Object Flow

A per-object dense motion descriptor

Juan Manuel Pérez Rúa

Supervised by: Tomas Crivelli and Patrick Pérez

Technicolor SA

A Thesis Submitted for the Degree of
MSc Erasmus Mundus in Vision and Robotics (VIBOT)

· 2014 ·

Abstract

Motion analysis in image sequences has undoubtedly shown good progress in terms of its two main research branches. Optical flow estimation and object visual tracking have been mostly studied as isolated problems, and high accuracy algorithms are available when needed as independent bricks. This thesis presents a framework for combining object tracking techniques with optical flow methods aiming towards a precise motion description for objects in video sequences. Firstly, we introduce a method to segment objects in videos, which is done by exploiting the inherent foreground-background separation hints given by object trackers, and the novel concept of superpixel flow, which is used to perform background regions tracking. Then, it is shown that long-motion awareness obtained from object tracking, together with a per frame object segmentation can improve the precision of the object motion description in comparison to several optical flow techniques. The proposed approach is called Object flow as it offers a dense and semantic aware description of the current apparent motion state of the studied object.

Todo arde si le aplicas la chispa adecuada. . . .

Enrique Bunbury

Contents

List of Figures

Chapter 1

Introduction

Object tracking and optical flow are two of the main components in the computer vision toolbox, and have been focus of great research efforts, leading to significant progress in the last years [16] [17]. The object tracking problem in videos consists on estimating the position of a target in every frame, given an initial position. On the other hand, the optical flow between a pair of frames consists on finding a displacement vector for each pixel of the first image, namely a *dense motion or displacement field*. Even though for several applications a complete (i.e. for every pixel) motion-field is needed, other applications like human-computer interaction, object editing in video or structure-from-motion, may only focus on an interest object and thus, only a subset of motion vectors is required. In such scenarios combining optical flow and object tracking in a unified framework appears useful and the precision of the object motion description could be enhanced. For instance, even with modern optical flow approaches, the long term dense motion estimation remains a challenge [20] [22]. At large, object trackers provide a more robust, longer term motion estimation featuring a global description of an object, specially after recent works based on tracking-by-detection approaches [16] [23] [24]. On the other side, they lack the (sub) pixel precision of dense optical flow estimators, as well as a deeper use of contextual information for bundle motion vector estimation. Even more, object trackers and optical flow give precious hints for other fundamental tools such as object segmentation in video. Nevertheless, these two techniques were not deeply studied in the literature as a unified problem. Though optical flow has been widely used as a motion feature for object tracking [25], feeding a dense motion estimator with tracking information is not being fully exploited. This being said, we introduce a new problem which we call *object flow*. Thus, for a given object of interest, the object flow is the set of displacement vectors for every pixel that belong to the target in a first frame, towards another frame of the sequence. In other words, a dense displacement field constrained to the spatial support of the object. Note that by definition this induces a segmentation of the target

and of the motion field.

1.1 Problem definition

We can define more precisely the object flow by starting with an image sequence, say $I_t, t : 0..N-1$, and an initial position of the interest object in the first frame of this sequence. See Fig. 1.1 for a simple diagram. Let $\mathcal{R} \in \Omega$ be the region corresponding to the support of the object in the bi-dimensional grid Ω. Then, the object flow, $\mathcal{O}(x)$, is defined as $\mathcal{O}(x) = d_{0,t}(x), \forall x \in \mathcal{R}$. Where $d_{0,t}(x)$ is a displacement vector from the image 0 to the image t in the point x. Thus, the object flow problem consists on finding the displacement vector field inside the object support.

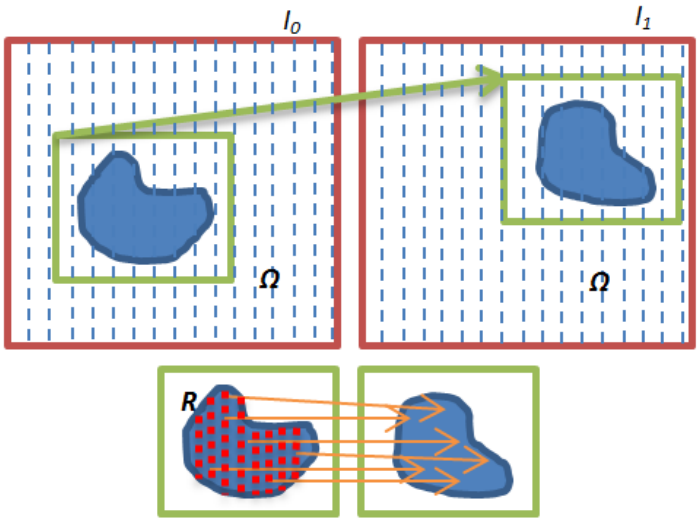


Figure 1.1: Object flow definition diagram. Top: Original frames with the tracker state, which indicates the object global motion (green arrow). Bottom: The object region is identified, and a dense motion field is obtained within its boundaries.

A straightforward solution to this problem would be to compute the optical flow motion field, and apply a segmentation mask to recover the desired motion vectors. Nevertheless, this approach carries several problems. For example, a globally computed optical flow method can affect small objects motion, because of the common use of heavy regularization prior. Moreover, even if the segmentation mask is extracted from a tracker window by an appearance based segmentation method, is likely that this mask is not going to be well suited for the interest object and some extra user interaction would be needed to refine this process. We propose an approach to reduce these problems.

1.2 Objectives

The main goal of this work is to define a dense motion descriptor for objects in video sequences. In order to achieve this goal, some specific objectives are defined:

- To show experimentally that the idea of mixing tracking techniques with optical flow methods enhance the precision of per-object motion description.
- To define an automatic segmentation approach for target objects in video sequences.
- To establish a flow estimation method that uses the segmentation mask provided and the region of interest given by the tracker.
- To show experimentally that the proposed object flow method is indeed more precise than regular optical flow techniques.
- To tackle applications such as automatic video editing and structure from motion which benefit from the object flow.

1.3 Document organization

The Section 2 starts by introducing the reader in the important topics and concepts that support the rest of this work: object tracking, optical flow and object segmentation. The concept of the superpixel flow is introduced in the Section 3. After this, in the Section 4, the object flow pipeline is explained in deep, together with the proposed algorithms. Then, a list of results and implementation details are discussed in Section 5, and applications of the object flow are discussed in Section 6. Finally, some insights and remarks are commented in Section 7.

Chapter 2

Background and Related Work

2.1 Introduction

The background of the object flow concept is mainly related to object tracker techniques and optical flow methods. A short state of the art review is presented in following sections, without going too deeply in any specific approach, since those are widely diverse. Of course, another important point is the object segmentation problem in videos. Several works have discussed this or similar problems in the state of the art, and only the more related approaches are presented. On the other hand, as the object flow itself is a novel problem, the related work is not large. However, in terms of applications, some works have been done towards specific oncomings to this concept. Some of these are superficially explained.

2.2 Object Tracking

As one of the most studied computer vision problems, object tracking has been constantly evolving since its early approaches. As the techniques were getting better, the benchmarking has been evolving too. The last global work in online object tracking evaluation was the work of Wu et al. [16] in 2013. Several remarks can be extracted from the deep benchmarking analysis of this work. For instance, it seems to be more clear that background information is a key hint towards better tracking methods. Some kind of modelling of the background should be an important step in the tracking pipeline. Whether this background modelling is implicit like in [23] or explicit like in [22], where is used as context, is just matter of design. Of course, this background modelling have to be accompanied with a local model to account with variations (occlusions, deformations) in the interest object. It seems that these two are the reasons why

Figure 2.1: Sequences used to evaluate object trackers in [16].

tracking by detection and learning approaches are among the most successful ones. Usually, these methods account with local and background modelling ([22] [23] [24] [25] [26]). However, even when motion or dynamical models are crucial for object tracking, very few of the last state of the art techniques focus on this element. The Fig. 2.2 shows results for the top 15 state of the art trackers as presented in [16], taking into account the variability of different initialization. According to this data, the tracking problem is still open and further exploration can be done. Moreover, it has to be observed that tracking by detection methods are indeed in the top of results in stability in initialization and peak performance. The Struck tracker [23], for instance, seems to hold the higher position w.r.t stability. Another curious observation is the fact that even traditionally good color based particle filter approaches are left behind by last years object trackers.

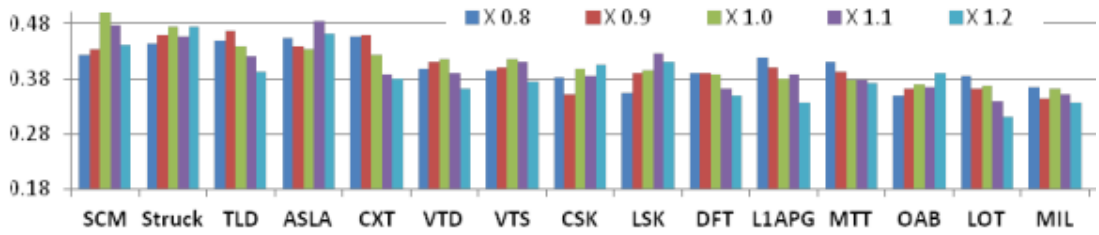


Figure 2.2: Performance summary for the top 15 trackers benchmarked in [16], initialized with different size of bounding box by factors of 0.8, 0.9, 1.0, 1.1 and 1.2.

2.2.1 Tracking-by-detection methods

Current tracking methods look at the problem as a classification task and use online learning techniques to update the object model [23]. The main reasons for the success of these approaches are the recent advances in object detection methods (*boosting*, *SVM*) and fast learning algorithms. The normal pipeline of these methods includes to sample the image to create a list of labelled training examples.

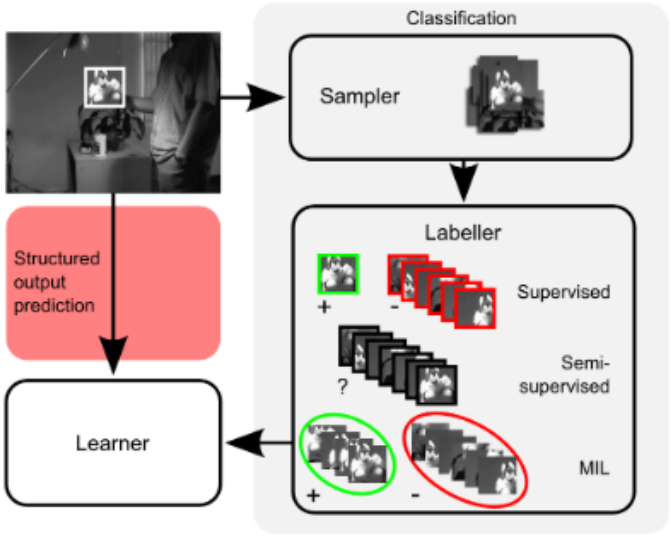


Figure 2.3: Usual approaches for tracking-by-detection methods. Right: generate a set of samples and, depending on the type of learner, produce training labels. Left: Structured output methods avoid these steps by operating directly in the tracking output. Extracted from [23].

Usually, the position of the object is computed from the sample with the maximum classification score in a local search window. This method, however, is not perfect, and most of the methods focus on increasing the robustness of the classifier to poor labelling and sampling, by including semi-supervised learning, robust statistics or multiple-instance learning [25]. Other approaches deal with these problems in a whole different way. For instance, the use of structured learning seems to outperform most of the tracking-by-detection methods [23]. In this way, the tracking is defined directly as predicting the change in object location between frames, by the use of structured output spaces (the same that is used in applications like text translation). Fig. 2.3 shows a diagram of most used approaches to tracking-by-detection methods.

2.3 Optical Flow

Figure 2.4: Frame pairs used to evaluate optical flow methods in [17], Ground truth flow coded with the proposed flow coding.

Optical flow is another traditional problem in computer vision, and it may be argued that its under-determined nature makes it one of the most difficult ones. Several well known databases and benchmarks have been proposed to account for the different intrinsic complications of the problem. The last and larger benchmarking work for this problem may be Baker et al. [17], but other useful studies are available [27]. These works reveal that most of the existing methods establish the problem as the optimization of an energy function that takes into account two terms (Eq. 2.1). The first one, E_{data} measures data consistency over input frames, and E_{prior} which favours flow with certain characteristics, e.g. smoothness in the flow. It seems that one of the major challenges remains to be the best pair of objective function and optimization method. A number of combinations have been proposed [29] [30] [32]. However, totally different approaches had been also proposed, e.g. by using polynomial expansions [28].

$$E_{global} = E_{data} + \lambda E_{prior} \tag{2.1}$$

According to the update of 2011 of the work in [17], the first 15 optical flow methods by interpolation error are shown in the Fig. 2.5. It has to be noted that the performance in several of the dataset is already very good for the most of these methods.

