto-correlation represents how much a signal is similar to a delayed version of itself. In the case in point, we estimate the correlation of the pixel trace of RGB distances / alpha differences in lag 1, i.e. the similarity of colour / alpha values of subsequent frames following the motion path. With this representation, the smoother the colour distances and alpha differences are along the vector, the higher the metric score is. The auto-correlation version utilised is the normalised one, therefore the smoothness results go from 0 to 1, from discontinuous to smooth.

4.1.2 Results

Output Layers Analysis

A first overview of the results is displayed in fig. 4.2. It shows some key frames that have a dark colour distribution for the three presented evaluation techniques without Guided Filter (GF). The first remark is that our layer initialisation technique creates artefacts both when using the custom and ground truth flows. Overall the custom and ground truth flow initialisation techniques show very little difference in the output layers, therefore some future diagrams only include the ground truth flow results for clearer visualisations. The artefacts that can be seen on frames 10 and 20 gradually appear starting at frame 1 when the first layer initialisation takes place. Frame 0 is similar for all shown techniques as the initialisation only starts from frame 1.

For all Sintel sequences tested with the two layer initialisation techniques, the output layers tend to blend into one another: originally opaque pixels slowly fade out, while originally transparent pixels slowly appear. Unfortunately, the output layers mix together as the video elapses, and this visually shows that the layer initialisation technique with optical flow is not inducing temporal smoothness. On the contrary, the frames computed independently without using the flow seem to perform much better

in that regard.

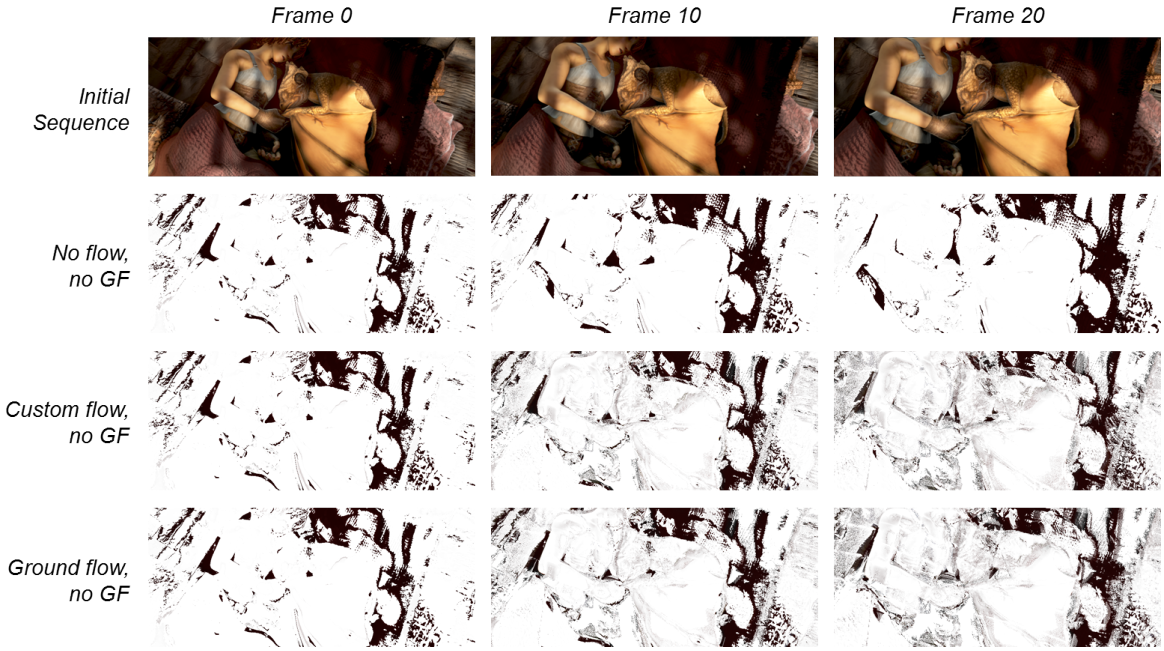

Figure 4.2: Sintel sleeping1 frames 0, 10 and 20 segmented with the three different video segmentation techniques. The layer shown corresponds to a dark brown colour distribution. The white background signals transparent pixels.

The Sintel sleeping1 sequence segmented with the additional Guided Filter step does not clearly show the difference between the independent segmentation (No flow) and our initialisation technique (Custom flow and Ground truth flow). Indeed the appearing artefacts are mostly smoothed out by this filter but the phenomenon still happens underneath (see percentage of opaque pixels graphs in fig. 4.3). However the videos with the output layers have clear flickering which are more frequent with the initialisation than without. This flickering is expected as the GF is a spatial filter and not a temporal one, which induces that subsequent frames may possess very different outputs. Nonetheless it still looks more spatially coherent when no optical flows are used as the amount of flickering is lower in comparison. The initialised layers are inclined to produce more artefacts and shift the layers further than it would be when estimating the layers independently. Therefore the spatial filtering of those already altered layers could differ even more from one frame to the other. Indeed we have seen that No flow outputs more coherent layers, which logically give more coherent spatial

filtering layers as the layers are more similar to start with.

The layer initialisation using optical flow visually yields temporally incoherent results. On the other hand, the independent approach without flow seems to produce proper temporal coherency. The following parts take a closer look at the quantitative results in terms of smoothness and time performance.

Temporal Smoothness

Both RGB and alpha temporal smoothness results computed using the temporal smoothness metric introduced earlier in eq. (4.1) are summarised in table 4.1 for five Sintel sequences and the six layer initialisation approaches.

