DLIG: Direct Local Indirect Global Alignment for Video Mosaicing

Marc Vivet, Brais Martinez, *Student Member, IEEE,* and Xavier Binefa

***Abstract*—In this paper, we present a framework for real-time mosaicing from video sequences recorded from an uncalibrated pan tilt zoom camera based on multiframe registration. To this end, a new frame alignment algorithm, the direct local indirect global (DLIG), is presented. The key idea of the DLIG alignment is to divide the frame alignment problem into the problem of registering a set of spatially related image patches. The registration is iteratively computed by sequentially imposing a good local match and global spatial coherence. The patch registration is performed using a tracking algorithm, so a very efficient local matching can be achieved. We use the patch-based registration to obtain multiframe registration, using the mosaic coordinates to relate the current frame to patches from different frames that partially share the current field of view. Multiframe registration prevents the error accumulation problem, one of the most important problems in mosaicing. We also show how to embed a kernel tracking algorithm in order to obtain a precise and extremely efficient mosaicing algorithm. Finally, we perform a quantitative evaluation of our algorithm, including a comparison with other alignment approaches, and studying its performance against interlaced videos and illumination changes.**

***Index Terms*—Interlaced, kernel tracking, multiframe alignment, real-time, video mosaicing.**

I. Introduction

VIDEO MOSAICING consists on generating an image that summarizes the entire visible area of a video sequence. This is done by selecting a reference frame from a video sequence and putting the rest of frames in correspondence with it.

Classically, mosaics have been used for video compression. In [1], Irani *et al*. described techniques of video compression for video-storage and video transmission applications, which consist on generating the mosaic image of the scene from the video sequence and encoding the residuals of the frames relative to the mosaic image. In [2], Lee *et al*. improved the video coding efficiency by exploiting the layered representation inside the context of MPEG-4. There, the mosaic image is a layer composed by the pixels in an object visible through the entire scene, which is called Sprite.

Besides video compression, mosaics are useful for many other applications. For instance, mosaics are used to generate a representation of the terrain using videos recorded from the sky [3] or from underwater [4], for video indexing [5], for motion detection [6], or for video surveillance [7], [8]. A recent application built upon a mosaicing algorithm is the generation of a dynamosaicing [9], which summarizes the whole scene by automatically generating dynamic mosaics, ignoring the chronological order of the actions in a video sequence and focusing on the activities themselves.

The process of putting a frame in correspondence with the reference frame is what is called frame alignment. Depending on how the alignment is performed, it is possible to divide the mosaicing algorithms present in the literature under two different criteria. First of all, they can be divided with respect to the information used to estimate the alignment, resulting on a division between direct or featureless methods, and indirect or feature-based methods [10].

Direct methods are those who use all the available image pixels to compute an image-based error, which is then optimized to compute the alignment. The results can be accurate, but they require an already accurate initialization and typically results in high computation costs. When the input is a video sequence, the accurate initialization can be assumed by using the previous alignment as an initialization. In a video sequence, consecutive frames have a large overlapping region. This means that the transformation that places two consecutive frames into the mosaic are similar, what we call *frame locality*.

In contrast, indirect methods compute the alignment by using a set of image locations. The point locations are selected to have representative local image properties so they are likely to be detected independently of the point of view of the camera. Finally, the transformation is computed as the one maximizing the matching between both the set of locations over the current frame and the set of locations over either another frame or the mosaic. Some examples of this family are [10] and [11]. Both extract a set of scale-invariant feature transform (SIFT) features from all the images and then use algorithms for eliminating outliers and identifying the correspondences between images. In particular, the first one uses a probabilistic model based on RANSAC while the second uses a geomorphic reverse measurement.

Feature-based methods are very interesting since they provide good performance and can estimate complex transfor-

Manuscript received September 2, 2010; revised February 28, 2011; accepted April 11, 2011. Date of publication May 12, 2011; date of current version December 7, 2011. This work was supported by the Universitat Autonoma de Barcelona PIF Research Grant 3394337802 B00SN0192 and by the Industry, Tourism and Commerce Ministry of the Spain Government SEDUCE Research Project (CENIT-2008 1012). This paper was recommended by Associate Editor F. Lavagetto.

The authors are with the Universitat Pompeu Fabra, Barcelona 08018, Spain (e-mail: marc.vivet@uab.cat; brais.martinez@upf.edu; xavier.binefa@upf.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSVT.2011.2154770

mations. They are suited for mosaicing from a set of images not required to belong to a video sequence, since they do not require an initialization as direct methods do. In contrast, they can be computationally expensive. Furthermore, for the case of video mosaicing, direct methods make a better use of the frame locality property, transforming the frame alignment problem from global search to local estimation.

The second distinction between the existing mosaicing algorithms can be done in terms of whether each frame is put in correspondence with the previous frame of the sequence or directly with the mosaic. These approaches are called frame-to-frame alignment and frame-to-mosaic alignment.

The frame-to-frame alignment approach consists on registering the current frame with the previous one, obtaining its alignment by accumulating the inter-frame registration with the previous alignment. This is the most common approach in the mosaicing literature, but it has an important drawback. Since each inter-frame registration produces a small error, computing the alignment as the accumulation of inter-frame registrations produces an accumulation of the error over time. This results in a bad alignment, especially important when the camera has loops in its trajectory. Some techniques, like [12] and [13], use a graphical model to deal with misalignments caused by the error accumulation. In these articles, nodes represent frames and edges represent adjacency in time or space. The mosaic image is generated by finding the optimal path that minimizes the alignment error. However, this approach is hardly applicable for real-time scenarios. First of all, complexity grows over time as new frames are added, while the minimization has to be repeated regularly. Second, if the mosaic has to be visualized, reestimating means warping and interpolating all the mosaic. Last, considering a reduced set of frames for lowering the computational cost might lead to not considering frames spatially but not temporally related.