Method	Avg	Avg IE by dataset							
		Mequ.	Scheffl.	Urban	Teddy	Backyd.	Basktb.	Dumptr.	Evergr.
CBF	3.5	2.3	5.3	1.3	3.3	4.0	3.0	3.7	5.3
Aniso. Huber-L1	4.6	4.0	11.3	2.3	4.0	8.3	1.7	1.0	4.0
Second-order prior	5.5	3.3	8.0	6.0	3.0	6.3	4.3	3.0	9.7
Brox et al.	6.3	5.7	3.0	4.7	3.3	2.3	14.0	16.3	1.0
F-TV-L1	7.1	14.7	11.0	5.0	7.7	4.0	2.7	5.7	6.0
Filter Flow	9.7	10.7	16.0	9.0	9.3	5.3	9.7	7.0	10.3
Fusion	10.0	4.7	2.0	6.3	6.7	13.3	21.3	10.0	16.0
Black & Anandan	10.1	12.7	17.7	15.7	12.3	4.0	7.7	7.7	3.0
DPOF	10.2	15.0	1.0	15.3	6.7	13.7	9.0	8.7	12.0
2D-CLG	11.0	8.0	15.7	9.7	12.3	17.3	6.7	13.3	5.0
Horn & Schunck	11.1	9.0	20.0	13.7	16.3	4.7	5.3	13.0	7.0
Adaptive	12.5	11.7	16.7	7.0	12.0	14.3	14.3	12.0	12.0
Complementary OF	12.5	13.3	4.3	19.0	13.0	14.7	9.0	11.0	15.3
TV-L1-improved	12.8	8.3	15.3	11.0	5.0	11.7	18.3	18.3	14.3
Graph Cuts	13.0	17.0	5.3	14.0	12.0	15.7	10.3	15.7	13.7

Figure 2.5: Performance summary for the top 15 opt. flow methods benchmarked in [17] by interpolation error.

2.3.1 Simple Flow

As the Simple Flow [21] is the optical flow technique that is used as base of the proposed object flow method, it is presented in more detail. The main characteristic of this method is its efficient approach, which tries to concentrate in the zones where there is evidence of motion, and uses linear interpolation for excluded zones (Fig. 2.6). Moreover, the problem is not solved in the usual way by minimizing a variation of (2.1), but using a simpler likelihood model that follows the constant-color assumption, without including explicitly the pairwise terms that account for smoothing. The smoothness prior it is taken into account, however, by implementation of local filters that uses pixel weighting to take into account pixel proximity (w_d) and color similarity (w_c), leading to the Eq. 2.2, which tries to find the vector (u, v) that explain the motions in the point (x_0, y_0), but also in its neighbourhood $\mathcal{N}_0$.

$$E(x_0, y_0, u, v) = \sum_{(x,y)\in\mathcal{N}_0} w_d w_c ||I_0(x, y) - I_1(x + u, y + v)||^2 \tag{2.2}$$

The practical implementation of this equation requires a cross-bilateral filter to compute E, as it takes into account pixel (x_0, y_0) centred $n \times n$ windows between I_0 and I_1, producing a n^2-dimensional vector for each pixel. The flow is the vector (u_0, v_0) that minimizes E, producing an integer value. The precision can be enhanced by fitting parabolas at (u_0, v_0) and extracting the minimum. A final bilateral filtering is applied to the flow field, discarding occluded pixels (determined by cross-matching using the computed forward and backward flows). To recover

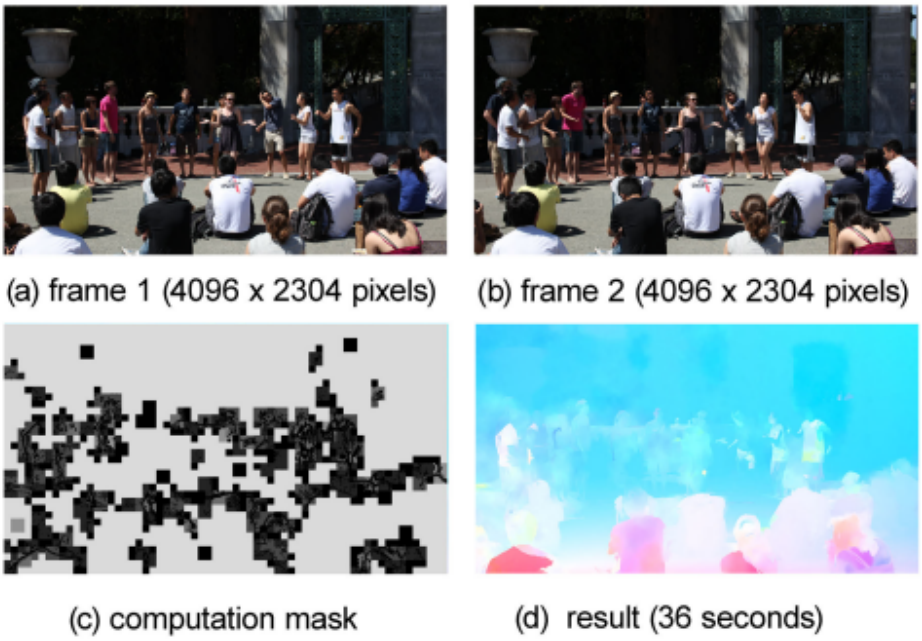


Figure 2.6: Results of the Simple Flow method in several datasets.

large motions, instead of using large $n \times n$ windows, a multi scale approach is followed. If more accuracy is desired, a more global optimization can follow, for instance, connecting the global approach used in Sun et al. [40]. This simple approach for computing optical flow ranked in a very good position in the Middlebury evaluation [17], and some results can be appreciated in the Fig. 2.7.

2.4 Object segmentation in video

Among the state of the art segmentation methods for objects in video sequences, point trajectories based ones stand for its performance and reliability [33], even when only sparse trajectories are known because of computational reasons [34]. On the other hand, for the problem of extracting out a preselected object in still frames, max-flow min-cut based approaches have demonstrated to be a powerful tool [14] [18].

The Fig. 2.8 shows high accuracy segmentation results when using sparse motion trajectories [34]. However, to overcome the sparsity of the flow the authors proposed a variational method to densify the output mask. The information propagation is done by presenting the problem as a non-linear diffusion process that takes superpixels into account. It is expected, then, that

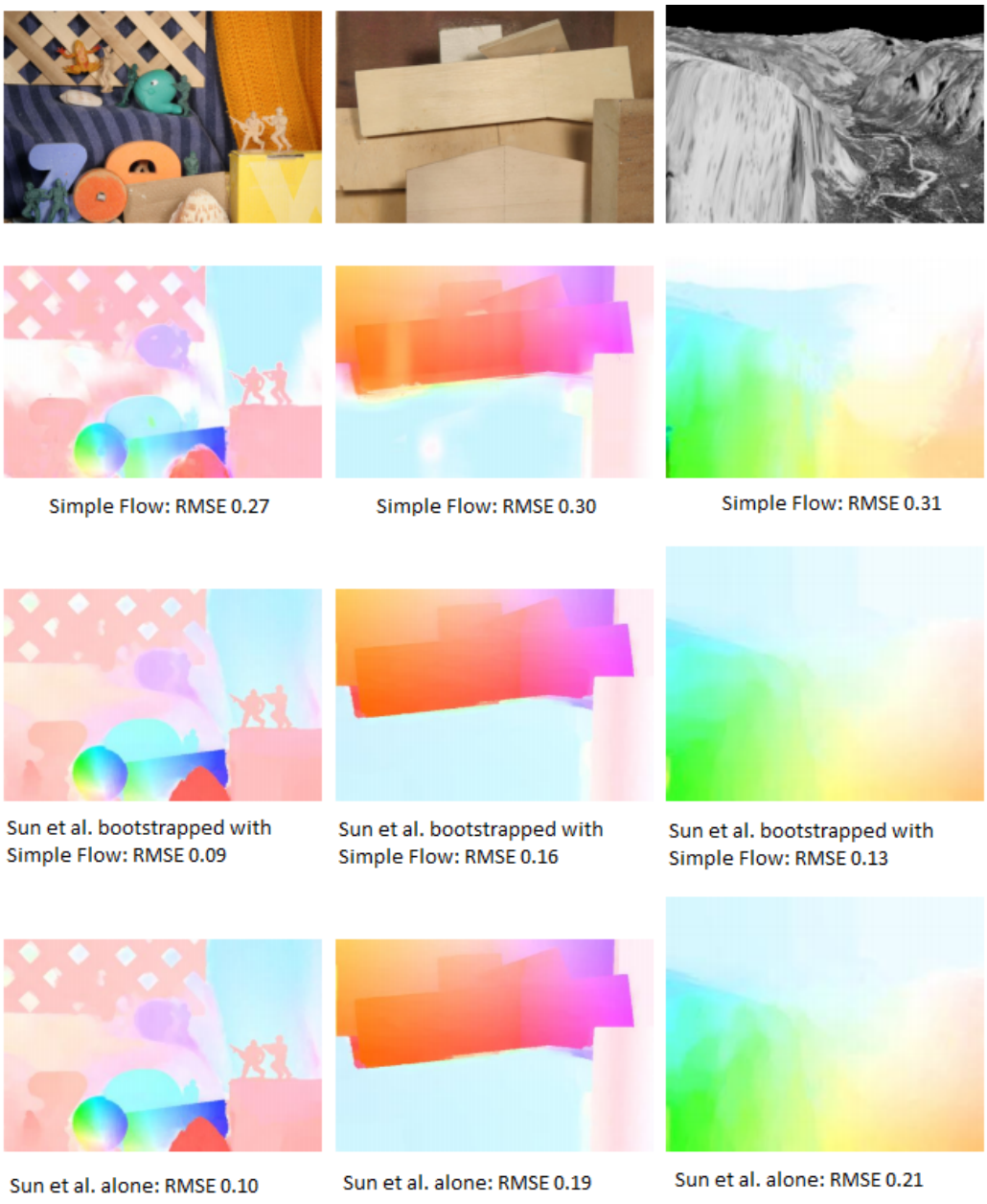


Figure 2.7: Results of the Simple Flow method in several datasets.

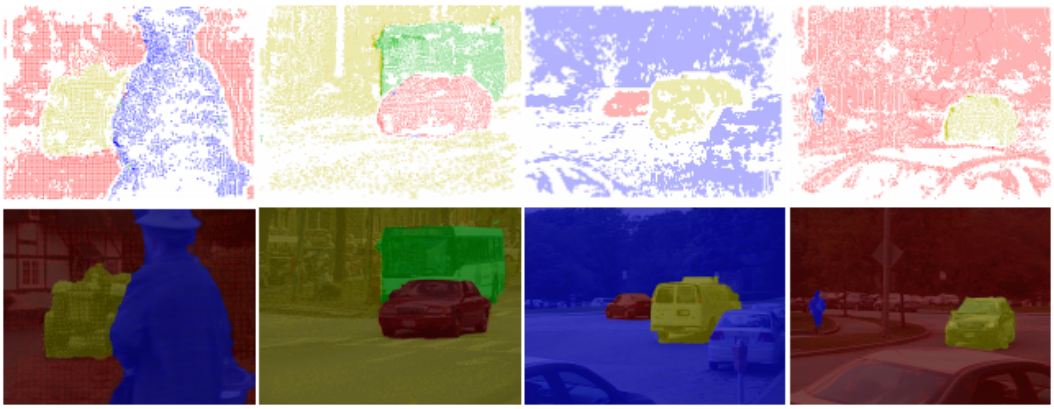


Figure 2.8: Sparse motion trajectories for segmentation. Results in several datasets [34].

this method can be precise, but slow. In this work we propose to mix single frame graph based segmentation approaches with the idea of background regions tracking. However, the sparsity of point trajectories is overcame by using superpixels. Therefore, instead of tracking points, background regions are tracked via the novel concept of superpixel flow. We show in future sections how this extra information can be used to complement the graph based techniques for an efficient foreground-background segmentation.

2.5 Related work

As previously mentioned, the related work is commented mostly from an application-wise point-of-view, due to the fact that the object flow concept is novel. The most used approaches in the literature to mix motion estimation with scene analysis are related to the use of this motion estimation to perform segmentation of scene [33] [34]. These works, however, depend intrinsically in the accuracy of the motion estimation to perform a good segmentation. Other authors have proposed to use simple low-order parametric motion models of the background movement, and by incorporating radial maps, obtain a frame-by-frame moving object segmentation [36], in contrast with the methods that combine motion awareness with appearance information [35].

From other point of view, regarding the problem of large displacement motion estimation, the use of superpixels seems to provide a reliable hint. The work presented in [39] provides an interesting method to combine the idea of optical flow with superpixels, to obtain an optical flow method which can perform well in difficult datasets [27]. The full set of these ideas can lead to some conclusions towards the object flow pipeline proposal. For instance, authors have already exploited the optical flow to improve tracking methods. It remains unexplored to complete the

cycle by improving the optical flow with the state of the tracker. Moreover, these works have shown that the use of superpixels can be combined with motion estimation to obtain both an object based segmentation and the enhancement of the motion itself.

On the other hand, other authors had used the optical flow constraints to track objects, which is, up to some extent, the inverse problem proposed for the object flow [37]. Some authors went further in this sense, to create specialized trackers for deformable objects by combining model based tracking with the optical flow constraint [38]. Even when [38] is very interesting for video editing tasks, the approach is limited by the availability of the mesh model. In this sense, the object flow proposal is more generic, and thus, more adequate for this kind of applications. Fig. 2.9 shows some results of [38] for the video edition tasks that can also be approached with the object flow. These results are obtained in a controlled environment without long range motion.

(a) Captured scene (b) Color of the logo changed (c) Logo replaced by another logo

Figure 2.9: Video edition with the method in [38].

In the same line of applications, an interesting work is the Unwrap Mosaics [41] which is a very different way to approach the video edition task.

The Unwrap Mosaics [41], is based in a new video representation, which is presented by defining a 2D-2D transformation supported in a segmentation mask (given by hand), and it is used to apply the desired editing in the interest object, and then, given the computed mapping, the edition can be back propagated to the initial sequence. Fig. 2.10 shows the video edition pipeline derived from this method. However, as it relies in precomputed segmentation and point trajectories, the Unwrap Mosaics can be mixed with the Object Flow pipeline to provide a robust and complete video edition framework.