		No flow				Init custom flow				Init truth flow			
		No GF		GF		No GF		GF		No GF		GF	
		μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
Sleeping1	RGB	0.58	0.23	0.65	0.21	0.57	0.20	0.68	0.21	0.59	0.20	0.68	0.21
	Alpha	0.62	0.31	0.65	0.25	0.51	0.28	0.51	0.25	0.61	0.27	0.52	0.26
Alley1	RGB	0.69	0.22	0.70	0.20	0.69	0.19	0.74	0.18	0.72	0.18	0.74	0.18
	Alpha	0.69	0.30	0.63	0.23	0.45	0.22	0.58	0.21	0.56	0.22	0.58	0.21
Mountain1	RGB	0.61	0.22	0.61	0.20	0.59	0.19	0.63	0.19	0.61	0.19	0.62	0.19
	Alpha	0.71	0.30	0.68	0.29	0.49	0.28	0.55	0.26	0.56	0.28	0.55	0.26
Cave2	RGB	0.49	0.17	0.52	0.17	0.50	0.16	0.58	0.15	0.54	0.15	0.58	0.15
	Alpha	0.65	0.30	0.49	0.19	0.36	0.16	0.48	0.16	0.43	0.18	0.49	0.16
Alley2	RGB	0.60	0.20	0.64	0.19	0.61	0.18	0.68	0.20	0.67	0.17	0.71	0.18
	Alpha	0.72	0.31	0.64	0.25	0.42	0.20	0.49	0.21	0.54	0.21	0.59	0.23

Table 4.1: Average temporal smoothness on all layers for both RGB and alpha values. μ is the average mean of the smoothness vector and σ the standard deviation.

Let's first analyse the results on the RGB temporal smoothness. To begin with, the average mean values are overall better with the Guided Filter (GF). Moreover, the RGB smoothness results are similar without flow and with the initialisation with custom flow but are usually better with the ground truth flow. This suggests that the initialisation with the ground truth flow offers the best spatial coherency, but the video layers and fig. 4.2 show otherwise.

Actually this outcome seems to indicate that the value computed using the smoothness metric is a measure of the layer's RGB temporal coherency. And as the layers are associated with a set of colour distributions with variance and mean, this metric may indicate more about the current layer's colour variance rather than the temporal consistency. On another note, cave2 has the worst RGB smoothness of all sequences. The layers of cave2 are not smooth because the initial video has large scope motions.

As for the alpha smoothness, not using the flow performs much better on all tested Sintel sequences. Furthermore the initialisation with ground truth flow technique outputs better results compared to the technique using the flow computed during run-time. The important variations induce that the optical flow custom estimation may not be satisfying enough. When comparing with and without GF, the alpha temporal smoothness is sometimes better when initialising with the flow as in alley1, cave2 or alley2 but sometimes lower for instance for sleeping1. The guided filter brings spatial consistency that creates smoother alpha spatially but it is not a temporal filter thus sometimes strong flickering appears and make the smoothness worse. Both alpha smoothness and RGB smoothness show high standard deviations highlighting that the output results differ greatly depending on which layer is evaluated. Indeed the layers can have a very different number of pixels represented depending on the colour distributions.

In summary, the studied smoothness metric shows that the colour smoothness is the best without flow and with the ground truth flow. However this result might be biased by the underlying layer colour variance. As for the alpha smoothness, it indicates that computing each frame independently without flow is better and is inconclusive regarding the guided filter step.

Even though the RGB temporal smoothness scores stay high when using the ground truth flow, fig. 4.2 shows that both initialisation techniques create artefacts. Yet these artefacts do not have a clear impact on the temporal smoothness results. To clarify this behaviour, we observe the percentage of non-transparent pixels ($\alpha \neq 0$) in the sequence sleeping1. The resulting curves are displayed on fig. 4.3. The custom flow and the ground truth flow are very similar, therefore only the ground truth flow graph is present on the figure for more clarity.

This figure demonstrates a specific behaviour of our layer initialisation technique:

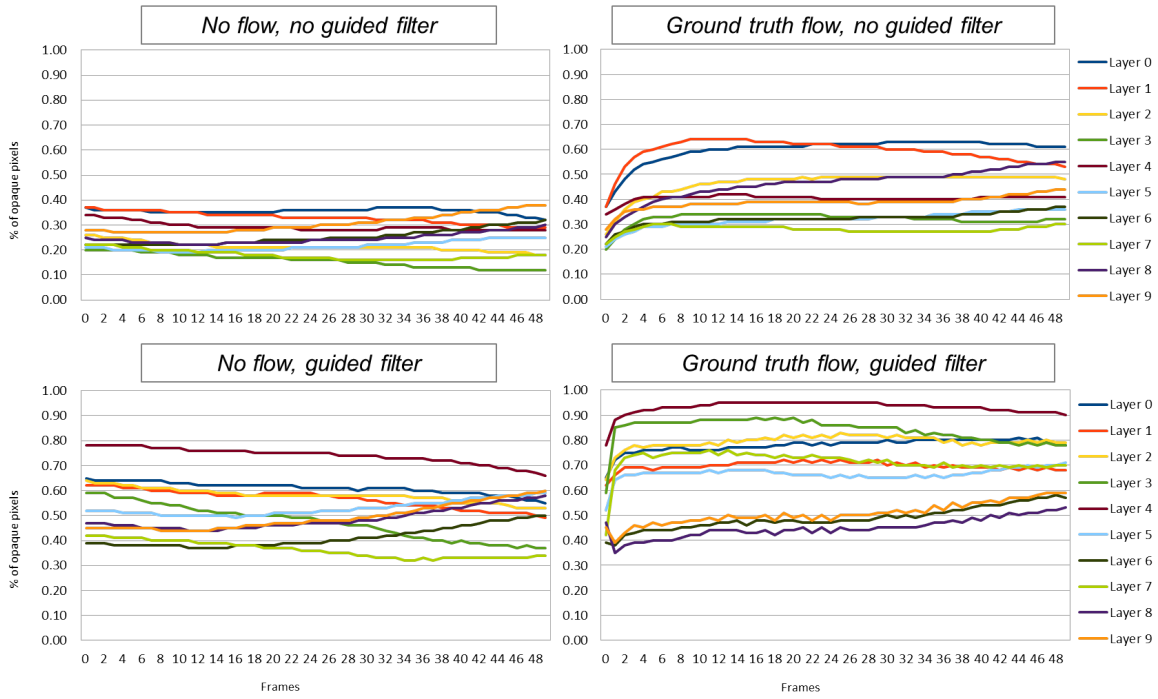

Figure 4.3: Percentage of non-transparent pixels for all layers over the 50 frames of the Sintel sleeping1 video sequence.

the pixels tend to spread on more and more layers as the video continues. Indeed, fig. 4.3 reveals that both with and without the guided filter, the percentage of non-transparent pixels quickly increases in the first ten frames with the output layers initialisation. What happens is that the minimisation step from the image-based soft colour segmentation has a tendency to spread the given pixel unmixing on more layers. Therefore, when initialising the output layers with the previous segmentation, this minimisation starts with pixels already partially transparent and diffuses them even more. This pixels diffusion on several layers keeps on happening until it stagnates after a certain amount of frames.