Another possible approach is frame-to-mosaic alignment, which consists on registering the current frame directly with the mosaic image [14], or using instead a representation of the mosaic (like pixel mean or pixel median for each frame aligned). This approach received some attention as a solution for preventing the error accumulation problem [15], [16]. Frame-to-mosaic alignment does not require prohibitive memory storage nor excessive computation time, but the alignment accuracy is affected. In this approach, either the frame has to be transformed to be matched with a mosaic region or vice versa. Small errors in the registration over the frame coordinates can be magnified when the registration is transformed into the mosaic coordinates. Similarly, using the mosaic coordinates is not a solution either. This effect is what we call precision degeneration (see Section II-B for a more in-depth explanation). Furthermore, the computational drawback associated is twofold: the mosaic needs to be explicitly computed and the frame needs to be warped before matching it with the mosaic. This also implies matching warped images obtained through pixel interpolation.

The remainder of this paper is structured as follows. Section II provides an intuitive overview of our method, explores the limitations of the related literature, and analyzes why our method can overcome them. Section III provides a more detailed description of the method, paying special attention to the DLIG alignment algorithm (Section III-B). We detail in Section IV the experiments conducted, including a set of quantitative and qualitative evaluations of the proposed algorithm. Final remarks and future work can be found in Section V.

II. Overview of the Proposed Method

A. Direct Local Indirect Global Alignment

We aim to achieve real-time video mosaicing. It is therefore natural to use the frame locality property to reduce the computational cost. Frame alignment is defined as a problem of local estimation; a similar approach to that of direct methods. But instead of computing the alignment of the whole image, which is computationally expensive, we compute the alignment of a discrete set of image subregions. Each of the subregions is matched using a tracking algorithm and an independent target model. Visual tracking is an important field of computer vision, and many robust and efficient algorithms have been proposed, a knowledge we take advantage of. However, the risk of errors during the tracking should be taken into account. For this reason, we also impose global consistency on the set of estimations. In an ideal scenario, the local matchings should be consistent with a global image transformation. It should be possible to compute it as in indirect methods, and the point matchings should be consistent with the transformation. For this reason, we iteratively alternate the local alignment stage with a global transformation estimation. This last step is used to correct erroneous trackings and to impose global consistency with respect to the interframe transformation. This process yields robust consistent interframe matchings. We call this methodology direct local indirect global (DLIG) alignment method, since the local alignment is performed like in direct methods, while the global transformation is computed as in indirect methods.

B. Multiframe Matching

Our method performs multiframe matching as a way to prevent error accumulation problems. We show that it is capable of doing so while preventing the problems of frame-to-mosaic approaches. However, it is fist necessary to provide a more in-depth explanation of the problems associated with frame-to-mosaic approaches.

We referred in the introduction to the problem of error accumulation in frame-to-frame alignment approach, and how multiframe matching through the use of graphical models can make real-time applications unfeasible. Frame-to-mosaic approaches alleviate such a problem, but present other drawbacks instead. First of all, the mosaic has to be computed explicitly. Second, we match deformed images (meaning interpolating). Last and most importantly, it suffers from what we called the precision degeneration problem. Here, we want to give a more insightful explanation of this last problem and to show how to prevent all of these problems when performing multiframe matching.

The precision degeneration problem happens when the coordinate system of the current frame i is very different from

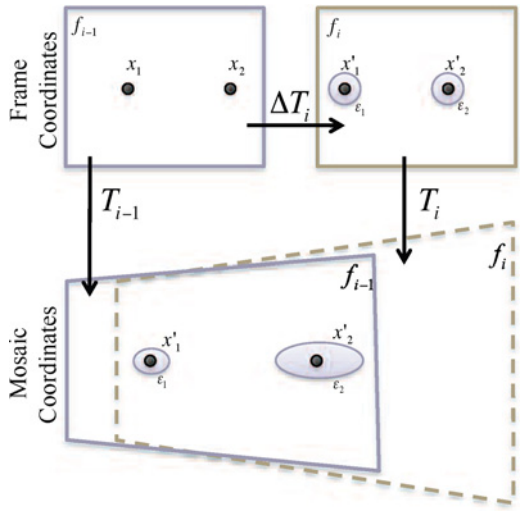


Fig. 1. Precision degeneration. Euclidean distance is equivalent to point matching error. The circles drawn around points x'_1 and x'_2 show points at equal distance to their central locations. The interframe distortion is almost non-existent. In contrast, when projected onto the mosaic, they can suffer a great distortion, which hinders frame-to-mosaic alignment.

the reference coordinate system. In this case, the computation of the alignment based on a set of matched points becomes unstable. If alignment T_i severely distorts the matched frame, so it does with the errors of the matching. Therefore, a small matching error in the mosaic can correspond to large matching errors in frame i or its reciprocal.

More formally, given two frame points so $x = \bar{x} + \vec{\epsilon}$, their distance in frame coordinates is $\|\vec{\epsilon}\|$, while the distance between their mosaic equivalents is $\|T_i(\vec{\epsilon})\|$. Since T_i is not an Euclidean transformation, same error $\|\vec{\epsilon}\|$ at different frame locations can yield (very) different errors in mosaic coordinates. Therefore, performing the estimation of alignment i in mosaic coordinates or in frame coordinates can yield very different estimates.

This effect is shown in Fig. 1. In this figure, two points x'_1 and x'_2 located at frame i are shown, together with a set of locations (circles around them) placed at a distance ϵ. When performing mean square error (MSE) estimation, the Euclidean distance is equivalent to the error, so the circles represent points producing all the same matching error. The problem arises when transforming these points of equal error into mosaic coordinates, since they are transformed into distorted ellipses.

In order to prevent these performance issues, we perform instead a multiframe matching. More precisely, we relate some locations typically belonging to different frames (called reference points for frame i) to a set of locations on the current frame i. The reference locations are selected using the mosaic coordinate system, so we can guarantee that they are seen in the current frame. We assume we have an initial hypothesis of the current frame alignment $\hat{T}_i^0$. Then, we can project the reference points into the mosaic coordinate system using their respective alignments, and then backproject them into the current frame through $\hat{T}_i^0$. This generates a set of current frame locations, which are dependent on $\hat{T}_i^0$. Each of these points is considered as the center of a image patch, which will be used to perform the matching. At this stage, we can use the DLIG alignment to refine this first hypothesis into a final alignment. By following this methodology, the matching is performed using undistorted information from the frames at which the reference points belong. Since we do not use mosaic aspect information, it is not necessary to explicitly build the mosaic, deforming and interpolating patches for matching is avoided, and the precision degeneration problem is prevented.