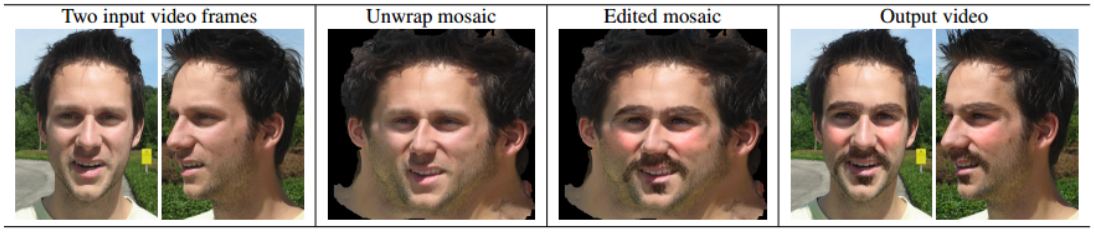


Figure 2.10: Video edition with the method in [41].

The object flow pipeline is, then, explained in detail in following sections. Some concepts are introduced first however. For instance, the superpixel flow and its mathematical modelling are deeply exposed in the next section. The proposed segmentation method is also explained later, as part of the object flow pipeline.

Chapter 3

Superpixel flow

During the initial stages of the object flow pipeline definition, the use of superpixel methods was considered. However, as the object flow refers mainly to the problem of estimating motion for objects in video, the superpixel method studied had to be able to maintain consistency between frames. Thus, usual approaches for time-consistent superpixels were studied. These methods carried some disadvantages, as explained later, and a novel idea had to be introduced in order to develop the concept of the object flow as it was being thought. This novel idea is the problem of matching precomputed superpixel between frames, by maintaining the time-smoothness which is natural of videos. This is, to incorporate the idea of flowing pixels to the superpixel level.

3.1 Problem definition

Superpixels and over segmentation techniques became a widely used pre-processing stage for a large number of machine vision applications, after the original concept was introduced [1]. Superpixels are traditionally used as performance booster for several other techniques. However, it is still mostly related to single frame processing [1] [10] [11]. In the search for consistency in superpixel labelling through video, some authors have proposed different techniques, which go from simple extension to supervoxels [9] [11], to more complicated approaches [8]. These approaches, nonetheless, usually require a global processing and knowledge of all (or several of) the video frames beforehand, which would affect the desired online characteristic of the target application, in this case, of the object flow.

Thus, as a prerequisite concept in the object flow pipeline, we propose a superpixel matching technique which assumes a flow-like behaviour in the image sequences (natural video), which can be used to track superpixels. Some previous work have been done towards a superpixel based image comparison using the Earth Mover's Distance, by taking superpixels as bins of

a global histogram [2]. The label propagation or superpixel flow can be achieved with this technique as a by-product, by selecting the superpixel in the second frame that maximize the EMD flow from each superpixel in the first frame. By taking into account superpixels computed separately in images, so the video process can be performed with only two frames at a time, we move towards a more time efficient approach. This matching, however, has to comply with a set of constraints. Firstly, two correspondent superpixels should be similar in terms of some appearance feature, which most likely depends on the way the superpixelization was performed (color, texture, shape). Also, the superpixel flow should maintain certain global regularity (at least for superpixels that belong to the same object). In this sense, it seems natural that the problem of superpixel flow could be solved with a discrete energy minimization procedure. If the size compactness of the superpixels is maintained, it actually seems to share some of the properties of the optical flow problem, with the difference that the smoothness is usually a very strong constraint for the last one. The strength of this smoothness prior relies not only in the nature of the problem, but also because it gives better cues towards an easier-to-minimize global approach.

The objective of the superpixel flow is therefore to find the best labelling l for every superpixel p (with $l_p \in 0, 1, ...M-1$, between a pair of frames (I_0,I_1), but holding a flow-like behaviour. M is the number of superpixels in the second frame.

Thus, the superpixelization should maintain certain size homogeneity within a single frame. Some super pixel techniques can cope with this requirement [9] [10]. For the experiments presented in this work, the SLIC method [9] is preferred, because it usually gives good results in terms of homogeneity of the superpixelization across the sequence. The proposed steps to solve the propagation problem assume this requirement is hold. For other kind of the techniques, other approaches should be followed.

3.2 Energy Formulation

Inspired by a large number of optical flow and stereo techniques [7] [12] [13], the superpixel flow can be modelled with a pairwise Markov Random Field in the super pixel space Γ. If the matching is performed with MAP inference, its posterior probability is:

$$P(l|I_0, I_1) = \prod_{p \in \Gamma} \mathrm{e}^{-D_p(l_p; I_0, I_1)} \prod_{(p,q): q \in \mathcal{N}_r} \mathrm{e}^{-S_{p,q}(l_p; l_q)}, \tag{3.1}$$

With l the set of labels of the super pixels in I_0, that match with those in I_1. $\mathcal{N}_r$ is a neighbourhood of the superpixel p containing all the superpixels which are inside a circle of radius r with center in p_c. This neighbourhood defines its adjacency, and its not of a constant

size, like in the pixel equivalent of these kind of equations, where a 4 or 8 neighbourhood is defined. Given this posterior probability, the equivalent energy function can be directly obtained by extracting the negative logarithm of the posterior,

$$E(l) = \sum_{p \in \Gamma} D_p(l_p; I_0, I_1) + \sum_{(p,q): q \in \mathcal{N}_r} S_{p,q}(l_p, l_q). \tag{3.2}$$

The terms D, and S in (3.2) stand for data term and spatial smoothness terms as they are popularly known in the MRF literature. The first one determines how accurate is the labelling in terms of consistency of the measured data (color, shape, etc.). In the classical optical flow formulation of this equation, the data term corresponds to the pixel brightness conservation [7] [5]. However, as superpixels are a set of similar (or somehow homogeneous) pixels, an adequate appearance based feature can be a low dimensional color histogram with N bins. So D can be written more precisely as the Hellinger distance between the histograms:

$$D_p(l_p; I_0, I_1) = \sqrt{1 - \frac{1}{\sqrt{\bar{h}(p)\bar{h}(p')N^2}} \sum_i \sqrt{h_i(p)h_i(p')}} \tag{3.3}$$

Where $h(p)$ and $h(p')$ are the histograms of the superpixel p and its correspondent superpixel in the second frame I_1. Note that the low dimensional histogram ($N = 2, N = 3$) gives certain robustness against noise, and slowly changing colors between frames.

On the other hand, the spatial term is a penalty function for horizontal and vertical changes of the vectors that have origin in the centroid of the superpixel of the first frame and end in the centroid of the superpixel of the second frame.

$$S_{p,q}(l_p, l_q) = \lambda(p)\sqrt{\frac{|u_{p_c} - u_{q_c}|}{\|p_c - q_c\|} + \frac{|v_{p_c} - v_{q_c}|}{\|p_c - q_c\|}} \tag{3.4}$$

$$\text{where, } \lambda(p) = (1 + \rho(h(p), h(q)))^2$$

In (3.4) the operator ρ is the Hellinger distance as used in the data term (3.3). The histogram distance is nonetheless computed between superpixels p and q, which belong to the same neighbourhood. The superpixels centroids are noted as q_c and p_c, and u and v are the horizontal and vertical changes between centroids. This term is usual in the MRF formulation and has a smoothing effect in superpixels that belong to the same object. It has to be observed that when two close superpixels are different, thus, more probable to belong to different objects within the image, the term λ allows them to have matches that do not hold the smoothness prior with the same strength. It has to be noted that the proposed energy function is highly non-convex.

3.3 Energy Minimization

A fair amount of work has been dedicated to discrete optimization techniques in computer vision, leading to well-defined and widely tested approaches to solve pairwise MRF [3] [4]. However, some of the approaches restrict the construction of the spatial term, and/or enforce limitations in the number of labels [3]. Because of the high amount of possible labels for each superpixel in the proposed approach, the use of the Fusion Moves [7] technique seems to be well suited. This algorithm employs the Quadratic Pseudo-Boolean Optimization (QPBO), to combine incremental sets of proposal labellings, resulting in a semi-globally-optimal solution [4]. Thus, the minimization starts by proposing a set of possible solutions, and iteratively merges them with the QPBO technique.

The candidate solutions depend on the problem to be solved. For example, in stereo superpixel matching, some assumptions related to the cameras layout can be made to generate solutions. In a more generic sense, other assumptions can be made towards candidate generation. The Quadratic Pseudo-Boolean Optimization (QPBO) [3] [4] is used to minimize the proposed energy function, by merging a set of candidate matches for every superpixel in the first frame. For instance, for a given superpixel in the initial frame, the corresponding matching would be the most similar one in terms of color, shape, or the spatial distance. More candidate solutions can be added by defining a neighbourhood in the second frame and select random pairs from every neighbourhood of every superpixel in the first frame. This is suitable for problems where the images are extracted from the same video sequence. This algorithmic process is presented more precisely in the Algorithm 1, where it can be seen that for every iteration, a single binary QBPO is optimized, and successively merged onto a final solution ($winner$). These solutions are extracted by selecting radial neighbourhood superpixels ($radius\_r$) in the frame A, to match with superpixels in a radial neighbourhood ($radius\_p$) in the frame B.

In this case, the proposed approach is, to enforce a proximity prior and to extract possible matches from a circular neighbourhood in the subsequent frame centred on the current superpixel centroid. The radius of this neighbourhood is not necessarily of the same value as the radius r, and it accounts for the maximum possible matching distance between superpixels. Thus, this maximum search distance becomes a parameter to control the maximum amount of displacement of the superpixels. There is a direct relation between the value of this parameter and the processing time. To speed-up the minimization procedure, the QBPO properties can be exploited. For instance, the fusion of the proposed solutions is always guaranteed to be of lowest or equal energy than the two proposals. Thus, it does not matter the order in which the solutions are merged. One could, then, split the fusion procedure in several cores and build a hierarchical chain as proposals are subsequently fused. The idea of this implementation is compatible with the modern tendency to move the algorithms to the GPU level.

Algorithm 1 Superpixel flow minimization algorithm

Require: list: $superpixelsA, superpixelsB$, float: $radius\_r, radius\_p$
graph: $graph$ vector: $option, winner$
{ Initialize winner labels }
$winner[p] = q|argmin(hellingerDistance(p, q))$
{ Build unary terms }
for all $niterations$ **do**
$graph.restart()$
for all $p \in superpixelsA$ **do**
for all $q \in superpixelsB$ **do**
$distance = hellingerDistance(p, q)$
$graph.addUnaryTerm(p, q, distance)$
end for
end for
for all $p \in superpixelsA$ **do**
$p_c = centroid(p)$
for all $q \in superpixelsA|distance(p_c, q_c) < radius\_r$ **do**
$option[q] = selectRandomSuppixInB(q_c, radius\_p)$
$d00 = robustDistance(winner[p], winner[q])$
$d01 = robustDistance(winner[p], option[q])$
$d10 = robustDistance(option[p], winner[q])$
$d11 = robustDistance(option[p], option[q])$
$graph.AddPairWiseTerm(p, q, d00, d0, d10, d11)$
$graph.solveQPBO()$
end for
end for
{ Update winner labels }
for all $p \in superpixelsA$ **do**
$winner[p] = graph.getLabel(p)$
end for
end for
return $winner$

3.4 Matching Results

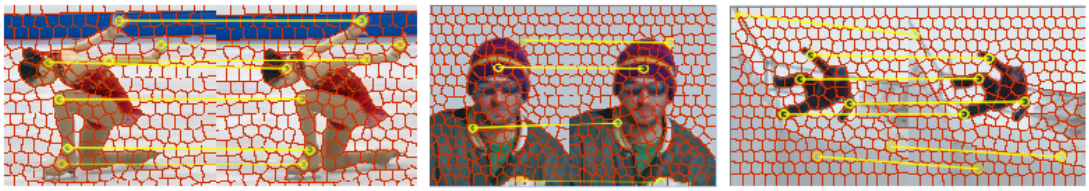


Figure 3.1: The yellow lines show selected superpixel matching between pairs of consecutive frames in a video with the proposed method. The video frames go from right to left. The images are a close-up of the actual frame, to appreciate the details.

The Fig. 3.1 shows some examples of superpixel matching between subsequent frames with the presented method. It can be seen that the matching performs well even in difficult cases, like the hands in the first column (*yunakim long2* sequence). It has to be noted as well that even in superpixels where there is a lack of texture, there is correct matching. This seems to be the effect of enforcing the regularization between superpixels that are close, but are also similar to each other. Moreover, unlike most of the optical flow methods, superpixel flow extends naturally for more distant frames. The Fig. 3.2 shows results for large separations between frames, without tweaking or adjusting any parameters. For this case, however, the matches in the texture-less part of the scene are mostly invalids. Though this is expected because of the aperture problem and heavy occlusions, thus, the matching in this region lacks of meaning.

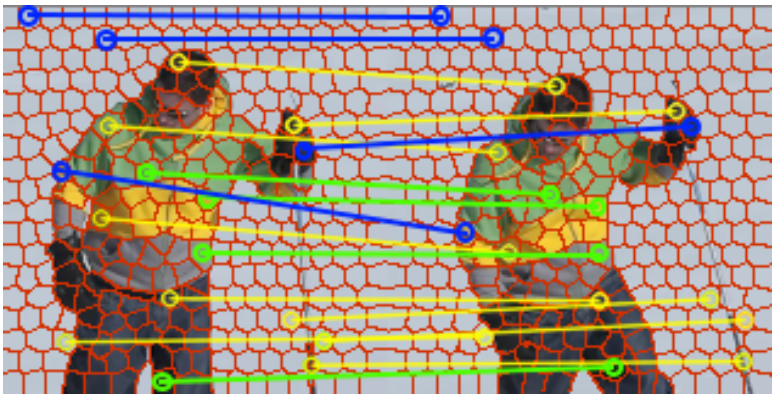


Figure 3.2: The coloured lines show selected superpixel matchings between a pair of distant frames in the Snow Shoes sequence.