Without the flow, the pixels given to the minimisation are initially fully opaque in a single layer. With the flow, the pixels given are initially non-opaque and keep spreading on several layers. This is undesirable as the video segmentation should mostly produce sparse and compact layers for proper video editing. Initialising the output layers before the minimisation removes this sparsity component initially present in the image-based

segmentation by Aksoy et al. [1].

Time Performance

After witnessing the unexpected behaviour of the minimisation step when initialised with non-opaque pixels, we also want to verify whether the initialisation approach speeds up the segmentation process or not. To that end, the computation time of different parts of our program is recorded. The various times are shown in table 4.2 for the sequences sleeping1 and alley1. The flow computation, frame unmixing and GF values are specified in seconds as average per frame. These computation time results are computed on a personal computer with medium performance (see section 3.2.1).

sleeping1	No flow		Init custom flow		Init truth flow	
CM = 10 colours	No GF	GF	No GF	GF	No GF	GF
Load video	3	3	2	2	2	2
Load flow	0	0	0	0	6	8
Avg Flow computation	0	0	6	6	0	0
Avg Frame unmixing	92	92	115	123	120	123
Avg Guided Filter	0	1	0	1	0	1
Total (sec)	4,661	4,709	6,140	6,602	6,093	6,283
Total (min)	77.68	78.48	102.34	110.03	101.56	104.72
alley1	No flow		Init custom flow		Init truth flow	
CM = 6 colours	No GF	GF	No GF	GF	No GF	GF
Load video	2	2	2	2	2	2
Load flow	0	0	0	0	6	7
Avg Flow computation	0	0	6	6	0	0
Avg Frame unmixing	59	59	89	91	86	92
Avg Guided Filter	0	1	0	1	0	1
Total (sec)	3,010	3,039	4,797	4,914	4,382	4,691
Total (min)	50.17	50.65	79.95	81.90	73.03	78.19

Table 4.2: Computation time in seconds of the main program sections for Sintel sleeping1 and alley1 with the six evaluation approaches.

Table 4.2 reveals that the techniques without using the optical flow are always faster. For instance, the average frame unmixing corresponding to the minimisation step is 1.33 times faster for sleeping1 and 1.5 times faster for the alley1 in comparison

to the techniques employing the layers initialisation. For alley1, the total computation of No flow without GF is almost twice as long as the custom flow with GF with values of 50 minutes and 82 minutes. The initialisation with ground truth flow is slightly faster than the initialisation with custom flow mainly because of the flow computation time. Indeed loading the ground truth optical flows takes a total of about 6 seconds for both sequences whereas the custom flow computation lasts about 6 * 50 = 300 seconds for all 50 frames. Concerning the guided filter step, it is performed in about a second for each frame which is pretty quick. Finally the table points out that alley1 total computation time compared to sleeping1 is about 1.5 faster without flow and 1.35 faster with flow. This is due to the number of colour distributions in the colour model which amounts to 10 for sleeping1 and 6 for alley1. In general, the larger the CM size, the longer the unmixing.

An additional time analysis of the first frames of sleeping1 is displayed in table 4.3. It indicates that the average frame unmixing time increases along with the number of non-transparent pixels shown in fig. 4.3. The more the program processes through the video, the longer the unmixing step takes. This interconnection between the computation time and percentage of non-transparent pixels confirms that the minimisation is slower when given partially transparent pixels already present on several layers.

sleeping1	No flow		Init custom flow		Init truth flow	
Frames	No GF	GF	No GF	GF	No GF	GF
0	86	86	84	86	86	88
1	93	93	104	111	105	116
2	92	92	110	117	113	116
3	93	93	114	117	117	118
4	93	93	116	116	116	118
5	91	91	118	116	119	119
6	93	93	114	116	119	118
7	91	91	115	120	118	118
Sequence avg	92	92	115	123	120	122

Table 4.3: Computation time in seconds of the frame minimisation step of the first 8 frames of Sintel sleeping1.

At the beginning of this dissertation, we expected the method with a layer initialisation approach to be faster as layers would initially have input values closer to their

convergence point. On the contrary, the average unmixing time is much higher. There again, it is due to the unexpected behaviour of the image-based soft colour segmentation and especially its minimisation which tends to spread a given pixel on more layers. The opposite action of restoring the pixel opacity and gathering back the pixel unmixing on fewer layers is not happening during the minimisation step. Therefore the time performance is greatly lower for both initialisation techniques in comparison to the independent frame computation as every partially transparent pixel unmixing takes longer to converge.

Conclusions and Analysis

This evaluation reveals that the minimisation step from the image-based soft colour segmentation by Aksoy et al. [1] tends to spread a given pixel unmixing to several layers. Therefore our initialisation technique with both custom and ground truth flows accentuates this behaviour by initialising the pixels before the unmixing with more and more transparent and diffused contributions.

The RGB smoothness is not distinctive as it seems to be more representative of the colour variances of the colour model. As for the alpha smoothness, it shows that No flow is better. The "spreading" behaviour is seen when observing the percentage of non-transparent pixels. Indeed more and more pixels are present in all layers with lower alpha values as the video elapses. Furthermore, the minimisation is longer when a pixel is present in several layers as it has to take into account the contributions of every layer. Finally the more colours in the colour model, the longer the unmixing.

The main limitation of this technique is due to the minimisation behaviour that expects fully opaque pixels and only diffuses the pixels when looking for convergence. Now that this behaviour is known, it would be more relevant to temporally filter the output layers in a post-minimisation step instead of the pre-minimisation initialisation, or change the minimisation process so that it can account for temporal information without introducing these artefacts.

4.2 Colour Model Estimation

4.2.1 Evaluation Process

The Colour Model (CM) estimation technique (section 3.1) is evaluated by comparing the four proposed techniques (see fig. 3.2) in terms of CM size, colour distributions, computation time and average reconstruction error per frame. The tested sequences contain six frames each, three from one of the Sintel scenes, and three from another. Hence the tested sequences all include a video cut between the third and fourth frames. The fig. 4.4 shows the different test sequences.