It is important to note that this formulation is equivalent to considering the reference points to lay on the mosaic, since alignments of previous frames are fixed. It is just necessary to take into account that the visual information used to perform their matching with points at the current frame will be extracted from their original frames. For simplicity, we will say these points lay on the mosaic.

As a final remark, note that this approach is only possible because we do not perform the frame matching typical of direct methods, but instead we split the frame information into parts that can be related to different frames. Since the mosaic efficiently summarizes the spatial information of the scene, it is natural to take advantage of it in order to obtain the frames spatially related to the current frame. But for the matching itself, we use the original frame information.

III. Method Description

Given a sequence formed by frames $\{f_i\}_{i=1:N}$, and a reference frame f_r, the aim is to compute for each frame f_i the alignment T_i that puts it in correspondence with the reference frame. We compute alignment T_i by using two sets of points put in correspondence, one from the current frame and another from the mosaic. In our paper, T_i is restricted to be a 2-D projective transformation, due to the good tradeoff provided between flexibility and simplicity on the alignment estimation. Thus, given a set of points $X' = \{x'_j\}$ in f_i coordinates and their corresponding points $X = \{x_j\}$ in the mosaic coordinates, we compute a transformation T_i that puts in correspondence these sets (being in our case a 3×3 transformation matrix[1]).

In the reminder of this section, we will first describe how we estimate correspondences between frame points and mosaic points. Afterward, the DLIG transformation estimation is described in detail. Then, a summary of our algorithm is provided.

A. Local Point Correspondences Using a Tracking Algorithm

Each of the reference points is put in correspondence with a point on the current frame by using a template-based tracking algorithm. We consider each reference point to be the center of an image patch. Such patch is used to extract a target representation (the model) to be used by the tracking algorithm. Then, this model is used to match the region against the current frame, and the center of the matched region is put in correspondence with the reference point. We note as $I_k(x_j)$ to an image patch within frame k centered at point x_j.

[1]We use a 3×3 transformation matrix, since we only focus on 2-D projective transformations. Nevertheless, the method can be extended to other types of transformations, as polymorphic or piecewise affine transformations, despite that they cannot be modeled using a 3×3 transformation matrix. Note that since the alignment includes translation, we use the homogeneous representation of points.

More precisely, given a patch representation φ, matching a template within an image consists on finding the ideal Δp so that $\varphi(W(I_i, \Delta p))$ is the closest under some criterion to the model $\varphi(I_k(x_j))$. Here, Δp contains the parameters of a transformation W over I_i, defining the patch from where the representation is computed, as, e.g., its center and size. It is important to note that the matching obtained depends on the type of transformation allowed. Δp can contain the parameters of, e.g., a translation, an affine, or a projective transformation. It is also important to note that at each iteration of a tracking algorithm $\varphi(W(I, \Delta p))$ has to be computed. This generally implies computing the patch warp $W(I, \Delta p)$, and its computational cost depends on the tracking algorithm.

In practice, it is possible to obtain a model for the template matching from the mosaic image, and we would obtain a frame-to-mosaic alignment algorithm. However, to prevent the problems linked to this approach, we compute the model representation directly from frame information. More precisely, provided that x_j is a reference point in mosaic coordinates, we backproject it to its original frame, which we note as $o(j)$, and compute the point representation using a subregion of $f_{o(j)}$ centered around $T_{o(j)}^{-1}(x_j)$. Therefore, the representation used is formally defined as follows:

$$\varphi(x_j) = \bar{\varphi}\left(T_{o(j)}^{-1}(x_j)\right) \tag{1}$$

where $\bar{\varphi}$ is an equivalent representation of φ applied to a different image, and we simplify the notation $\varphi(I(x_j))$ as $\varphi(x_j)$.

The representation $\bar{\varphi}$ depends on the tracking algorithm used. For example, it can be the patch intensity values if we use correlation or a kernel-weighted histogram if kernel tracking (KT) is used. Some other tracking algorithms are flexible regarding the representation used.

Each time a new part of the scene is observed, some new reference points are added to the set of reference points and their representation is computed. The definition of the set of reference points is incremental, so it is possible to add more but those added are always kept.

Since the information stored is just a model representation for each reference point (typically a vector), the memory storage benefit of this approach is important respect to approaches requiring storing all the mosaic. Furthermore, at each new frame we are using previously computed representations, so we do not need to recompute them at each step, speeding up the process.

Finally, it is important to remark that the properties of the tracking algorithm used are inherited by the mosaic. For example, a tracking algorithm incapable of coping with varying illumination will fail in such cases, while using a representation robust to illumination changes will boost the performance. We will provide some examples in Section IV, where we show the performance of different tracking algorithms.

B. DLIG Alignment Computation

We now describe the computation of alignment T_i using the DLIG algorithm. We need to find and put in correspondence two sets of reference points: X_i, belonging to the mosaic, and a set X_i', belonging to frame i.

Algorithm 1 DLIG alignment computation

1 $\hat{T}_i^0 = T_{i-1}$;
2 **for** $t \leftarrow 0$ **to** *Until convergence* **do**
3 $\quad \bar{X}_i^t = \left(\hat{T}_i^t\right)^{-1} X_i$;
4 $\quad \bar{X}_i^t \rightarrow \hat{X}_i^t$ using a tracking step;
5 $\quad \hat{X}_i^t \rightarrow W_i^t$;
6 $\quad \left(\hat{X}_i^t, W_i^t\right) \rightarrow \hat{T}_i^{t+1}$ using WMSE;
7 $T_i = \hat{T}_i^{end}$;

We rely on the region matching procedure described in the previous section. However, we impose global consistency over the estimated matchings to prevent local mismatchings that might arise from tracking errors, or from the content of the current frame, as, e.g., when moving objects are present. Algorithm 1 summarizes this process, and its details are provided in the following.