Chapter 4

Object Flow Pipeline

Following the definition of the problem given in the Section 1.1, a system for computing the object flow have been devised. This system is composed of three main steps, which will be explained in short.

4.1 Algorithm description

Fig. 4.1 shows a simplified block diagram of the proposed system. It is important to recognize the two main components of our work-flow: object tracking and segmentation, and pixel-wise flow computation on the pixels of interest. The scheme is completed by feeding back the flow information to improve the tracker as depicted in the figure. For instance, one can make the most of dense displacement vectors to refine the motion of the target, and the segmentation information can be used to improve the sampling process of the learning stage in several trackers by detection methods [22]. Thus, the tracker and motion flow algorithm can work for mutual enhancement.

The object tracking in the object flow pipeline can be selected according to a specific need for a given application. Here, recent methods based on tracking-by-detection are preferred, given that they are in the top of modern benchmarks for object tracking [16] in terms of accuracy and stability to initialization changes. Even more, as previously mentioned tracking-by-detection techniques benefit from the background-foreground separation of the next stage in the object flow pipeline. For instance, the sampling process in the *Struck* tracker can be improved by selecting positive samples only inside the region of the target object. In the second place, for the object segmentation in video, the labelled background regions through the concept of superpixel flow, as explained in Section 3, is shown. The last step, the flow computation, uses a

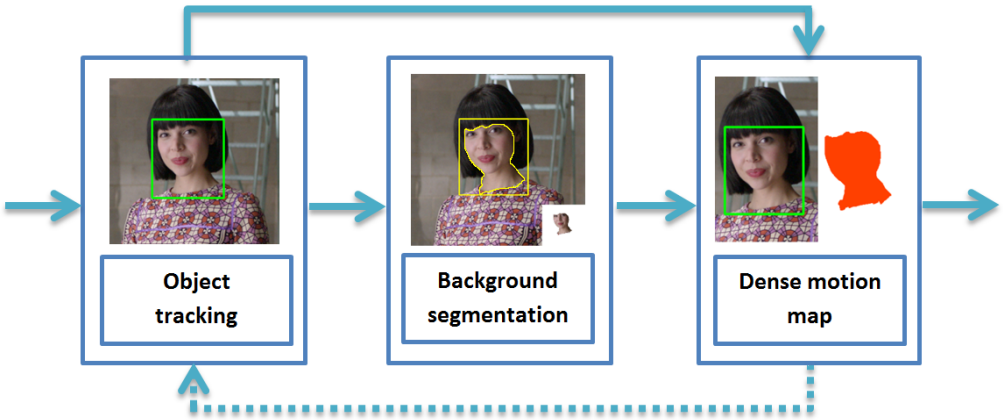


Figure 4.1: Block diagram of the proposed pipeline.

modified *Simple Flow* algorithm, which uses the provided segmentation masks for an accurate estimation.

4.2 Background regions tracking and segmentation

The algorithm proposed in [18], offers a good deal in terms of background-foreground separation from user interaction. A technique like this, however, performs very well in still images, but it may not be well adapted for sequential videos. Extensions to this method, like the GrabCut algorithm [14], work by implementing an iterative graph-cut based minimization to separate regions according to appearance information from a loosely drawn rectangle around the object, and small user-interaction-based hints. Given the tracker state for every frame, the minimization procedure of the methods in [18] and [14] could be extended to video. However, a lot of details in the segmentation contour may be lose if no fine hints are given. These hints usually depend on on-the-fly supervised methods. However, this need could be minimized in videos, given the extra information that offers the dynamics of the sequence. Some authors had approached the graph-cut based segmentation techniques in sequential videos to propagate a consistent segmentation [15]. However, some more work on reducing user interaction given the extra flow-like information that video sequences offer is still needed. Determining the spatial support of the object in a given frame benefits from the output of the tracker. Appearance cues alone, learned inside and outside the tracking window can result in a misleading modelling of the foreground and the background. In contrast, we propose to perform foreground-background segmentation by tracking background pixels surrounding the target, thanks to the tracker out-

put. Thus, the pixels that are initially outside the tracker window, are followed through the sequence and as long as they enter the tracked region, they can be safely labelled as background. This idea can be observed in the Fig. 4.2, where the object window given by the tracker (green) loosely separates the foreground from the background. Points outside the tracker (blue) are labelled as background in previous frames, and as they enter the tracker window (red points), they can be used to improve the modelling of the foreground and the background. The red window is used to save computational power by avoiding to track points that are too far from the interest object.

Figure 4.2: Example image of points entering a tracking region (green) due to object motion in a video sequence.

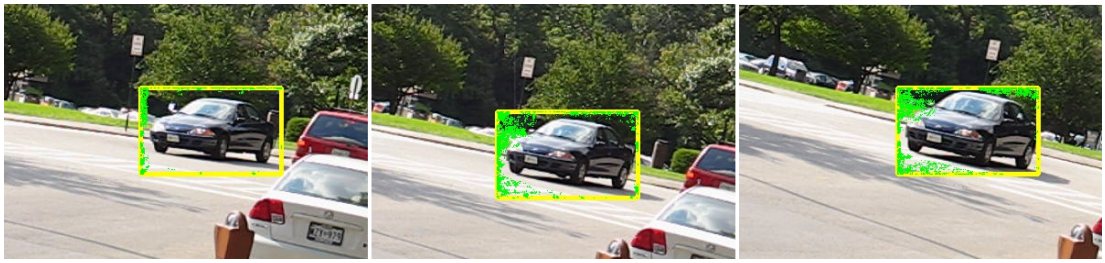


Figure 4.3: Point tracking using an optical flow method.

This idea can be applied in several levels in a video sequence. The first method to develop such an algorithm is to do it in the pixel level. The Fig. 4.3 shows some results by applying a dense optical flow technique (Fast Block Matching) in an external window. However, as it can appreciated, the results are rather sparse and after some time, the regularization of the optical flow method leads to some of the points being wrongly tracked inside the object boundaries. Both problems, sparsity and tracking errors due to drift and regularization, can be overcome

by implementing the same idea, but in the superpixel level. To save computational power, the tracked superpixels are limited to the ones that fall inside a control region (red box in the Fig. 4.2).

Normally, after several frames, the labelled superpixels will almost completely cover the unwanted areas in a dynamic scene. We call this process background segments tracking and the process is summarized in the Algorithm 2. The Fig. 4.4 shows this idea in a real scenario. From left to right, initially the superpixels with elements outside the bounding box are labelled as background (green), then, as the sequence changes, the labelled superpixels flow inside the window, giving hints for the model initialization in the background-foreground separation algorithm.

Algorithm 2 Background regions tracking between a frames A and B

Require: list: $superpixelsA, superpixelsB$, rect: $trackerA, trackerB$, vector: $prev\_labels$
  vector: new_labels
  $computeSuperPixelFlow()$
  for all $superpixel \in superpixelsA$ **do**
    $matches[superpixel] = getMatchesFromFlow(superpixel)$
    { Check is previously labelled superpixels fall inside $trackerB$ }
    if $superpixel \in prev\_labels$ **then**
      $matchAB = superpixelsB[matches[superpixel]]$
      if $matchAB \in trackerB$ **then**
        $new\_labels.push\_back(matchAB)$
      end if
    end if
    { Check is new labelled superpixels fall inside $trackerB$ }
    if $superpixel \notin trackerA$ **then**
      $matchAB = superpixelsB[matches[superpixel]]$
      if $matchAB \in trackerB$ **then**
        $new\_labels.push\_back(matchAB)$
      end if
    end if
  end for
  return new_labels

At this point, some generic segmentation technique can be connected to the pipeline to refine the segmentation (e.g. region growing). However, graph based segmentation methods are preferred ([18] [15]) because the usual user interaction can be replaced by the tracked background regions, and the algorithms are faster and more robust.

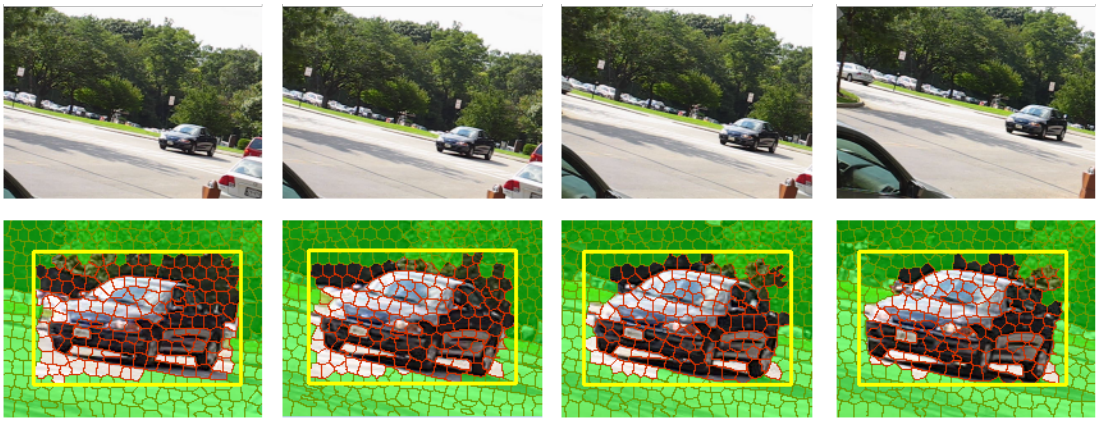


Figure 4.4: Background segments automatic labelling and propagation, the flow goes from left to right.

4.2.1 Segmentation results

Fig. 4.5 shows the results for an image sequence where the interest object is the head of a person. The head tracker and the superpixel flow provide information for better background-foreground separation. The background-foreground models are updated as the frames go on, giving more robustness for sequential propagation of the segmentation. The method is tested in the Walking Couple sequence, by allowing only a small amount of iterations in the graph based segmentation. Observe how the contour in the man's head is correctly delineated when another person's head occludes part of it. In this case, the superpixels that belong to the womans face were correctly propagated and thus, labelled as background. Its also impressive that the segmentation process recovers after a heavy occlusion. In this case, the fact that the tracker is also robust to the occlusion is a key factor that help the system maintain a correct segmentation through such a difficult case.

In order to understand the effect of including superpixel propagation in a video sequence for object segmentation, some results are shown in the Fig. 4.6. For these experiments only one iteration is allowed in the graph-cut based methods. The top row frames (Fig. 4.6) were initialized only with the tracker, and the bottom row was initialized with the superpixel tracking technique. Observe that in general, the contour delineated is usually better in terms of precision and stability for the later one. Complete segmentation results can be observed in the Appendix B.

Figure 4.5: Segmentation through the sequence Walking Couple (Yellow contour) initialized in the mans head. The yellow box correspond to the tracker output. The labelled background superpixel are not shown for clarity.

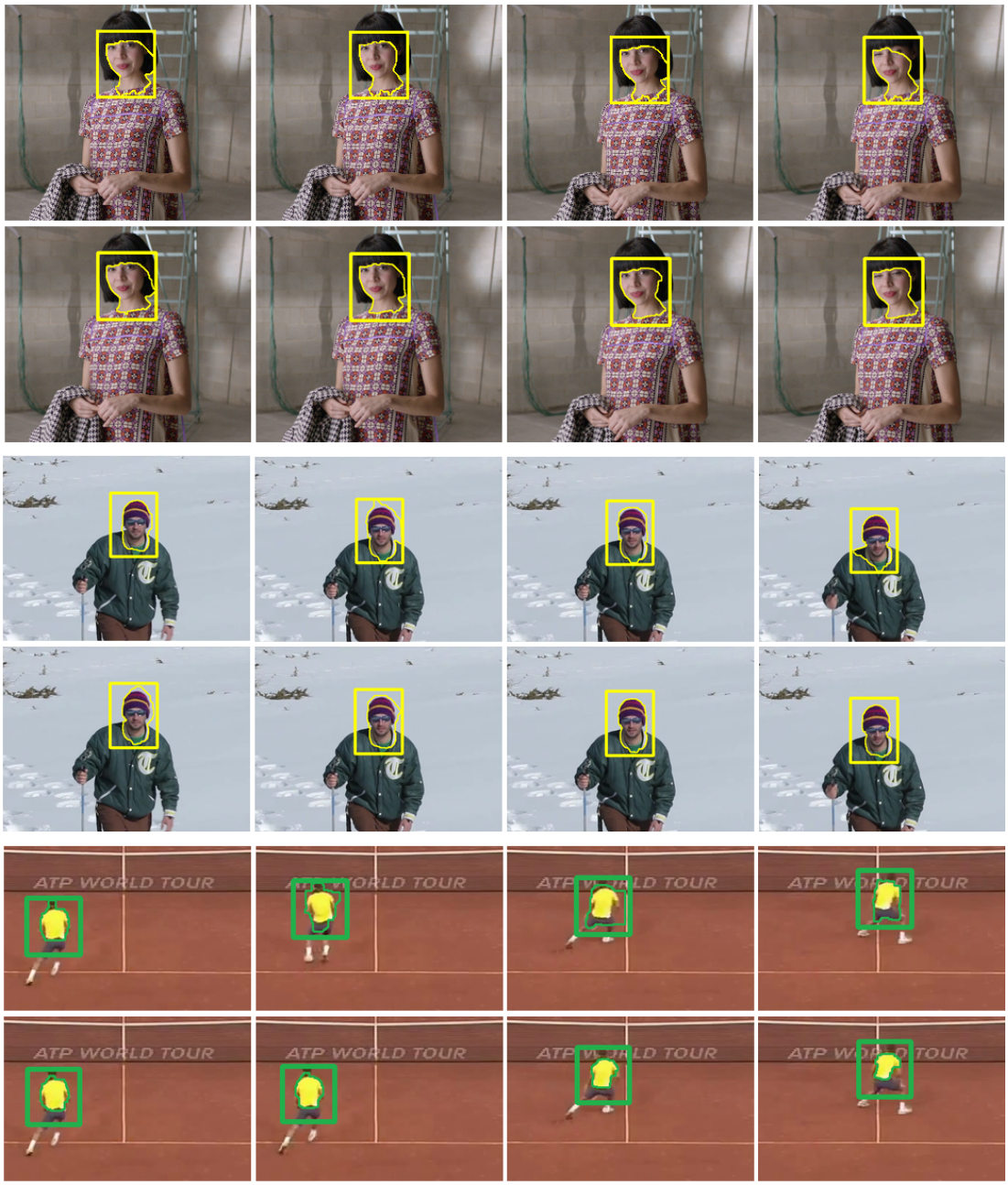


Figure 4.6: Face segmentation in the Amelie Retro and the Snow shoes sequences in several different frames, and T-shirt extraction from Tennis sequence. For each group, the Top Row: One-iteration window-based graph-cuts; and the Bottom Row: One-iteration graph-cuts initialized with superpixel tracking.