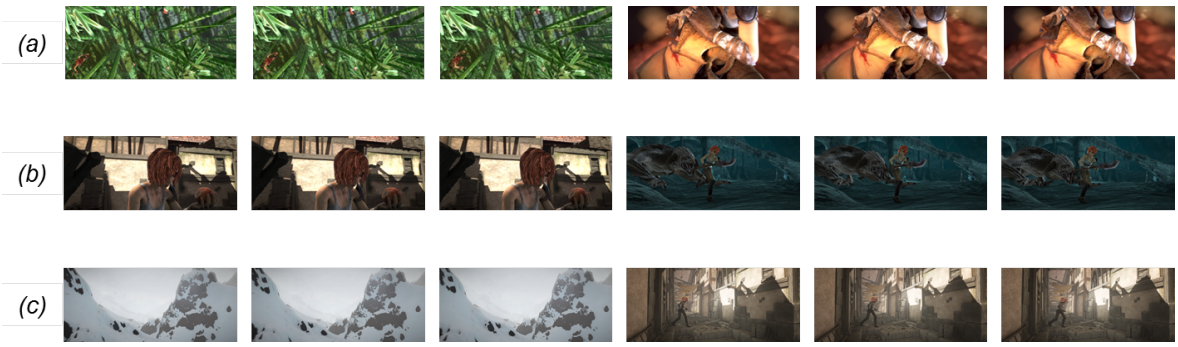

Figure 4.4: Sintel sequences used for the colour model evaluation: (a) combines bamboo1 and bandage1, (b) combines alley1 and cave2, and (c) combines mountain1 and alley2.

To compute the reconstruction error, the output layers are first added up together to reconstruct the sum image. The reconstruction error is calculated as the sum of euclidean distances of every pixel between the reconstructed and original images. It is defined as follows:

$$\epsilon_{rec} = \sum_i \sum_j ||I_{i,j} - S_{i,j}||_{L2}, \tag{4.2}$$

where $I_{i,j}$ is the initial image RGB value at pixel (i, j), and $S_{i,j}$ is the sum image RGB value at pixel (i, j). Having a low reconstruction error is a necessary condition to enable correct video manipulation such as colourisation or compositing.

The histogram comparison described in section 3.2.4 and used in the *Fixed CM with*

cut detection technique is evaluated on the "Big Buck Bunny" animated movie of the Blender Foundation www.blender.org because this video contains many cuts. When the colour histogram distances of two subsequent frames are higher than a certain threshold, a video cut is identified and the colour model is reset. The threshold value is empirically set to 0.4 by evaluating the histogram distances on 5,000 frames from the "Big Buck Bunny" animated movie. Among the 5,000 frames, all the video abrupt cuts are correctly detected. Gradual cuts where the video slowly fades to black are not detected by this current implementation. Their detection would necessitate a supplementary technique such as the twin-comparison method by Zhang et al. [57]. Additionally the computation time of this histogram distance per frame lasts about 0.5sec. Therefore, the colour histogram distance is time-efficient and it accurately detects abrupt cuts, which confirms its profitability.

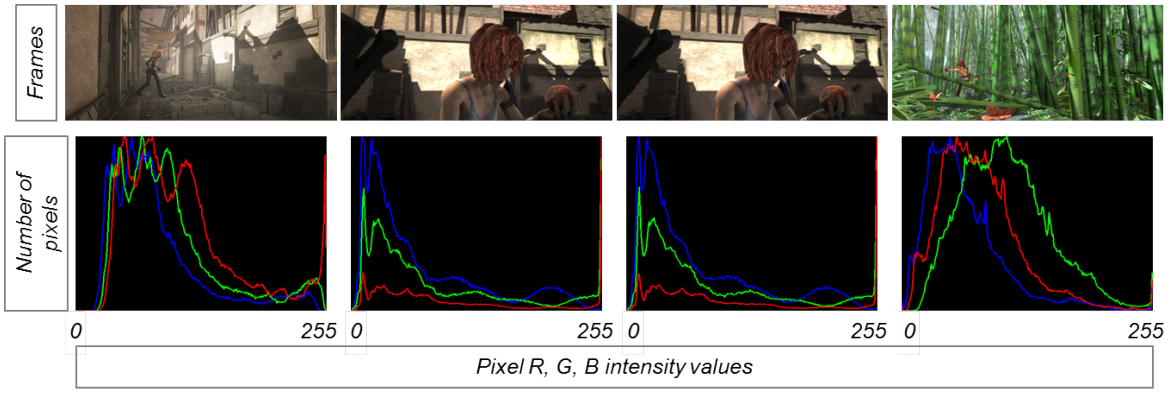

Figure 4.5: Histogram visualisation for a set of four Sintel frames. The red, green and blue histograms correspond to the normalised number of pixels present in each intensity bin of width one.

As an example, the normalised histograms for red, green and blue channels are displayed in fig. 4.5 for an artificially created Sintel sequence of four frames with two cuts. Their corresponding histogram distances are: 0.88 from frame 0 to frame 1, 0.04 from frame 1 to frame 2, and 0.78 from frame 2 to frame 3. It illustrates the quality of the histogram comparison metric as the two successive frames 1 and 2 have a very low distance, and the cuts between frames 0 and 1 as well as between frames 2 and 3 are correctly identified with the current threshold.

4.2.2 Results

CM Size and Colour Distributions

The Colour Models (CM) of the test sequence (a) are displayed in fig. 4.6 for all four techniques proposed.