We perform an iterative estimation of T_i, starting with $\hat{T}_i^0 = T_{i-1}$ as the initial guess.[2] Given the set of mosaic points X_i, the first step consists on projecting them into the current frame coordinate system using the current estimate of the alignment, $\hat{T}_i^0$, obtaining a set of points $\bar{X}_i^0 = \left(\hat{T}_i^0\right)^{-1} X_i$ (line 3 of Algorithm 1). Due to the frame locality property, the points $\bar{X}_i^0$ are close to the yet unknown locations X_i', which we aim to estimate.

The tracking algorithm is then applied to the locations $\bar{X}_i^0$. However, instead of sequentially performing a set of iterations, we just apply one (or a small predefined set of) iteration of the tracking algorithm. Through this step, a new estimate of X_i' is obtained, noted $\hat{X}_i^0$ (Algorithm 1, line 4). Furthermore, it is possible to obtain an error associated to each one of the estimates, inherently given by the tracking algorithm. The particular form of such error depends on the tracking algorithm. For example, it is possible to use the sum of square differences between the target model and the representation of the estimated region.

We then combine the direct local step with an indirect global step. More precisely, the indirect global step consists on obtaining a new estimate of alignment i by using the point estimates provided by the direct local step. The transformation estimate is obtained by weighting each of the local estimates (Algorithm 1, line 5). We will define how to compute the weights W_i in the following section, although the errors associated to each point matching play an important role in such definition. The weighted estimation is achieved by using weighted MSE (WMSE) minimization. The resulting alignment estimation is noted as $\hat{T}_i^1$. By adding the indirect global step, the next iteration of the direct local step will be performed over the set of points $\bar{X}_i^1 = \left(\hat{T}_i^1\right)^{-1} X_i$ instead of the points $\hat{X}_i^0$. In this way, the correctly estimated locations help computing the alignment estimate, and the alignment estimate is used to project back the mosaic points onto the current frame, correcting possible errors obtained on the direct local step.

[2]It is possible to add a dynamic model to predict the evolution of T_i and obtain a better initial guess.

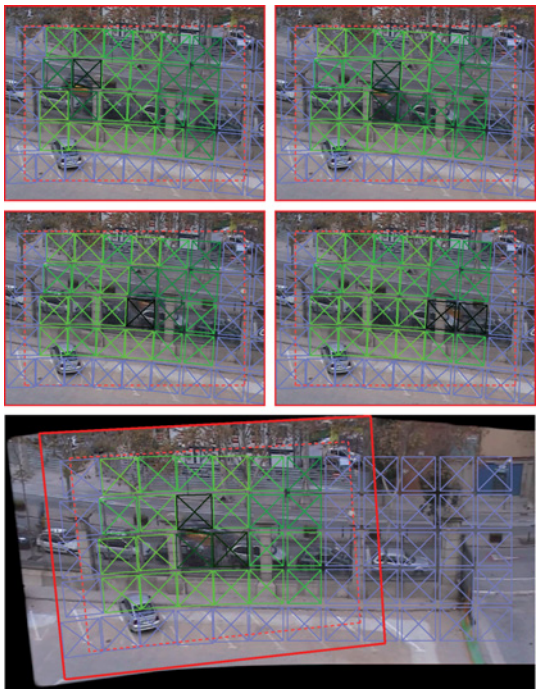


Fig. 2. Green squares are patches centered at $\bar{X}_i^t$. The intensity of the color reflects the weights used by the WMSE. Those regions over the moving car have consistently low weight and therefore have a low impact in the alignment computation. Upper images are four consecutive frames; the lower image represents the corresponding mosaic.

Fig. 2 contains an example showing the robustness of our method to moving objects and changing background. In this sequence, the regions to be matched with moving objects yield high errors and are correctly assigned with a low weight when computing the alignment.

C. Mosaic Construction Using the DLIG Algorithm

Here, we provide the remaining details of our algorithm. First of all, the mosaic is initialized as the first frame of the sequence. Therefore, the first alignment is the identity, $T_1 = I_3$. Afterward, a set of reference points are created using f_1 at locations placed over a regular lattice. This means a set of locations $\{x_j\}_{j=1:N_1}$ from the mosaic are selected, and their representations $\varphi(x_j)$ are computed.

For frame i, a subset of the reference points $\{x_j^i\}_{j=1:N_i}$ are selected. Alignment $i-1$ is used to select them, and to initialize the computation of alignment i. Thus, when projected onto frame i, the selected reference points produce a set $\{x'_j\}_{j=1:N_i} = \{T_{i-1}^{-1}(x_j^i)\}_{j=1:N_i}$. Now, the DLIG alignment is computed as described by Algorithm 1.

In our case, we use two criteria to define the weights used. The first one is the error yielded by the direct local step, noted $\{\varepsilon_j\}$, and its particular form depends on the type of tracking algorithm used. The second criterion is based on the number of times the reference point has been successfully used, noted as τ_j. The resulting form of their computation is

$$w_j = C_2 \cdot \frac{\tau_j \cdot \eta_j}{\sum_k^N \tau_k \sum_k^N \eta_k} \qquad (2)$$

$$\text{where } \eta_j = C_1 \cdot e^{-\frac{N\varepsilon_j}{\sum_k^N \varepsilon_k}}$$

Algorithm 2 Mosaic construction using DLIG alignment (RP stands for reference points)

Data: Video sequence $F = \{f_i\}_{i=1:N}$
Result: Mosaic image M
begin
  • $T_1 = I_3$; $RP = \varnothing$; $M = f_1$;
  • $RP \leftarrow$ Generate RP using f_1;
  for $i \leftarrow 2$ **to** N **do**
    • $RP' \leftarrow$ Select a subset of RP fitted by T_{i-1};
    • DLIG alignment as depicted in Algorithm 1;
    • Update the RP' models using f_i;
    • Update the mosaic M using f_i;
    • $RP \leftarrow$ Add new RPs from f_i if they belong to new parts of the mosaic;
end

where C_1 and C_2 are normalization factors. This definition of the weights makes the estimation robust respect to moving objects in the scene, reflected by a high error (implying low η_j) and to unstable image regions, reflected in the value of τ_j.