4.3 Flow estimation

The object flow consist on computing the motion field for an object of interest through an image sequence. The most usual approach to solve a problem like this is to implement some of the available optical flow techniques through the complete sequence and perform the flow integration. However, this process results in high levels of motion drift [18] [19] and usually the motion of the interest object is affected by a global regularization. In some extreme cases, the interest object motion may be totally blurred and other techniques have to be incorporated. Moreover, the diversity of natural video sequences makes difficult the choice of one optical flow technique over another, even when specialized databases are at hand [17], because currently no single method can achieve a strong performance in every of the available datasets. Most of these methods consist in the minimization of an energy function with two terms (As was previously mentioned in the Sec. 1). The data term is mostly shared between different approaches, but the prior or spatial term is different, and basically states under what conditions the optical flow smoothness should be maintained or not. In a global approach, however, this is a difficult concept to define. Most of these smoothness terms rely in appearance differences or gradients. All these meaning that, unavoidably, some methods may be more reliable for some cases but weaker for others. It can be argued that this behaviour may be caused because most of the techniques do not count with a way to identify firmly where exactly this smoothness prior can be applied.

It is difficult, nevertheless, to blame the authors for the choice of some regularization terms, since the more robust the regularization terms is, and the higher level knowledge is applied, the more difficult is to minimize the energy function, and thus, the more difficult is to obtain a reliable global solution. The modern advances in non-convex optimization methods have allowed some authors to go further in the writing of these energy functions, but at the end they still rely in the pixel level to determine whether a strong regularization should be applied to a zone or not.

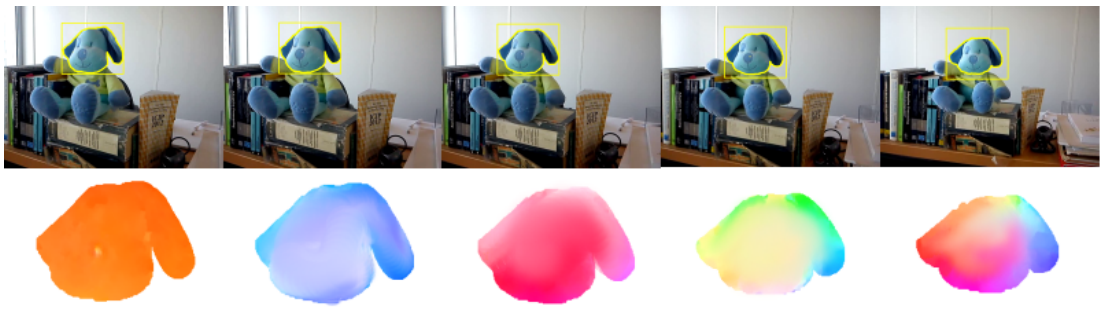


Figure 4.7: Object flow with the color code of [17] (bottom) for frames in the Puppy sequence (up).

The main idea behind the object flow is that given the availability of several robust tracking techniques, and the proposed segmentation method for video, the optical flow computation can be refined by computing it successively between pairs of tracked windows. The basic proposal to perform this refinement consist on considering the segmentation limits as reliable smoothness boundaries. This is, of course, under the assumption that the motion is indeed smooth within the object region. This is assumption is not far from reality in most scenes with an interest object, and it is indeed a better way to determine the limits of the regularization than using the difference between raw pixel values. Naturally, as the object tracker is included, is expected that the object flow should be more robust to rapid motions than the optical flow. Thus, the full motion is split in two, the long range motion, given by the tracker window, and the precision part, given by the targeted optical flow. The Fig. 4.7 shows the object flow for a frame in the Puppy sequence. Observe the motion vectors are computed only inside the object of interest, preserving a strong smoothing prior, but also allowing internal variations in the flow.

As a first approximation to the object flow, the Simple Flow technique [21] is taken as core base. This is because of its scalability to higher resolutions and because its specialization to the concept of object flow is only natural. The reason behind this is that in the Simple Flow pipeline the smoothness localization can be easily specified through computation masks. More specifically, the initial computation mask is derived from the segmentation performed as prior step. An iterative multi-scale approach based on this initialization follows. The original method uses bilateral filtering to refine each flow result. However, as a reliable segmentation mask is provided, this bilateral filtering is replaced by a Gaussian filter applied within the limits of this mask, highly improving the speed of the flow estimation.

In more detail, for every pixel within the segmentation boundaries a vector e of dimension n^2 is computed by extracting a color difference nxn window centred in that pixel. The energy E (as in 2.2) is computed within the segmentation boundaries by applying a bilateral filter, using the color data of the initial frame to define the filter weights. The flow for the given pixel is the vector that minimizes E. The final bilateral filter applied over the computed flow field for every scale is replaced by a Gaussian filter limited by the segmentation masks. The process is done twice, from I_t to I_{t+1} and vice-versa. An occlusion mask is generated by eliminating the flows that do not correspond with its backward counterpart ($||(u_f, v_f) - (-u_b, -v_b)||$ is too high). The process is repeated in several scales to account for possible large motions inside the studied object (Fig. 4.8).

However, direct modifications in other optical flow methods can be further studied. For instance, in graph-cut based minimization approaches, the regularity constraints can be precisely targeted by disconnecting foreground pixels from background ones.

The object flow concept is valid, however, for those cases where there is not a separable object, and the segmentation is not valid or possible to extract. In order to show this and the

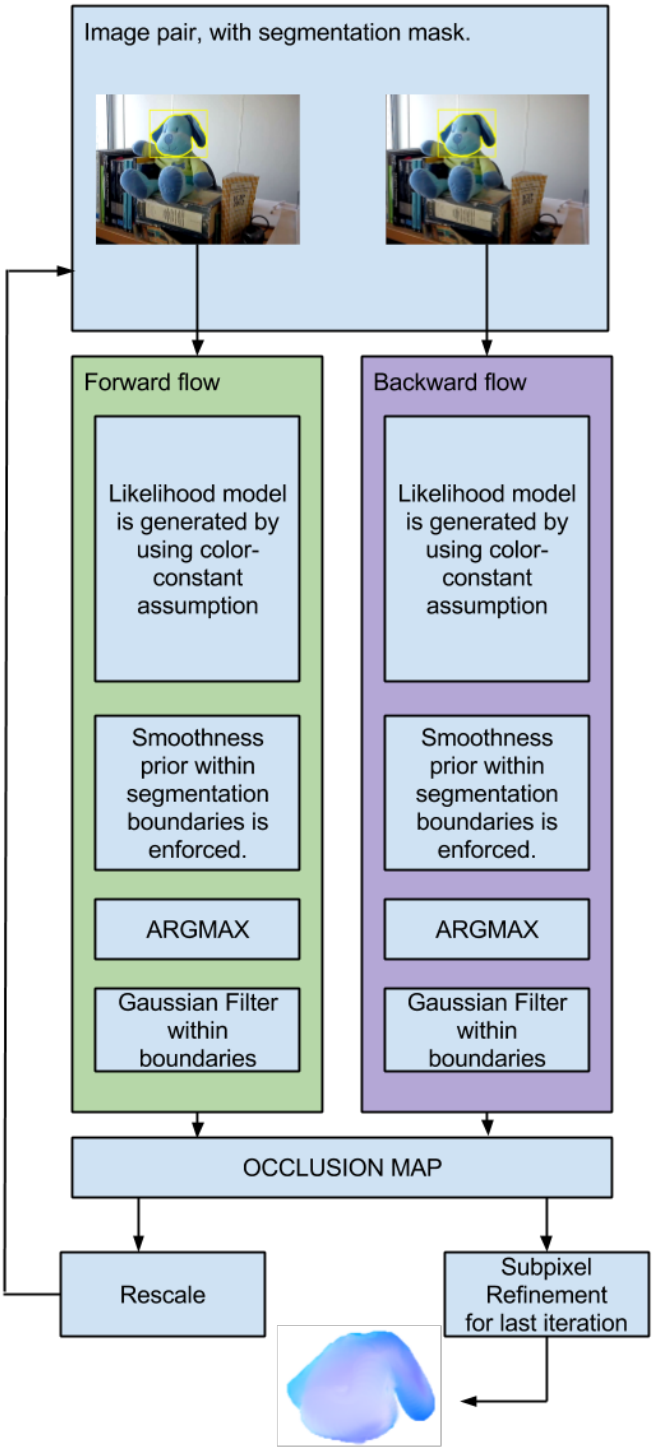


Figure 4.8: Simple object flow algorithm diagram.

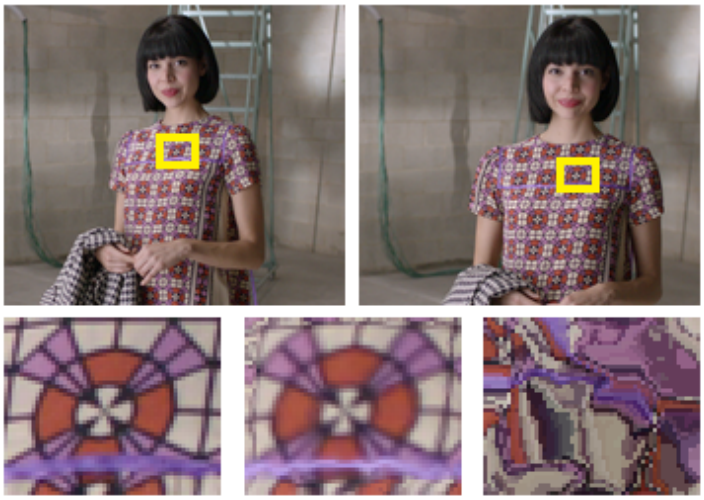


Figure 4.9: Top row: First and last frames used from the Amelia sequence. Bottom row: From left to right: Groundtruth patch; patch generated with the basic object flow combining the *Struck* Tracker and the *TVL1* optical flow; and patch generated with the globally computed *TVL1* optical flow.

fact that only combining the tracker information with a locally computed optical flow is already an improvement to a global optical flow, the Fig. 4.9 shows an experiment where a patch from a dress is extrapolated by using the most basic object flow definition (no segmentation) and by a globally computed optical flow. It can be seen that the results are better for the first method.

4.4 Feedback for tracking methods

As previously shown in the Fig. 4.1 the acquired knowledge by the segmentation mask, and flow estimation can be used to feedback the tracker method, and generate a more precise position of the object in the next frame. There are several ways to perform this feedback depending on the specific tracking method used. However, to mantain certain generality in the pipeline of the object flow, the used cues are the ones given by the segmentation mask and the tracked background regions that could behave as occlusions. In the most of the tracking by detection methods, a sampling step (which can be explicit like in the *MIL* tracker, or implicit, like in the *Struck* tracker) is performed to extract features that are valid as possible to represent the target object.

When occlusions happen, the most of the methods have a hard time to select a correct bag of features to represent the said object, as it gets difficult for the learning method to explain the object after and before occlusions with a given set of features. The Fig. 4.10 shows how

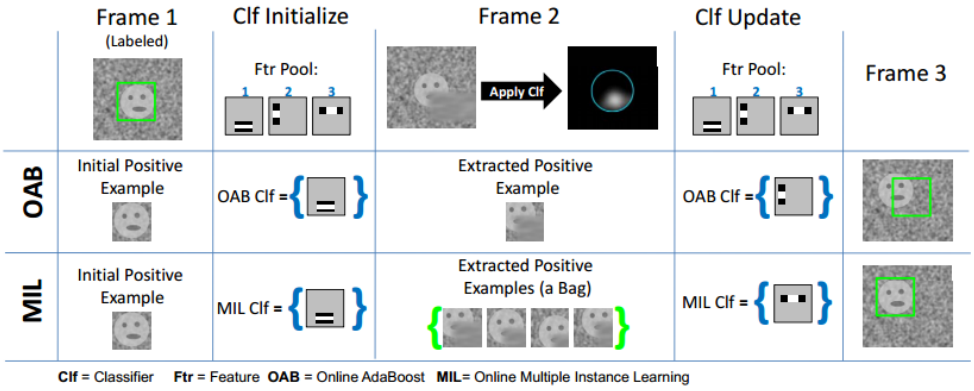


Figure 4.10: An overview on how tracking-by-detection methods try to deal with occlusions. Extracted from [25].