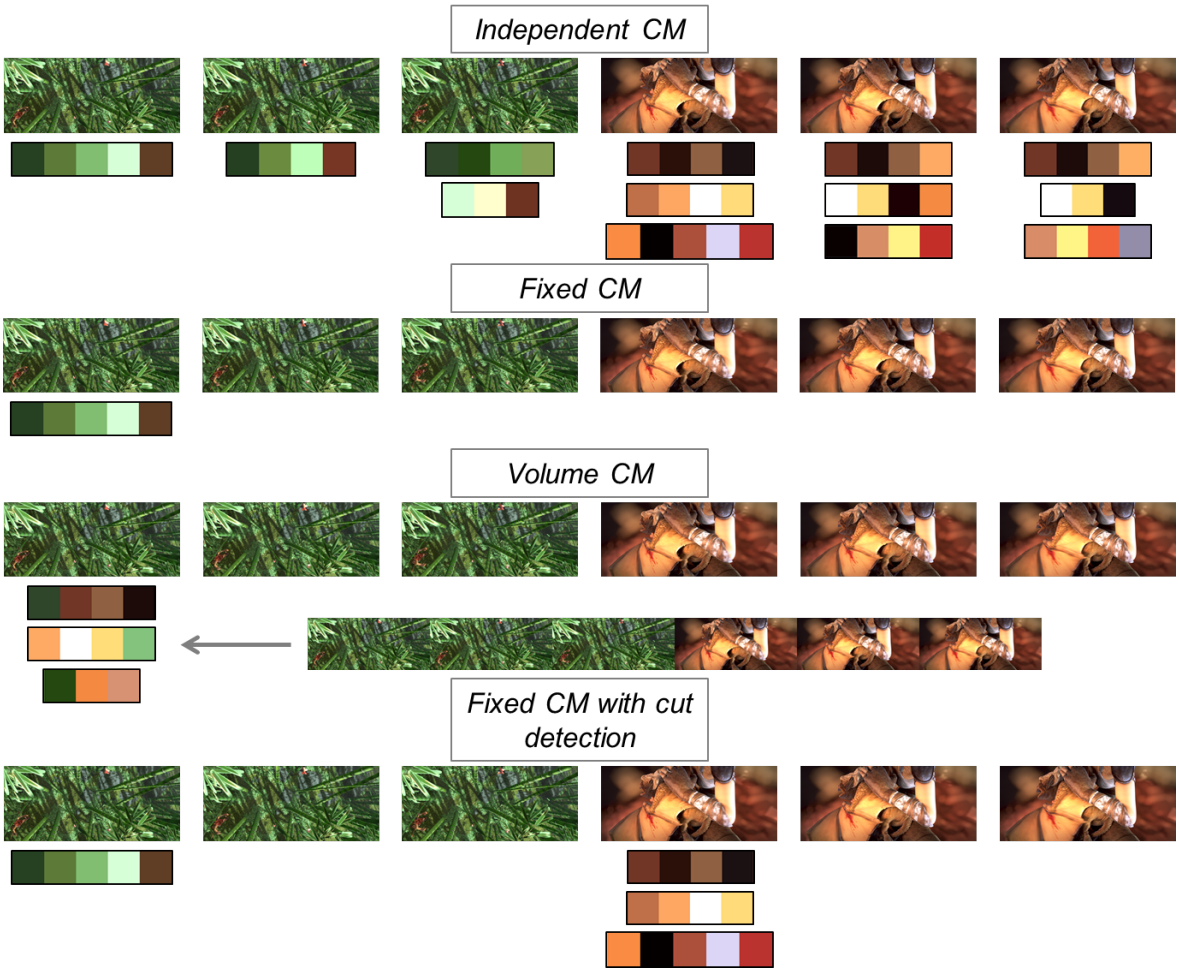

Figure 4.6: Output colour models of the four techniques on the test sequence (a).

Firstly a remarkable result on the *Independent CM* technique is that the output CM can have a very different number of colour distributions within the same video shot. Even though the three "bamboo" green frames look similar, the difference is high with a respective number of 5, 4 and 7. Similarly the three following "bandage"

orange frames have a CM of size 13, 12 and 11. The same behaviour appears in the (b) and (c) sequences, which proves that the CM computation is too unstable in its size. The colour distributions are also unstable as analogous frames mostly output distinct CM. With a different number of layers in each frame, it is impossible to obtain coherent video segmentation per layer. Therefore it is necessary to fix the colour model over a certain number of frames. This first brute force independent technique can then already be removed from the pool of appropriate video methods.

Both video shots of sequence (a) have different colour distributions so the *Fixed CM* technique is expected to face difficulties when minimising the bandage sequence using the green shades of the bamboo colour model. As for the *Volume CM*, the displayed colour model seems to enclose all the prevalent colour distributions of the video volume. Finally the cut is correctly identified in the last method *Fixed CM with cut detection*.

Time Performance

The table 4.4 summarises the computation time in the three tested sequences. Concerning the time performance, the *Fixed CM* with and without cut detection are the fastest at computing their colour model(s). The CM computation without cut detection is 15 times (a), 1.2 times (b), and 2.1 times (c) faster than the one with cut detection. The reason is that the CM is only computed once for the *Fixed CM* without cut detection and twice with cut detection. The *Volume CM* only estimates the colour model once from the full video volume but its computation lasts 101 times more (a), 10 times more (b), and 34 times more (c) when compared to the individual CM computation of *Fixed CM*. This demonstrates that the colour model computation scales badly for larger images.

Sintel sequence	Independent CM	Fixed CM	Volume CM	Fixed CM w/ cut detection
(a)	242	5	507	79
(b)	86	21	209	26
(c)	60	11	371	23

Table 4.4: Colour model(s) computation time in seconds for each test sequence.

An additional test on Sintel alley1 is given in table 4.5 with the CM computation time for different volumes. It further illustrates that the more frames that are stacked to compute the volume CM, the longer the estimation. Observing that only a second of video (24 frames) takes 467 seconds confirms that the *Volume CM* technique is unusable as it does not scale properly to larger video volumes.

Nb of frames	Size CM	Computation time CM
1	6	22
3	6	72
6	6	155
12	5	211
18	5	311
24	5	467

Table 4.5: Computation time in seconds for different numbers of frames in the CM volume.

Reconstruction Error

Let's analyse the reconstruction error of all techniques and especially of the two remaining techniques. The *Fixed CM* technique fixes the colour model for the full duration of the given video frames, while the *Fixed CM with cut detection* technique fixes a different colour model for each distinct shot of the video, if any. Table 4.6 summarises the reconstruction error in the three tested sequences.

Sintel sequence	Independent CM	Fixed CM	Volume CM	Fixed CM w/ cut detection
(a)	0.000630	0.000812	0.000902	0.000789
(b)	0.000123	0.000128	0.000234	0.000108
(c)	0.000164	0.000218	0.000192	0.000231

Table 4.6: Reconstruction error of the evaluated sequences.