This process is iterated until convergence. Again, any convergence criterion can be used. We set two different criteria. The first one consists on a maximum number of iterations, imposed for preventing an excessive computational cost. The lack of convergence can be produced by misestimates of the DLIG algorithm, but also due to natural image situations as moving objects or regions with low texture. We impose a second criterion based on the lack of change in successive iterations, which is a typical convergence criterion.

This process finally results on the estimation of alignment T_i. Once T_i is computed, we update all the reference point models in order to adapt them to the new conditions of the video (e.g., changes in illumination). Any updating strategy can be used. We opt for updating the model as a linear combination of the current model and the performed estimation [17]. Furthermore, we compute the variance of the model in all the occasions it was observed in a frame. This value can be used to lower the weight of badly conditioned reference points, as, e.g., those placed in a region with varying aspect (e.g., a tree, a part of the image with a lot of movement, and others).

The pseudocode of our mosaicing algorithm using DLIG alignment is presented in Algorithm 2.

IV. Experimental Results

In this section, we have three main aims. The first one is to evaluate the performance of our method using different types of tracking algorithms. Second, we confront the performance of the DLIG alignment with frame-to-frame and frame-to-mosaic alignment strategies. In third place, we select the best performing tracking algorithm and show some performance examples. We include in our evaluation interlaced videos and changes in illumination, since they are very common in practical applications.

The different tracking algorithms considered are the following: normalized cross correlation (NCC)-based tracking [18], Lucas-Kanade optical flow (LKOF) [19], SIFT [10], and KT

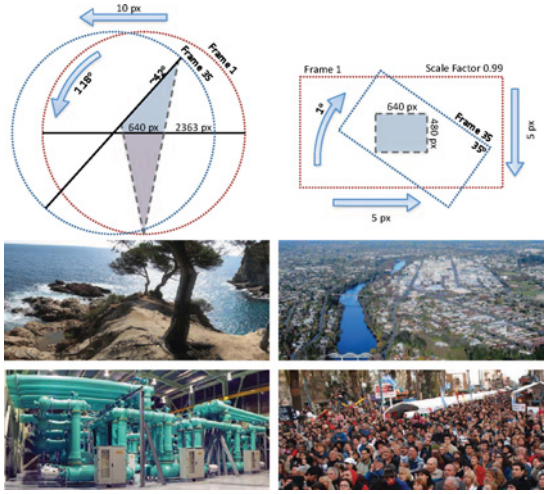


Fig. 3. Top: scheme of how the video sequences are generated (projective pattern on the left, affine one on the right). Bottom: the images used to generate the 16 test sequences.

Fig. 4. (a) Progressive frame from a test sequence. (b) Same frame of the interlaced version of the video sequence.

[20]. They are selected to be representative from different approaches to tracking and for being applied to already existing mosaicing algorithms. NCC is used as a comparison baseline. LKOF is the basis of a large number of mosaicing algorithms, and it is a widely extended tracking paradigms as well. In the case of SIFT, despite not being by nature a tracking algorithm, it can be used to find correspondences between frames. We include it since SIFT features have shown to be very effective when applied to mosaicing. Finally, KT has some properties which make it especially suited for mosaicing purposes. It is robust to small changes in the target aspect, as those produced by lightning and small target deformations, and it is computationally very efficient.

In order to evaluate the performance of the different mosaicing algorithms, we generated a set of synthetic video sequences from large real images, simulating different camera movements. We use synthetically created videos because it is otherwise very difficult to obtain a ground truth allowing a quantitative performance evaluation. More precisely, these videos are generated by applying a transformation pattern to 2363×1073 static images of different characteristics and then cropping a 640×480 patch from the center of the transformed image. We have generated two different transformation patterns. The first one is defined by (3). It contains a projective warp and a translation as follows:

$$T_i = \begin{pmatrix} 1 & 0 & -10i \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} cos(\alpha_i) & 0 & sin(\alpha_i) \\ 0 & 1 & 0 \\ -sin(\alpha_i) & 0 & cos(\alpha_i) \end{pmatrix} \quad (3)$$

Fig. 5. Comparison of light changes in a long interlaced sequence recorded using a Sony handycam DCR-PC110E. The blocks building the image correspond to frame 150, frame 37 000, and their difference. It is important to note the color change, which produces some differences when subtracting the frames.

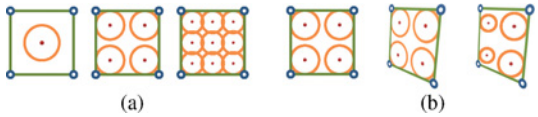


Fig. 6. (a) Region representation using different number of kernels (one, four, or nine). (b) Projective deformation of a region produces anisotropic kernels. It can be approximated using isotropic kernels.

where $\alpha_i = \frac{-0.001\pi i}{180}$. The second pattern is defined by (4). It contains an affine transformation, including scaling, rotation, and translation as follows:

$$T_i = \begin{pmatrix} 0.99^i cos(\beta_i) & -0.99^i sin(\beta_i) & 5i \\ 0.99^i sin(\beta_i) & 0.99^i cos(\beta_i) & 5i \\ 0 & 0 & 1 \end{pmatrix} \quad (4)$$

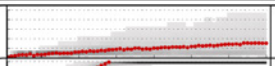
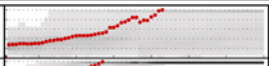
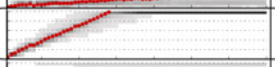
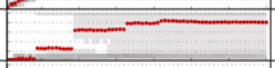
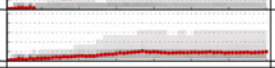
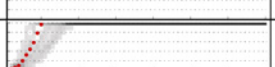
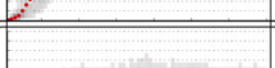
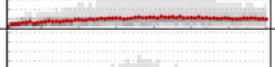
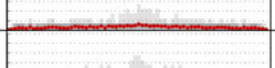
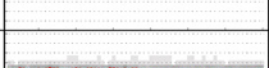
where $\beta_i = \frac{i\pi}{180}$.