Online Adaboost and Online Multiple Instance Learning try to deal with this problem when a small occlusion happens.

Frame 1: Consider a simple case where the classier is allowed to only pick one feature from the pool. One positive patch and several negative patches are extracted, and the classiers are initialized. Both OAB and MIL result in identical classiers - both choose feature no.1 because it responds well with the mouth of the face. Frame 2: The mouth is now occluded, and the classier trained in the previous step does not perform well. Thus, the most probable image patch is no longer centered on the object. OAB uses just this patch to update; MIL uses this patch along with its neighbors. Frame 3: When updating, the classiers try to pick the feature that best discriminates the current example as well the ones previously seen. OAB has trouble with this because the current and previous positive examples are too different. It chooses a bad feature. MIL behaves better in this case because a correct sample was included in the positive bag [25]. This may not be the situation for heavier oclussions, where the chances of taking the object as sample are low or null. In these cases, having a precomputed occlusion mask of the last frame, can help to select better which features to take into account, or even to avoid completely the sampling process in a given frame.

This idea is summarized with the experiment performed with the *Struck* tracker. These results can be observed in the Fig. 4.11. The sampling process is altered to receive feedback from the background segmentation. As it can be seen, for the top row (original implementation using only Haar features) the tracker starts learning the tree (oclussion object) since the third shown frame. In the other hand, as the sampling process is modified by rejecting the zones labelled as background (including the tree), the tracker quickly recovers after the oclussion in the bottom row. Both experiments were performed with a very narrow search window, and

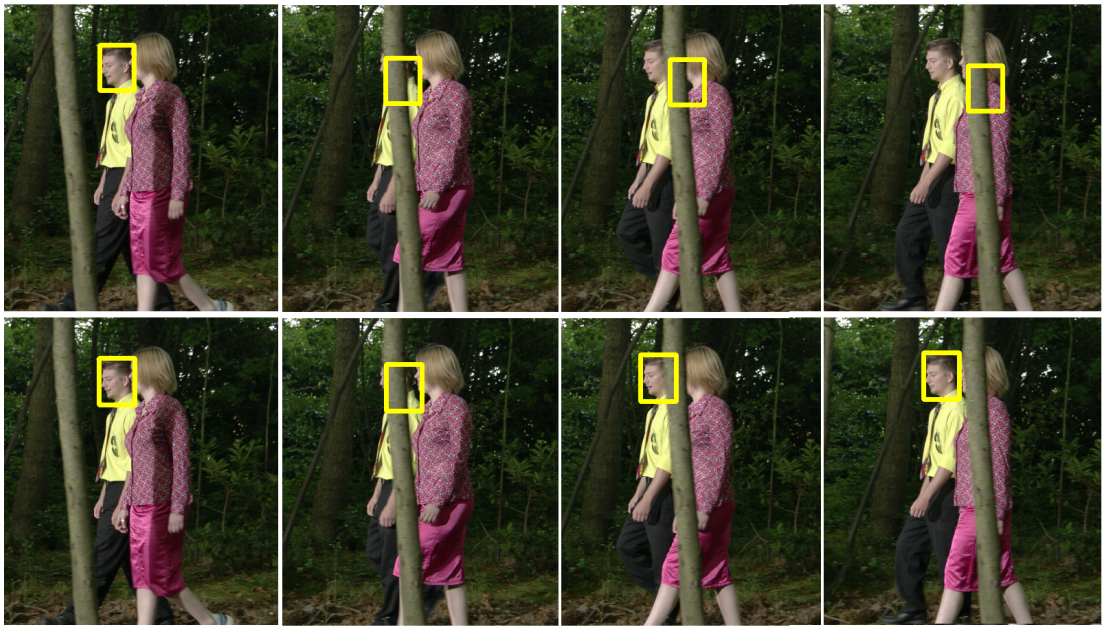


Figure 4.11: Struck tracker in the Walking Couple sequence. Top: Original Implementation. Bottom: Struck tracker with sampling refinement by background regions tracking.

by using only Haar features. This is, the tracker did not learn the features extracted from the occluding object, and the original object is recaptured when it reappears in the scene. Altough these experimentes are promising, a more deep study is needed, since the *Struck* tracker can be more robust if an optimal search window is used, and more features are included in the machine learning process.

Chapter 5

Implementation Details and Evaluation

The evaluation methodology is presented now, including a description on the ground truth data acquisition, and the performed experiments. Implementation details are also discussed at the end of this section.

5.1 Experiments

To evaluate the performance of the object flow in comparison with optical flow techniques, we performed a number of experiments on several video sequences. We annotated an initial bounding box for the videos, and a segmentation contour (Fig. 5.2) of the interest object for every frame. The experiment measures the ability of the method to reconstruct an image from the initial frame and the integrated flow. For every pair of frames the video sequence, the PSNR between the annotated current state of the object and the extrapolated images is computed. The Fig. 5.1 is a sample of the performed experiment, each column is an image generated from the given flow. Two types integration are evaluated, *From-the-reference*, or forward integration, and *To-the-reference*, or backward integration, as discussed in [20]. So, for each row in the Fig. 5.1, two columns correspond to the object flow, and two columns correspond to optical flow, with both types of integration.

The Fig. 5.3 shows PSNR graphics for 4 different sequences. For every pair of frames an image is reconstructed, and the PSNR with the ground-truth object is computed. The results are shown with both, Euler integration (Labelled as *forward* in the figs.) of the used flow, and using the integration method described in [20], labelled as *backward* in the figures. The results

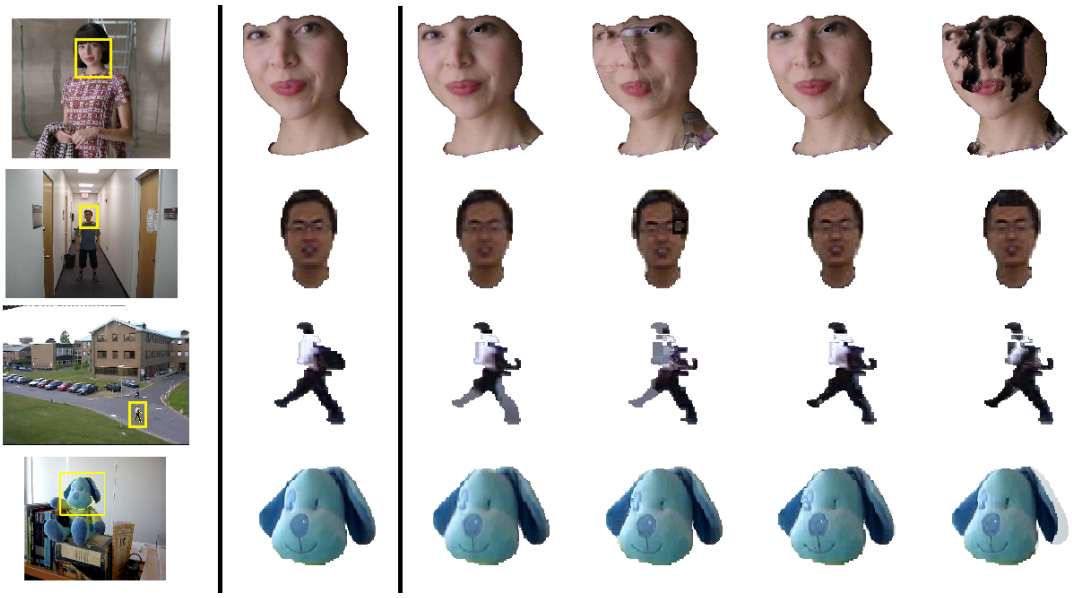


Figure 5.1: Reconstruction results from integrated flow in 4 sequences. In descending order: Amelia Retro, Boy, Walking, Puppy. From Left to Right: Annotated object, Backward object flow, Backward optical flow, Forward object flow, Backward optical flow.

show that the object flow methods are generally more precise than its optical flow counterparts. Moreover, the object flow method with backward integration usually performs much better than any other combination of techniques. For this experiment, the object flow is compared with the simple-flow optical-flow method. This experiment directly shows that the object flow concept is indeed capable of increasing accuracy of a given optical flow method.

Now, in order to study how the object flow concept compares to several state of the art optical flow methods, another extrapolation experiment is performed. The Fig. 5.4 presents a visual comparison between the object flow and several optical flow techniques in the Amelia sequence for object reconstruction, and the involved frames (the first and last used frames in the sequence). The Fig. 5.5 shows the PSNR results for every extrapolated frame in the full sequence, the object flow performs better than all the studied optical flow techniques.

Observe that the object details are lost in comparison with the ground-truth object image (Fig. 5.4). For example, the closed eyes detail is missing in the most of the optical flow methods. Furthermore, several of the methods lost any significance, and the output barely holds any resemblance with the original image. This is possibly the result of a couple of details that are naturally better attacked with the object flow: long motion (the tracker always centers the object in any given frame), and smoothness prior (the segmentation mask is a good delimitation to establish this prior).

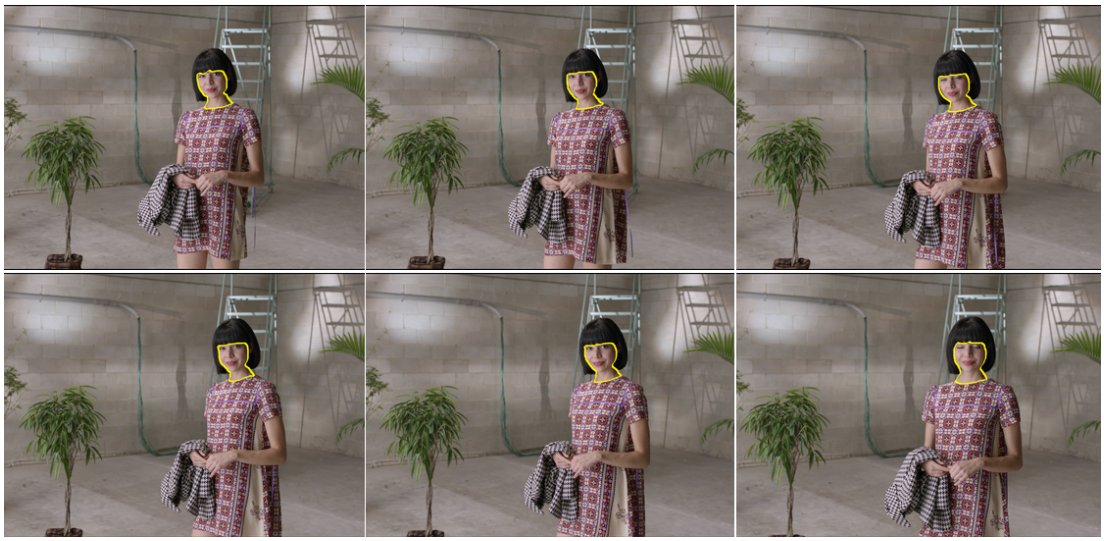


Figure 5.2: Handmade contour (yellow) in the Amelia dataset used as ground truth.

5.2 Implementation details

A framework to combine state-of-the-art optical flow and tracking methods was implemented in C++, and using the OpenCV 3.0 and Eigen 3 libraries. The framework consists on 3 main virtual classes: Tracker, OpticalFlow and ObjectFlow. The function of these base classes is to define the interfaces for several implementations of each algorithm. For instance, the implemented or adopted tracking methods are:

- Color Histograms Mean-Shift [44],
- Color Spatiograms Mean-Shift [46],
- Color Spatiograms Based Particle Filter [46],
- Color Histogram Cam-Shift [45],
- Multiple Instance Learning Tracker [25],
- Boosting Tracker [47],
- Struck Tracker [23],
- TLD Tracker [48].

And the used optical flow methods are:

- Lucas-Kanade [31],
- Horn [30],
- TV-L1 [32],
- Fast Block Matching,
- Simple Flow [21],
- Farneback [28],
- Affine Transformation based on Keypoint matching,
- Thin Plate Spline Transformation based on Keypoint matching [49],
- Brox method [29].

The Appendix A shows a simplified class diagram for the implemented framework. It can be appreciated that all major interfaces inherit from the *cv::Algorithm* class to provide implementation flexibilities through instantation by smart pointers (*cv::Ptr*) and to mantain organization in the interfaces. Also, for some of the implementations, only interface wrappers had to be programmed due to the availability of some of the methods in the OpenCV library (e.g. CamShift, MIL and Boosting trackers). Another detail in the implentation of the framework is the use of *GPU* implementations of several of the methods to improve time efficiency of the object flow. This was done with the Nvidia paralell computing toolkit (CUDA), obtaining a performance for object based video edition of 3.5 frames per second, computed in a video of 1920x1080 pixels per frame, on a computer with a low level Nvidia card (NVS 3100M), and a 2.5 GHz processor.