All four techniques unexpectedly yield excellent reconstruction results. This is understandable for the independent, volume and fixed CM with cut detection approaches which strive to represent each frame of the sequence. However the fixed CM fails to

consider the colours from frames appearing after a video cut, thus the reconstruction error should be higher after the cut. A deeper analysis of the layers obtained for the *Fixed CM* and shown in fig. 4.7 clarifies this result. Indeed, when a frame very distinct from the current colour model is minimised, the pixels tend to converge all together towards the most corresponding layer distribution colour. The following paragraph explains this behaviour.

In the example figure, the input frame mainly encompasses hues of orange and therefore largely converges to the brown colour distribution while avoiding the green colour distributions. As most pixels can only be represented by the brown distribution, this technique additionally converges 1.4 times faster than with the cut detection. Nonetheless fig. 4.7 clearly shows an incorrect layer representation as only one layer comprises most of the input frame. These output layers cannot be utilised for any kind of video editing. Ultimately, the *Fixed CM* approach is a time-efficient technique raising no reconstruction error. But these misleading results hide an incorrect layer representation when a frame is segmented after a cut. Thus it is not usable for video manipulations.

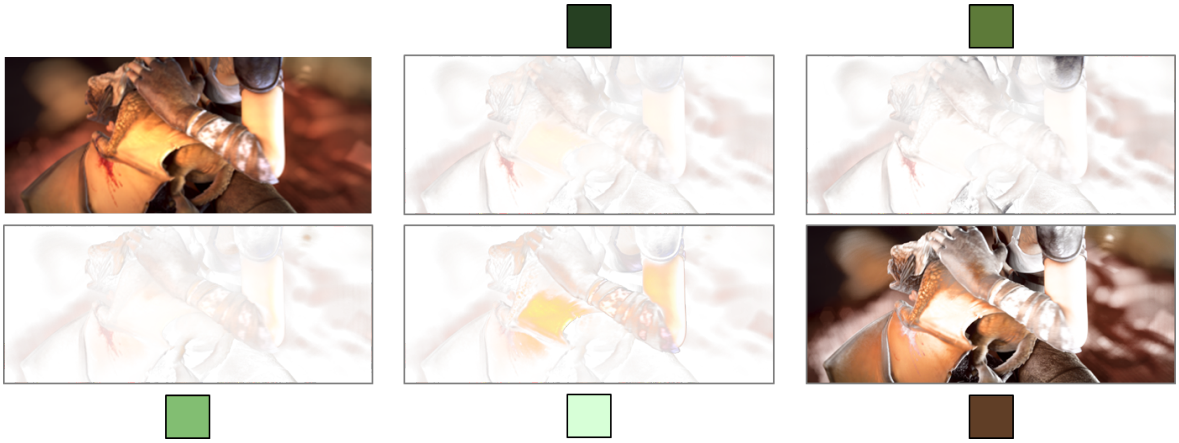

Figure 4.7: Top-left: Input frame after the cut. The other images are the output layers with corresponding colour model using the fixed CM technique.

Conclusions and Analysis

To conclude the analysis of the colour model estimation, all computation techniques yield layers with very low reconstruction errors. The *Independent CM* cannot achieve layer temporal coherency as the number of layers is often modified even for similar input frames. The *Volume CM* is representative but computationally intensive; indeed the more frames that are used to compute the volume CM, the longer the computation time. The fastest techniques are first the *Fixed CM* and second the *Fixed CM with cut detection*. However the *Fixed CM* technique produces an incorrect layer representation after a video cut. This leaves us with the *Fixed CM with cut detection* way which is time-efficient, produces output layers properly reconstructing the initial frame and possesses a correct layer representation contrary to the *Fixed CM*.

The colour model technique of computation from Aksoy et al. [1] shows an important variation in the number of colour distributions estimated. This is clear in fig. 4.6 with five colour distributions on the bamboo sequence but thirteen on the bandage sequence. Both also seem to possess very similar colour means, in particular the black and brown hues. This impacts the overall time performance for the minimisation: the larger the colour model, the longer the segmentation of each frame in the video sequence. A potential solution could be to set a limit on the number of possible colour distributions. However if the size of the colour model does affect the computation time, it does not reduce the output layers quality. Indeed when too many colour distributions are found, a post-processing step could simply combine the layers with close colour distributions and subsequent edits could then be applied on the combined layers.

On another note, the fixed CM with cut detection technique does not handle disappearing or appearing objects in the same video shot. So for instance let's consider a red apple coming into a scene where the red colour is not represented in the current colour model. Then the apple would either fully go to the closest, and potentially distant, colour distribution layer, either be a mixture of several layers or directly not converge. In all those cases, performing a layer colourisation or other editing would prove to be inappropriate for the new object(s). A solution could be to add an object detection in the code but that might be too computationally-heavy. Another way would be to allow for the user to inform the program when a new object with new colour distribu-

tion(s) appears. Afterwards, the program would automatically detect which colour(s) are missing and add them to the current colour model.

Chapter 5

Conclusion

5.1 Main Contributions

The proposed video soft colour segmentation method automatically extracts video colour layers. The program shows its coding robustness as it ran many times on several reference sequences with several comparative approaches.

The layer initialisation using optical flow technique refutes our initial thoughts that it would: 1. enforce temporal coherency in the layers and 2. speed up the minimisation step as pixels are already closer to their convergence values. Instead it demonstrates an unexpected behaviour of the image-based soft colour segmentation technique: the minimisation tends to spread a pixel unmixing to several layers. Therefore it induces a slow and poor convergence when initialised with non-opaque pixels, which is the case when the layers are initialised with the previous shifted layers.

As for the colour model estimation, fixing the colour model on each detected video cut performs the best. It allows for faster colour model computation and more coherent layers.

Ultimately, these two evaluations establish that the best performing case is a frame-independent segmentation associated with a fixed colour model with video cut detection. Additionally the post-minimisation spatial filtering tends to improve the layers quality.