The final sequences consist of 35 frames obtained by incrementally applying the transformation patterns and another 35 frames consisting on the reverse of the first 35 frames. This results in 70 frames per sequence, where the first frame and the last frame match. Fig. 3 shows a scheme of how these videos are generated.

Additionally, we created an interlaced version of each of these sequences in order to evaluate the algorithm performance in such conditions. It is important to note that the minimum error to expect in interlaced videos is dependent on the camera movement. The first frame of the sequences is not interlaced and when compared to the rest of frames, which are already interlaced, an error is naturally produced and propagated over all the process (our error measure is an average of pixel position errors). Therefore, the objective in interlaced videos is not to have no error but instead to not accumulate further error throughout the sequence. Fig. 4 shows a comparison between the progressive and the interlaced version of a frame. In total, we have 16 different video sequences for quantitative testing purposes.

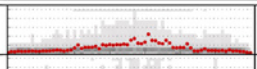
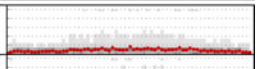
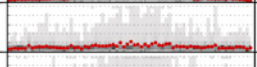
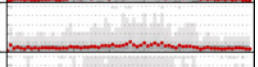
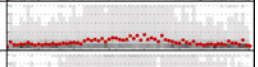
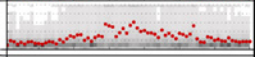
Each test is initialized using a set of reference points placed at a regular 7×5 lattice, using patches of 70×70 pixels to construct their representation.

TABLE I
ALIGNMENT ERROR COMPARISON

		Progressive Videos				Interlaced Videos			
Aligment	**Tracker**	**F.T.**	**• px**	**• px**	**Error/Frame D.**	**F.T.**	**• px**	**• px**	**Error/Frame D.**
Frame to Frame	NCC	–	3.555	2.406		1	18.78	27.10	
	LKOF	–	25.94	15.85		1	26.52	14.97	
	SR	1	12.10	17.05		8	–	–	
	KT	–	2.781	2.170		8	–	–	
Frame to Mosaic	NCC	–	3.162	2.180		8	–	–	
	LKOF	2	4.794	2.367		8	–	–	
	SR	6	–	–		8	–	–	
	KT	–	225.5	195.5		8	–	–	
DLIG Aligment	NCC	–	1.486	0.944		1	5.532	3.459	
	LKOF	–	3.952	1.8849		–	6.259	3.582	
	SR	–	1.240	0.841		8	–	–	
	KT	**–**	**1.186**	**0.810**		**–**	**4.646**	**1.386**	

SR stands for SIFT + RANSAC and KT for kernel tracking. The graphics show in the x-axis the frame number, from 1 to 70, and on the y-axis (ranging from 0 to 20) the computed error in terms of mean for the frame pixels (red) and their quantized distribution (gray). See text for details on the graphic construction. Error/Frame D. means Error/Frame Distribution.

TABLE II
KT DLIG MOSAICING ACCURACY UNDER TWO ILLUMINATION CHANGE PATTERNS

	Global						Block					
		I.V.		Error				I.V.		Error		
I.V. Range	**F**	**•**	**•**	**•**	**•**	**Error/Frame**	**F**	**•**	**•**	**•**	**•**	**Error/Frame**
[-5,5]	–	2.47	1.46	2.77	3.79		–	2.48	1.44	1.65	1.60	
[-10,10]	4	5.15	2.82	2.33	2.43		2	5.00	2.88	2.19	2.28	
[-15,15]	3	7.54	4.27	3.89	5.46		2	7.52	4.30	3.72	4.25	
[-20,20]	6	–	–	11.1	14.9		3	10.1	5.75	5.50	7.11	

I.V. stands for intensity variation. F stands for failed mosaics. The graphics are constructed like in Table I (see text for further explanation).

The results of comparing the different tracking algorithms considered, together with a performance comparison with respect to the different alignment approaches, are shown in Table I. To quantitatively evaluate the mosaicing error, we compute for each pixel the Euclidean distance between its location through the estimated transformation and its real location. We provide the mean and variance of these errors in the first columns. This table also includes graphics summarizing the evolution of the error along the sequence, so, e.g., it is possible to see the error accumulation phenomenon for the case of frame-to-frame alignment. These graphics are composed by 70 concatenated vertical histograms (one per frame), where the x-axis represents frame number (from 1 to 70) and the y-axis shows the error in pixel distance (from 0 to 20). Each of these histograms is constructed by considering the error for each pixel in the corresponding frame for all the test videos. Darker tones of a bin represent high density of pixels having an error corresponding to this bin. We also show over the graphics the mean error using red dots. Note that the errors computed are referred to the accumulated estimations and do not refer to interframe errors.

It is possible to see from the evaluation the consistently superior performance of the DLIG alignment. One important aspect is that the error is not accumulated over the sequences. In contrast, the frame-to-frame approach presents consistent error accumulation problems. For the case of frame-to-mosaic approaches, NCC and LKOF perform well, but the error again increases with the sequence. For the case of frame-to-mosaic, we use a frame to mean mosaic approach. Since the DLIG alignment relies on the tracking performance, it is safe to say