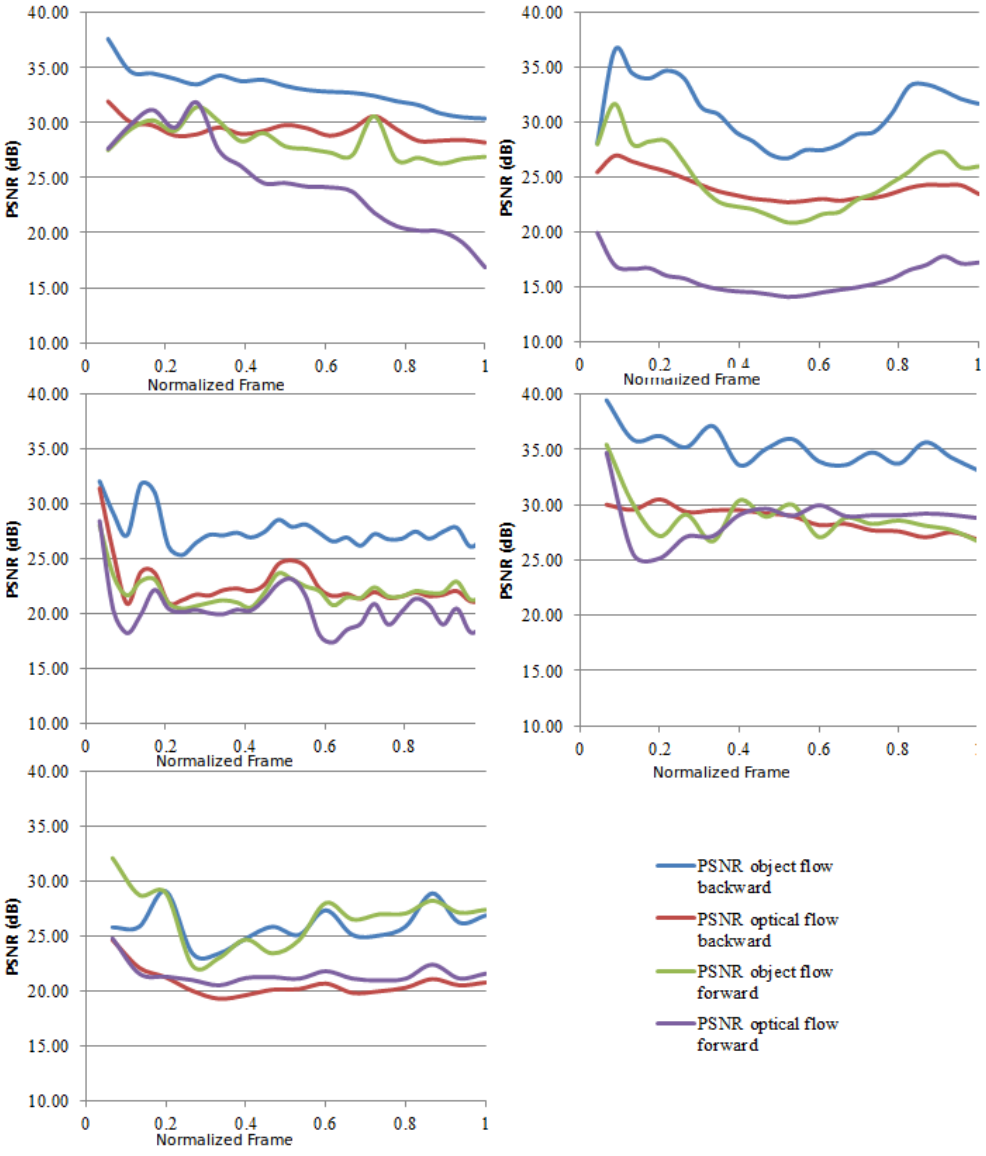


Figure 5.3: PSNR graphs for reconstructed images using Object flow and the Simple Optical Flow for 4 sequences. From left to right and up to bottom: Puppy Seq.; Amelia Retro Seq.; Boy Seq.; Walking Seq.

Figure 5.4: Top: The first frame and the accumulated flows are used to reconstruct objects in the frame number 30. The used methods from left to right: Groundtruth object, Object flow, TVL1, Block Matching, Brox, Farneback and Simple Flow. Bottom: First and frame#30. The reconstructions are performed using backward accumulation of the flows.

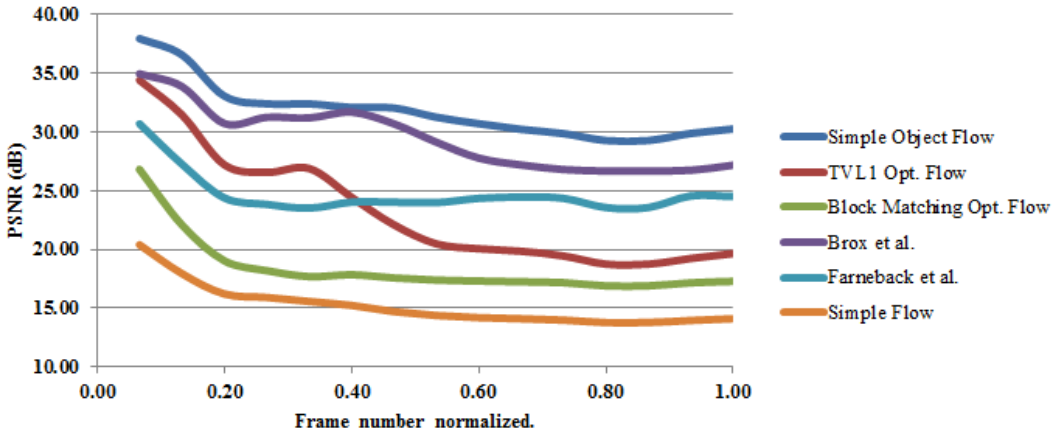


Figure 5.5: PSNR graphs for reconstructed images using Object flow and the different Optical Flow techniques for the Amelia sequence.

Chapter 6

Applications

A couple of applications for the object flow are presented. Firstly, in complex video editing and augmented reality, and secondly as part of the object-based structure-from-motion. Only visual results are provided in both cases.

6.1 Video editing

The video editing that is approached here is the insertion or modification of an element in a sequence. For instance, during post-production it is possible that an element of a sequence presents inadequate content for the expected audience, and usually removing these elements require a hard manual art-edition work. The object flow can be exploited in order to reduce drastically the amount of manual work that has to be done.

The proposed approach is to edit manually the initial frame. The elements of this editing can be propagated through a number of subsequent frames with the object flow. Moreover, for complex scenes, with heavy occlusions and sudden illumination changes, several instances of the object flow can be used, to take into account the difficult cases. Thus, the overall editing work is greatly reduced. Some examples in augmented reality / video editing can be observed in Figs. 6.1 and 6.2.

6.2 Structure from motion

The idea of the structure from motion problem (*SfM*) is to recover the shape of objects or scenes from a sequence of images obtained from a camera that follows certain motion. Usually, it is assumed that the scene contains rigid objects undergoing an Euclidean motion [43].

Figure 6.1: Selected frames in the Dmitry test sequence for video editing. Top: Inserted logo. Bottom: original frames

Figure 6.2: Selected frames in the Juan test sequence for video editing. The logo of the University of Burgundy is inserted in the sequence.

The most common way to solve the *SfM* is to find a set of correspondences between the acquired frames by using a feature matching method, and use the epipolar geometry constraint for two images ($x_1^T E x_2 = 0$). Usually, the *8 points algorithm* is used to obtain a relative rotation and translation of the camera by factorizing the essential matrix (E). However, given the simplicity of the approach, the algorithm is very sensitive to noise and several other approaches have been proposed: *7 points* or *5 points* algorithms are at hand. Nevertheless, one of the major weakness of such methods is still the feature matching (w.r.t precision and sparsity) [43].

When matching is replaced by optical flow, the scene structure and the camera motion are tied together by the equation:

$$\mathbf{u}(x) = A(x)\mathbf{v}/Z + B(x)\omega, \tag{6.1}$$

where, $A = \begin{bmatrix} 1 & 0 & -x \\ 0 & 1 & -y \end{bmatrix}$, and, $B = \begin{bmatrix} -xy & (1+x^2) & -y \\ -(1+y^2) & xy & x \end{bmatrix}$.

In Eq. 6.1, ω is the camera angular velocity, and $\mathbf{v}$ is the camera linear velocity. The 3D point position is $\mathbf{X} = [X, Y, Z]$, and its projection over the focal plane is $\mathbf{x} = [x, y]$.

The optical flow, however, has been traditionally inaccurate for long term point tracking, and due to the fact that an optimal structure in the least square sense can be obtained from camera velocities (which are more accurately extracted from the flow), the research was mainly focused on capturing camera ego-motion [43].

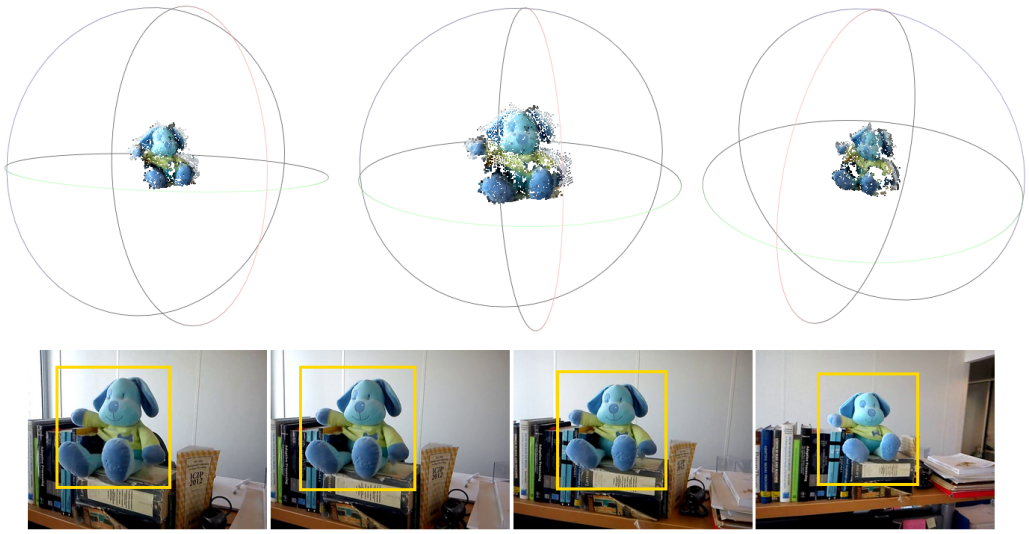


Figure 6.3: SFM based on the object flow matches in the Puppy dataset. Top: views of the computed structure. Bottom: Used frames.

As it has been demonstrated that the object flow increases accuracy of the point trajectories

estimation, it can be connected to a *SfM* pipeline as point matching method, not only as camera motion direct estimator. Some results can be appreciated in Figs. 6.3 and 6.4. These results are generated by using the Changchang Wu's public tools based on his work on Linear-Time Incremental Structure From Motion [42] and sets of matches generated with the object flow between pairs of frames.

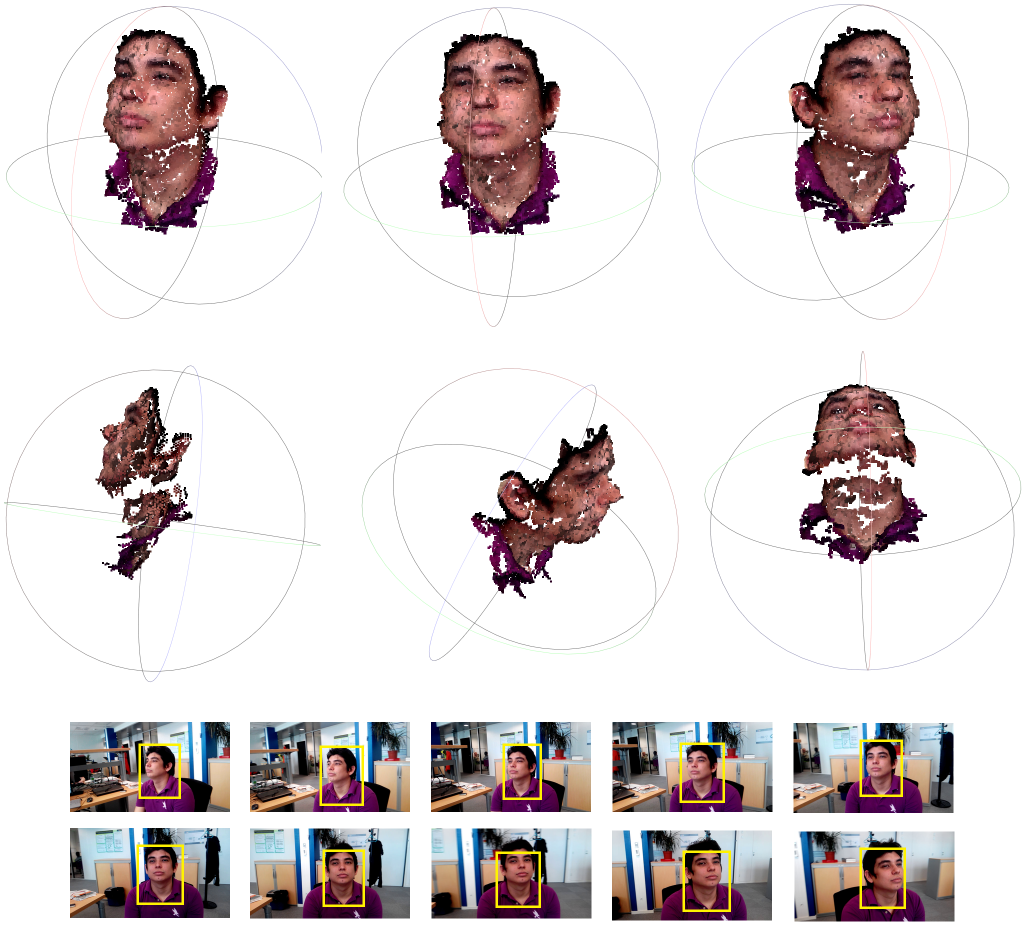


Figure 6.4: SFM based on the object flow matches in the JuanFace dataset. Top: views of the computed structure. Bottom: Used frames.