5.2 Perspectives

Even though using optical flow in the initialisation step does not ensure temporal coherency, we still strongly believe that using optical flow is the best way to take into account the temporal motion of the video. Therefore we suggest a novel approach to utilise optical flow information in a post-minimisation spatio-temporal filtering step instead of the pre-minimisation initialisation. The Permeability Filter (PF) employed by Schaffner et al. [39] already used in our implementation to spatially and temporally refine the optical flows could be used to apply this temporal filtering step on the resulting layers. Hence the spatial and temporal PF would replace the previously used spatial Guided Filter (GF). As this filtering would affect the pixel unmixing values, an additional refinement step to minimise the energy function once again with different constraints would then be required. Another approach could also explore a different unmixing technique that would incorporate temporal consistency during minimisation and would therefore eliminate the need to filter afterwards.

The results regarding the colour model estimation work point out that the current technique is unable to detect appearing or disappearing objects as it is only conceived to identify video cuts. Possible future work could focus on detecting these appearing and disappearing objects in the video. Thereafter, instead of fully re-computing the colour model, the newly detected object(s) colour distribution(s) could be added to the current colour model. And conversely disappearing object(s) one(s) could be removed.

This dissertation work is a video soft *colour* segmentation. It would be interesting to investigate segmentation methods relying on other characteristics such as texture extraction [58][59], or object motion deblurring [60].

5.3 Final Thoughts

This master thesis was a very valuable experience. I learnt a lot in the fields of image and video processing, in particular the state-of-the-art methods to estimate optical flow or induce temporal filtering. Moreover, the whole conduct of the project gave me a broad insight into the research domain.

The topic of video soft colour segmentation is of great interest to me. These five months have confirmed that I would gladly work in animation, CGI, and VFX industries. The combination of computer graphics and artistic contents vastly appeals to me.

To conclude this dissertation, I would like to thank my supervisors, my family, my friends and all those I have met during my time in Dublin. They participated in making this master year a very rewarding and constructive one. I wish them all the best and look forward to seeing them again.

Bibliography

[1] Yağız Aksoy, Tunç Ozan Aydın, Aljoša Smolić, and Marc Pollefeys. Unmixing-based soft color segmentation for image manipulation. *ACM Trans. Graph.*, 36(2):19:1–19:19, 2017.

[2] E. Forgy. Cluster analysis of multivariate data: Efficiency versus interpretability of classification. *Biometrics*, 21(3):768–769, 1965.

[3] Yağız Aksoy, Tunç Ozan Aydın, Marc Pollefeys, and Aljoša Smolić. Interactive high-quality green-screen keying via color unmixing. *ACM Trans. Graph.*, 35(5):152:1–152:12, 2016.

[4] Dimitri P. Bertsekas. Chapter 2 - the method of multipliers for equality constrained problems. In Dimitri P. Bertsekas, editor, *Constrained Optimization and Lagrange Multiplier Methods*, pages 95 – 157. Academic Press, 1982.

[5] A. Levin, D. Lischinski, and Y. Weiss. A closed-form solution to natural image matting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):228–242, Feb 2008.

[6] Eduardo S. L. Gastal and Manuel M. Oliveira. Shared sampling for real-time alpha matting. *Computer Graphics Forum*, 29(2):575–584, May 2010. Proceedings of Eurographics.

[7] K. He, J. Sun, and X. Tang. Guided image filtering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(6):1397–1409, June 2013.

[8] Y. W. Tai, J. Jia, and C. K. Tang. Soft color segmentation and its applications.

IEEE Transactions on Pattern Analysis and Machine Intelligence, 29(9):1520–1537, Sept 2007.

[9] Jianchao Tan, Jyh-Ming Lien, and Yotam Gingold. Decomposing images into layers via RGB-space geometry. *ACM Transactions on Graphics (TOG)*, 36(1):7:1–7:14, November 2016.

[10] Anat Levin, Alex Rav-Acha, and Dani Lischinski. Spectral matting. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(10):1699–1712, October 2008.

[11] D. Singaraju and R. Vidal. Estimation of alpha mattes for multiple image layers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(7):1295–1309, July 2011.

[12] Qifeng Chen, Dingzeyu Li, and Chi-Keung Tang. Knn matting. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(9):2175–2188, September 2013.

[13] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision (darpa). In *Proceedings of the 1981 DARPA Image Understanding Workshop*, pages 121–130, April 1981.

[14] Berthold K.P. Horn and Brian G. Schunck. Determining optical flow. *Artificial Intelligence*, 17(1):185 – 203, 1981.

[15] T. Brox and J. Malik. Large displacement optical flow: Descriptor matching in variational motion estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(3):500–513, March 2011.

[16] L. Xu, J. Jia, and Y. Matsushita. Motion detail preserving optical flow estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(9):1744–1757, Sept 2012.

[17] Deqing Sun, Stefan Roth, and Michael J. Black. A quantitative analysis of current practices in optical flow estimation and the principles behind them. *International Journal of Computer Vision*, 106(2):115–137, Jan 2014.

[18] J. Kim, C. Liu, F. Sha, and K. Grauman. Deformable spatial pyramid matching for fast dense correspondences. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2307–2314, June 2013.

[19] J. Hur, H. Lim, C. Park, and S. C. Ahn. Generalized deformable spatial pyramid: Geometry-preserving dense correspondence estimation. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1392–1400, June 2015.

[20] Y. Hu, R. Song, and Y. Li. Efficient coarse-to-fine patch match for large displacement optical flow. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5704–5712, June 2016.

[21] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman. Patchmatch: A randomized correspondence algorithm for structural image editing. In *ACM SIGGRAPH 2009 Papers*, SIGGRAPH '09, pages 24:1–24:11, New York, NY, USA, 2009. ACM.

[22] K. He and J. Sun. Computing nearest-neighbor fields via propagation-assisted kd-trees. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 111–118, June 2012.

[23] S. Korman and S. Avidan. Coherency sensitive hashing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(6):1099–1112, June 2016.

[24] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In A. Fitzgibbon et al. (Eds.), editor, *European Conf. on Computer Vision (ECCV)*, Part IV, LNCS 7577, pages 611–625. Springer-Verlag, October 2012.

[25] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*, pages 839–846, Jan 1998.

[26] H. Takeda, S. Farsiu, and P. Milanfar. Kernel regression for image processing and reconstruction. *IEEE Transactions on Image Processing*, 16(2):349–366, Feb 2007.

[27] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. Image denoising by sparse 3-d transform-domain collaborative filtering. *IEEE Transactions on Image Processing*, 16(8):2080–2095, Aug 2007.