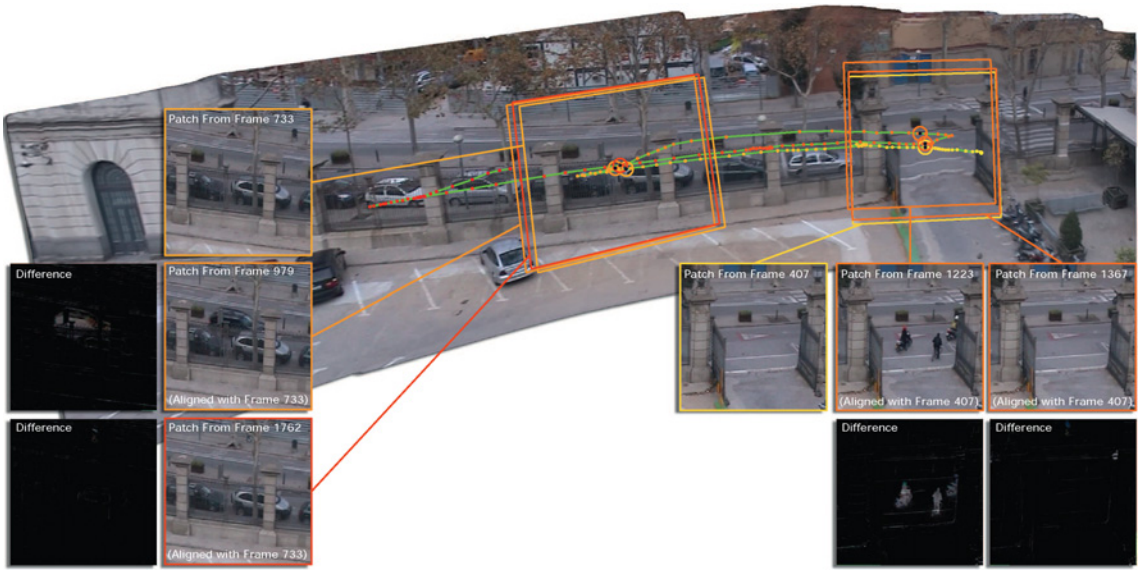


Fig. 7. Mosaic generated with all the sequence (2200 frames). The green line shows the camera trajectory and the dots mark the point of view each ten frames. The color code indicates the frame, where the yellow color is used for the first frames, and the red ones for the lasts. The comparisons results from subtracting frames 733 from 979, and from 1762 (two differences on the left) and 407 from 1223 and 1367 (the two on the right).

that the final mosaicing algorithm inherits the properties and limitations of the tracking algorithms. The most illustrative case is the performance with interlaced videos for SIFT and KT. Whereas they perform very similarly for deinterlaced videos, SIFT completely fails with the interlaced ones. This is due to the nature of the SIFT descriptor, based on orientation of edges, which are completely distorted in interlaced videos. In contrast, KT performs very well with interlaced videos. The representation used in KT, kernel-weighted histograms, is well known for being very robust against blurring. In practice, for KT the representations of the interlaced and deinterlaced version of the same region are very similar.

Given the results of the performance evaluation, we adopt the KT as the most reliable, both in interlaced and progressive videos. It is also the less computationally demanding.

We made some further performance evaluations to decide on the optimal parameters of the representation for KT. More precisely, we selected the optimal number of kernels and whether isotropic or anisotropic kernels should be used. We tested kernel-weighted representations formed by one, four, and nine kernels (see Fig. 6). The experimental results show that there is an improvement between the performance between using four kernels respect to using just 1, but instead using nine kernels leads to almost no performance improvement, while the computational cost grows with the number of kernels. The use of anisotropic kernels further improve the performance (1.186 versus 1.015 of average error, respectively). In order to match the same scene region as seen in two different frames, it would be necessary to consider projective transformations. Therefore, the kernels responsible for creating the patch representation should undergo the same transformation, leading to anisotropic kernels. However, it is possible to approximate them using isotropic kernels, as depicted in Fig. 6. It is therefore reasonable that the obtained results are less accurate. But anisotropic kernels present a computational drawback, since the computational cost growth is significant. When using isotropic kernels instead, it is possible to generate look-up tables. A look-up table is a large set of precomputed kernels at different scales, so whenever a kernel suffers a scaling, it is only necessary to look for it on the table instead of computing its values again. This is possible when the kernels are isotropic, since only the scale can change. For anisotropic kernels instead, the variability is too high, since an anisotropic kernel is defined by three parameters instead of the one required for isotropic kernel. If the computational cost is not a restriction, the use of anisotropic kernels would fit best. In our case, we use isotropic kernels because of the real-time requirement.

In order to test the capability of the DLIG algorithm to resist illumination changes, we designed a modification of the eight synthetic test videos described before. For each of these frames, a random illumination variation pattern was added. The results of this experiment are shown in Table II. We performed two different experiments depending on the illumination pattern added (numbers refer to a scale of $[0, 255]$). In the first one, uniform global noise was added to the frame, where the value of the illumination change was randomly selected within an interval following a uniform distribution. These results are shown in the left part of the table. It is important to note that having an interval of $[-10, 10]$ means the illumination change between two consecutive frames might be of up to 20. The second pattern used dividing the image in 16 blocks, and each of them was transformed using a uniform illumination pattern, again with a value randomly selected using a uniform distribution within a range. It is worth mentioning that these tests are performed without applying any illumination normalization to the frames, which would be recommended for the final application.

It is possible to see that for the block illumination pattern, the results are better than for the global illumination one. Since the intensity of the variation is random, it is expected that some

of the 16 blocks have a smaller illumination change. In these cases, the alignment can rely more on such regions.

The robustness of the alignment method to illumination changes depends on the properties of the tracking algorithm. For these experiments, we used 12-bin kernel-weighted histograms. In practice, the bin width is of about 20. This implies robustness against illumination changes smaller than the width of the bin (like for those changes within the $[-5, 5]$ range), while for comparable or larger illumination changes the matching becomes unstable.

We also tested the performance of our method in a large sequence with slow changes in the illumination, so the performance depends to a large degree on the ability to adapt the model to the ongoing changes. In Fig. 5, we combine blocks of frame 150, frame 37 000, and their difference. It is possible to see the precision of the matching performed, even when there are 36 850 frames in-between, which at 25 f/s adds up to more than 25 min. Again, no illumination normalization was used in this sequence.