Chapter 7

Conclusions

7.1 Discussion

A framework to combine tracking and optical flow methods to improve object based dense motion description is presented. The pipeline is composed of three main steps, object tracking, segmentation and flow estimation. For the segmentation step a new promising video object segmentation algorithm was proposed, and, to the best of our knowledge, the introduced superpixel flow is the first energy based algorithm for superpixel matching. This superpixel matching technique is used to track regions that belong to the background of the scene and allows a fine initialization of segmentation techniques, leading to a precise and stable segmentation method for objects in video sequences. For the last step, a flow estimation method based on a modification of the simple-flow method to use the obtained segmentation mask is presented.

The experiments showed that this object based flow estimation improves the dense motion estimation in comparison to optical flow techniques. This is done, of course, at the expense of involving a manually initialized object tracker in the process. An object detector can be added at the beginning of the pipeline, but this kind of adding would only worth for very specific tasks. Moreover, it is also shown than a simple mixing of tracking and locally computed optical flow is already better than global optical flow methods. This mixture seems to be in contradiction with the classic aperture problem. However, as the results are evaluated only inside an interest region (the object), this problem disappears. Thus, as the tracker provides a global motion displacement, and the dense flow is only computed within the given tracker windows, a more precise flow is obtained.

Future work can be further explored in the use of the object flow as feedback hint for tracking-by-detection methods. Also, several kind of applications of the object flow can be

more deeply approached. For instance, in the structure-from-motion pipeline, video based rendering, automatic video edition, and video in-painting among others. Moreover, it is worth to generalize the concept of object flow to create a precise (global) optical flow method. One idea to accomplish this is, for instance, to over-segment the two involved frames, and initialize simple trackers to solve for the global motion of every segment, and then apply a heavily regularized flow estimation for every pair of segmented objects in the sequence.

7.2 Final remarks

Figure 7.1: Sequence with different difficult cases for flow estimation methods: Autobalanace, illumination changes, specularities.

Known weaknesses. A few remarks have to be disclosed about the object flow. As it was presented, it is also tied to the same drawbacks as the normal optical flow methods. For instance, the Fig. 7.1 shows how the object flow based video edition is affected by common

problems related to videos: automatic white balance of the camera, illumination changes and specular reflections. The Fig. 7.2 shows a close-up to the generated artefacts in some of the frames. This means, for some kind of videos, for the object flow to behave accurately, some pre-processing would be needed. Nevertheless, as it was shown before, the object flow stands accurate when these problems are not very strong.

Figure 7.2: Close-up to the generated artifacts due to the illumination changes and specular reflections.

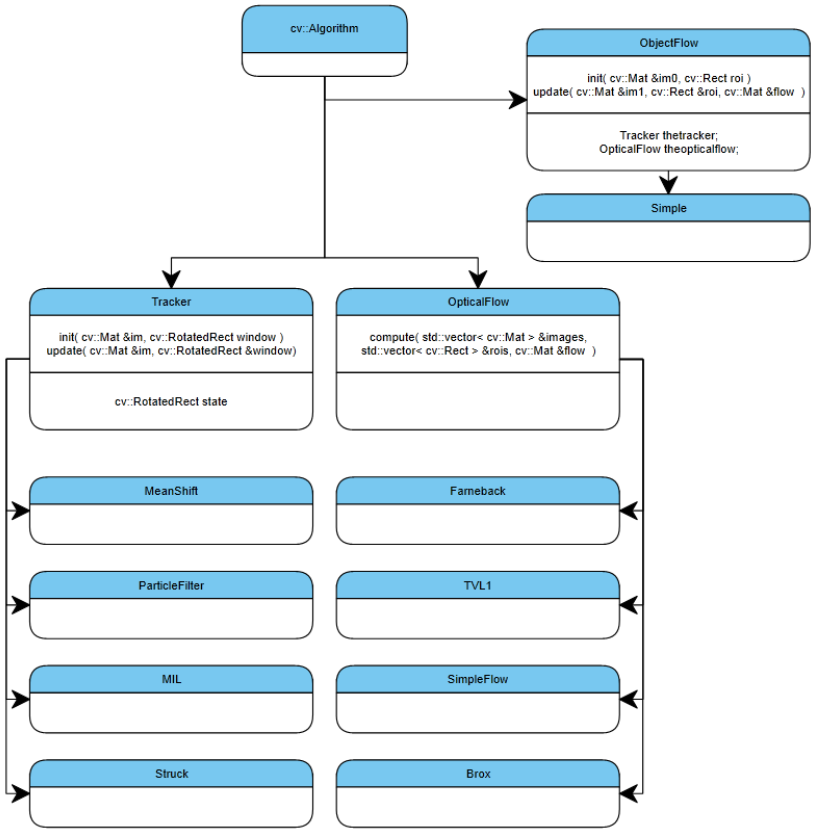
In addition, as it was shown in the Fig. 5.1, the assumption of strong spatial smoothness within the object boundaries seem to affect the results for articulated objects. A piecewise smoothing prior can be used in a superpixelated object to minimize these kind of artifacts. This, however, was not tested.

Other results. The background tracking device through the superpixel flow as method for object segmentation in video and to enhance object tracking is under a pending patent for the USA, EU and Singapore. The object flow pipeline, and the results obtained from the experiments presented in this work are summarized in a paper to be submitted in an international conference. A demo application for augmented reality is under development, and to the moment of the writing of the final draft of this document, 13.922 lines of code were written (11.564 in *.cpp* files, 1.192 in *.hpp* files, and 1.166 in *.h* files).

Appendix A

UML Diagram

A UML diagram of the implemented framework is presented. All the implemented algorithms inherit from the OpenCV *cv::Algorithm* interface.



Appendix B

Segmentation in Video

Some segmentation results in video are shown. In the Snow Shoes sequence, the segmentation contours (man on the right) are shown in yellow in 4 of the frames.

For the Amelia sequence, the face is tracked and segmented (yellow).

Bibliography

[1] J. Malik and X. Ren, Learning a classification model for segmentation, *Computer Vision, International Conference*, 2003.

[2] S. Boltz; F. Nielsen and S. Soatto, Earth mover distance on superpixels, *International Conference on Image Processing*, 2010.

[3] E. Boros; P. Hammer and G. Tavares, Preprocessing of unconstrained quadratic binary optimization, *RUTCOR*, 2010

[4] E. Boros and P. Hammer, Pseudo-boolean optimization, *Discrete applied Mathematics.*, 2002

[5] B. Horn and B. Schunck, Determining Optical Flow, *Artificial Intelligence*, 1981

[6] H. Ishikawa and P. Bouthemy, Multimodal estimation of discontinous optical flow using Markov random fields, *TPAMI.*, 1993

[7] V. Lempitsky, S. Roth and C. Rother, Fusion Flow: Discrete-Continuos optimization for optical flow estimation, *Computer Vision and Pattern Recognition*, 2008

[8] M. Reso and J. Jachalsky, Temporally Consistent Superpixels, *International Conference Computer Vision.*, 2011

[9] R. Achanta; A. Shaji; K. Smith; Aurelien Lucchi; P. Fua and S. Susstrunk SLIC, Superpixels compared to state of the art superpixel methods, *Discrete applied Mathematics.*, 2002

[10] F. Perbet and A. Maki, Homogeneus superpixels from random walks, *MVA.*, 2011

[11] C. Xu and J.J. Corso. Evaluation of super-voxel methods for early video proccesing, *Computer Vision and Pattern Recognition*. 2012.

[12] A. Shekhovtsov, I. Kovtun and V. Hlavac. Efficient MRF deformation model for non-rigid image matching, *Computer Vision and Pattern Recognition*. 2007.

[13] J. Sun, N.N Shen and H.Y. Shum. Stereo matching using propagation belief, *TPAMI*. 2003.

[14] C. Rother, V. Kolmogorov and A. Blake. Grabcut: Interactive foreground extraction using iterated graph cuts, *SIGGRAPH*. 2004.

[15] L. Yang, Y. Guo, X. Wu and X. Wang. A new video segmentation approach: Grabcut in local window, *Soft Computing and Pattern Recognition*. 2011.

[16] Y. Wu, J. Lim and M.-H. Yang. Online object tracking: A benchmark, *Computer Vision and Pattern Recognition*. 2013.

[17] S. Baker, D. Scharstein, J.P. Lewis, S. Roth, M.J. Black and R. Szeliski. A Database and Evaluation Methodology for Optical Flow, *International Journal Computer Vision*. 2013.

[18] Y. Boykov, M-P. Jolly. Interactive Graph Cuts for Optimal Boundary & Region Segmentation of Objects in N-D images, *International Conference on Computer Vision*. 2013.

[19] W. Li, D. Cosker and M. Brown. An anchor patch based optimization framework for reducing optical flow drift in long image sequences, *Asian Conference on Computer Vision*. 2012.

[20] T. Crivelli, P.-H. Conze, P. Robert, M. Fradet and P. Perez. Multi-step flow fusion: towards accurate and dense correpondence in long video shots, *British Conference Machine Vision*. 2012.

[21] M. Tao, J. Bai, P. Kohli, and S. Paris. SimpleFlow: A Non-iterative, Sublinear Optical Flow Algorithm, *Computer Graphics Forum, Eurographics*. 2012.

[22] T.B. Dinh, N. Vo, and G. Medioni. Context tracker: Exploring Suporters and Distracters in Unconstrained Environments. *Computer Vision and Pattern Recognition* . 2011.

[23] S. Hare, A. Saffari, and P.H.S. Torr, Struck: Structured Output Tracking with Kernels. *International Conference on Computer Vision*. 2011.

[24] X. Jia, H. Lu, and M.H. Yang. Visual Tracking via Adaptive Structural Local Sparse Appearance Model. *Computer Vision and Pattern Recognition* . 2012.

[25] B. Babenko, M.H. Yang, and S. Belongie. Visual Tracking with Online Multiple Instance Learning. *Computer Vision and Pattern Recognition*. 2009.

[26] Z. Kalal, J. Matas, and K.Mikolajczyk. P-N Learning: Bootstrapping Binary Classiers by Structural Constraints. *Computer Vision and Pattern Recognition*. 2010.

[27] D.J. Butler, J. Wulff, G.B. Stanley, and M.J. Black. A naturalistic open source movie for optical flow evaluation.*European Conference in Computer Vision*. 2012.

[28] G. Farneback. Two-Frame Motion Estimation Based on Polynomial Expansion. *Scandinavian Conference, SCIA* . 2003.

[29] Brox, A. Bruhn, N. Papenberg, J. Weickert. High accuracy optical flow estimation based on a theory for warping. *European Conference in Computer Vision*. 2004.

[30] Berthold K.P. Horn and Brian G. Schunck. Determining Optical Flow. *Artificial Intelligence* . 1981.

[31] Lucas, B., and Kanade, T. An Iterative Image Registration Technique with an Application to Stereo Vision, *International Joint Conference on Artificial Intelligence* . 1981.

[32] Zach, T. Pock and H. Bischof. A Duality Based Approach for Realtime TV-L1 Optical Flow, *Proceedings of Pattern Recognition (DAGM)* . 2007.

[33] T. Brox and J. Malik. Object Segmentation by Long Term Analysis of Point Trajectories, *European Conference in Computer Vision*. 2010.

[34] P. Ochs and T. Brox. Object Segmentation in Video: A Hierarchical Variational Approach for Turning Point Trajectories into Dense Regions, *International Conference in Computer Vision* . 2011.

[35] A. Bugeau and P. Perez. Detection and segmentation of moving objects in highly dynamic scenes, *Computer Vision and Pattern Recognition* . 2007.

[36] S.M. Smith. ASSET-2: Real-Time Motion Segmentation and Shape Tracking. *Computer Vision*. 1995.

[37] Carlo Tomasi and Takeo Kanade. Detection and Tracking of Point Features. *Carnegie Mellon University Technical Report CMU-CS-91-132* , 1991.

[38] A. Hilsmann and P. Eisert. Deformable Object Tracking Using Optical Flow Constraints. *European Conference in Visual Media Production* , 2007.

[39] H-S. Chang and Y-C. Frank Wang. Superpixel-Based Large Displacement Optical Flow. *International Conference on Image Processing* , 2013.

[40] D. Sun, S. Roth J.P. Lewis, and M.J. Black. Learning Optical Flow. *European Conference on Computer Vision* , 2008.

[41] A. Rav-Acha, P. Kohli, C. Rother and A. Fitzgibbon. Unwrap Mosaics: A new representation for video editing. *ACM SIGGRAPH*, 2008.

[42] C. Wu. Towards Linear-time Incremental Structure from Motion. *International Conference on 3D Vision*, 2013.

[43] M. Zucchelli. Optical Flow Based Structure From Motion. *Doctoral Dissertation for the Royal Institute of Technology, Stockholm*, 2002

[44] Y. Cheng. Mean Shift, Mode Seeking, and Clustering, *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 1995.

[45] Gary Bradski and Adrian Kaehler: Learning OpenCV: Computer Vision with the OpenCV Library. *O'Reilly Media*, 2008.

[46] S. Rangarajan and S.T. Birchfield. Spatiograms versus histograms for region-based tracking. *Computer Vision and Pattern Recognition*, 2005.

[47] H. Grabner, M. Grabner, and H. Bischof. Real-time Tracking via On-line Boosting. *British Machine Vision Conference*. 2006.

[48] Z. Kalal, K. Mikolajczyk, and J. Matas. Tracking-Learning-Detection. *Pattern Analysis and Machine Intelligence*. 2011.

[49] F.L. Bookstein. Principal Warps: Thin-Plate Splines and the Decomposition of Deformations. *Pattern Analysis and Machine Intelligence*. 1989.