[28] Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:603–619, 2002.

[29] Roman Rosipal and Leonard J. Trejo. Kernel partial least squares regression in reproducing kernel hilbert space. *J. Mach. Learn. Res.*, 2:97–123, March 2002.

[30] P. Milanfar. A tour of modern image filtering: New insights and methods, both practical and theoretical. *IEEE Signal Processing Magazine*, 30(1):106–128, Jan 2013.

[31] Zeev Farbman, Raanan Fattal, Dani Lischinski, and Richard Szeliski. Edge-preserving decompositions for multi-scale tone and detail manipulation. *ACM Trans. Graph.*, 27(3):67:1–67:10, August 2008.

[32] Eduardo S. L. Gastal and Manuel M. Oliveira. Domain transform for edge-aware image and video processing. *ACM Trans. Graph.*, 30(4):69:1–69:12, July 2011.

[33] Manuel Lang, Oliver Wang, Tunc Aydin, Aljoscha Smolic, and Markus Gross. Practical temporal consistency for image-based graphics applications. *ACM Trans. Graph.*, 31(4):34:1–34:8, July 2012.

[34] Tunç Ozan Aydin, Nikolce Stefanoski, Simone Croci, Markus Gross, and Aljoscha Smolic. Temporally coherent local tone mapping of hdr video. *ACM Trans. Graph.*, 33(6):196:1–196:13, November 2014.

[35] Sylvain Paris, Samuel W. Hasinoff, and Jan Kautz. Local laplacian filters: Edge-aware image processing with a laplacian pyramid. *Commun. ACM*, 58(3):81–91, February 2015.

[36] Cevahir Cigla and A. Aydin Alatan. Information permeability for stereo matching. *Image Commun.*, 28(9):1072–1088, October 2013.

[37] Genzhi Ye, Elena Garces, Yebin Liu, Qionghai Dai, and Diego Gutierrez. Intrinsic video and applications. *ACM Trans. Graph.*, 33(4):80:1–80:11, July 2014.

[38] Nicolas Bonneel, James Tompkin, Kalyan Sunkavalli, Deqing Sun, Sylvain Paris, and Hanspeter Pfister. Blind video temporal consistency. *ACM Trans. Graph.*, 34(6):196:1–196:9, October 2015.

[39] M. Schaffner, F. Scheidegger, L. Cavigelli, H. Kaeslin, L. Benini, and A. Smolic. Towards edge-aware spatio-temporal filtering in real-time. *IEEE Transactions on Image Processing*, 27(1):265–280, Jan 2017.

[40] Jiawen Chen, Sylvain Paris, and Frédo Durand. Real-time edge-aware image processing with the bilateral grid. *ACM Trans. Graph.*, 26:103, 2007.

[41] Holger Winnemöller, Sven C. Olsen, and Bruce Gooch. Real-time video abstraction. *ACM Trans. Graph.*, 25(3):1221–1226, July 2006.

[42] D. Wang. Unsupervised video segmentation based on watersheds and temporal tracking. *IEEE Transactions on Circuits and Systems for Video Technology*, 8(5):539–546, Sep 1998.

[43] Jaemin Kim and J. W. Woods. Spatio-temporal adaptive 3-d kalman filter for video. *IEEE Transactions on Image Processing*, 6(3):414–424, Mar 1997.

[44] A. J. Patti, A. M. Tekalp, and M. I. Sezan. A new motion-compensated reduced-order model kalman filter for space-varying restoration of progressive and interlaced video. *IEEE Transactions on Image Processing*, 7(4):543–554, Apr 1998.

[45] Allison W. Klein, Peter-Pike J. Sloan, Adam Finkelstein, and Michael F. Cohen. Stylized video cubes. In *ACM SIGGRAPH Symposium on Computer Animation*, pages 15–22, July 2002.

[46] Daniel Dementhon and Remi Megret. Spatio-temporal segmentation of video by hierarchical mean shift analysis. In *Center for Automat. Res., U. of Md, College Park*, 2002.

[47] Jue Wang, Bo Thiesson, Yingqing Xu, and Michael Cohen. Image and video segmentation by anisotropic kernel mean shift. In Tomás Pajdla and Jiří Matas,

editors, *Computer Vision - ECCV 2004*, pages 238–249, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[48] S. Khan and M. Shah. Object based segmentation of video using color, motion and spatial information. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, volume 2, pages II–746–II–751 vol.2, 2001.

[49] C. W. Zitnick, N. Jojic, and Sing Bing Kang. Consistent segmentation for optical flow estimation. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 2, pages 1308–1315 Vol. 2, Oct 2005.

[50] W. Brendel and S. Todorovic. Video object segmentation by tracking regions. In *2009 IEEE 12th International Conference on Computer Vision*, pages 833–840, Sept 2009.

[51] M. Grundmann, V. Kwatra, M. Han, and I. Essa. Efficient hierarchical graph-based video segmentation. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2141–2148, June 2010.

[52] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181, Sep 2004.

[53] Eckehard Steinbach, Peter Eisert, and Bernd Girod. Motion-based analysis and segmentation of image sequences using 3-d scene models. *Signal Processing*, 66(2):233 – 247, 1998.

[54] Xue Bai and Guillermo Sapiro. Geodesic matting: A framework for fast interactive image andvideo segmentation and matting. *International Journal of Computer Vision*, 82(2):113–132, Apr 2009.

[55] Blender institute. https://www.blender.org/institute/.

[56] Irena Koprinska and Sergio Carrato. Temporal video segmentation: A survey. *Signal Processing: Image Communication*, 16(5):477 – 500, 2001.

[57] HongJiang Zhang, Atreyi Kankanhalli, and Stephen W. Smoliar. Automatic partitioning of full-motion video. *Multimedia Systems*, 1(1):10–28, Jan 1993.

[58] Majid Mirmehdi and Maria Petrou. Segmentation of color textures. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(2):142–159, February 2000.

[59] Sai-Kit Yeung, Tai-Pang Wu, and Chi-Keung Tang. Extracting smooth and transparent layers from a single image. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–7, June 2008.

[60] J. Pan, Z. Hu, Z. Su, H. Y. Lee, and M. H. Yang. Soft-segmentation guided object motion deblurring. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 459–468, June 2016.