In order to provide a qualitative vision of the performance of our method, we show some experiments obtained with real sequences. The first experiment shows that our method does not accumulate error when building a mosaic from large sequence. The obtained result is shown in Fig. 7. We used a sequence containing a typical surveillance situation, where the camera rasterizes a scenario, with mostly panning movement but not only. The trajectory of the camera is shown over the mosaic. The sequence is composed by 2200 interlaced frames recorded using a Sony handycam DCR-PC110E with a resolution of 640×480 pixels, and the video is interlaced. The experiment consists on showing regions of different frames which share the same mosaic coordinates. If the mosaic is perfect these regions should be equal (except for illumination changes or moving objects). For example, frames 733, 979, and 1762 have a significant overlap, and the difference between the overlapping parts is shown. The same is done with frames 407, 1223, and 1367. The difference between these frames is small, even taking into account that the video is interlaced. Only regions with moving objects show a significant difference.

Some more qualitative examples are shown in Fig. 8. The figure shows the mosaics obtained and summarizes the field of view evolution within the sequences.

Fig. 8. Top example shows a 256 frames interlaced video, recorded from an UAV with a resolution of 720×576 pixels. The bottom example is a 1000 frames progressive video, recorded using a handheld photo camera (Cannon ixus 750) with a resolution of 640×480 pixels. A trapezoid is plotted every ten frames showing the field of view, while the color of the trapezoid indicates time, where yellow mark the first frame and red the last one.

A. Memory Requirements and Computational Cost

Our algorithm is capable of performing real-time video mosaicing, considering that the mosaic is not explicitly computed nor visualized. In order to achieve real time also for these functionalities, it is necessary to use multithread programming. For example, [21] achieve it by using CUDA language and two parallel threads.

We have built a C++ single-thread implementation of this algorithm capable or running in real time for videos with high resolution (which we will provide through the author's web). For example, we are capable of processing a 720×512 resolution video, excluding frame load, at 48.04 frames per second using a PC with a Core i7 920 (2.66 GHz) CPU, or at 28.98 frames per second on a laptop with a Core 2 Duo T7200 (2 GHz) CPU. Another important property is that the computational cost remains constant with time.

The memory storage requirements are very small. The amount of stored data grows linearly with respect to the extent of the mosaic, since only the representation of the reference points has to be stored, being each of these representations just a vector.

V. Conclusion and Future Work

We have presented the DLIG alignment which combines the precision of direct methods and the simplicity when estimating complex transformations of indirect methods. We also have shown how to prevent the error accumulation problem and the precision degeneration problem by using a multiframe alignment computation.

Additionally, by embedding a tracking algorithm, we take profit of the frame locality property. We also have shown that it is possible to deal with incorrect matchings by using

weighted alignment estimation, and how to deal with background changes using a model update strategy.

We have tested different tracking algorithms, concluding that KT is the most suited. It can deal with progressive and interlaced videos with high accuracy and low computational cost and memory requirements.

As future work, we will introduce new target representations robust to lightning changes. Also we will extend the current work to situations with parallax. To this end, we should be capable of identifying two different clusters in the movement estimation of the reference points.

REFERENCES

[1] M. Irani, S. Hsu, and P. Anandan, "Video compression using mosaic representations," *Signal Process.: Image Commun.*, vol. 7, nos. 4–6, pp. 529–552, 1995.

[2] M.-C. Lee, W.-G. Chen, C. Lin, C. Gu, T. Markoc, S. Zabinsky, and R. Szeliski, "A layered video object coding system using sprite and affine motion model," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 7, no. 1, pp. 130–145, Feb. 1997.

[3] F. Caballero, L. Merino, J. Ferruz, and A. Ollero, "Homography based Kalman filter for mosaic building applications to UAV position estimation," in *Proc. Int. Conf. Robot Automat.*, 2007, pp. 2004–2009.

[4] N. R. Gracias and J. Santos-Victor, "Underwater video mosaics as visual navigation maps," *Comput. Vision Image Understand.*, vol. 79, no. 1, pp. 66–91, 2000.

[5] M. Irani and P. Anandan, "Video indexing based on mosaic representations," *Proc. IEEE*, vol. 86, no. 5, pp. 905–921, May 1998.

[6] M. Vivet, B. Martinez, and X. Binefa, "Real-time motion detection for a mobile observer using multiple kernel tracking and belief propagation," in *Proc. Iberian Conf. Patt. Recog. Image Anal.*, 2009, pp. 144–151.

[7] A. Mittal and D. Huttenlocher, "Scene modeling for wide area surveillance and image synthesis," in *Proc. IEEE Conf. Comput. Vision Patt. Recog.*, Jun. 2000, pp. 160–167.

[8] R. Collins, A. Lipton, T. Kanade, H. Fujiyoshi, D. Duggins, Y. Tsin, D. Tolliver, N. Enomoto, and O. Hasegawa, "A system for video surveillance and monitoring," Robotics Inst., Pittsburgh, PA, Tech. Rep. CMU-RI-TR-00-12, 2000.

[9] A. Rav-Acha, Y. Pritch, D. Lischinski, and S. Peleg, "Dynamosaicing: Mosaicing of dynamic scenes," *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. 29, no. 10, pp. 1789–1801, Oct. 2007.

[10] M. Brown and D. G. Lowe, "Automatic panoramic image stitching using invariant features," *Int. J. Comput. Vision*, vol. 74, no. 1, pp. 59–73, 2007.

[11] W. Wei, H. Jun, and T. Yiping, "Image matching for geomorphic measurement based on SIFT and RANSAC methods," in *Proc. Int. Conf. Comput. Sci. Softw. Eng.*, vol. 2. 2008, pp. 317–320.

[12] H. S. Sawhney, S. C. Hsu, and R. Kumar, "Robust video mosaicing through topology inference and local to global alignment," in *Proc. 5th Eur. Conf. Comput. Vision*, vol. 2. 1998, pp. 103–119.

[13] E.-Y. Kang, I. Cohen, and G. Medioni, "A graph-based global registration for 2-D mosaics," in *Proc. Int. Conf. Patt. Recog.*, 2000, pp. 1257–1260.

[14] H. S. Sawhney and R. Kumar, "True multi-image alignment and its application to mosaicing and lens distortion correction," *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. 21, no. 3, pp. 235–243, Mar. 1999.

[15] F. Winkelman and I. Patras, "Online globally consistent mosaicing using an efficient representation," in *Proc. IEEE Int. Conf. Syst., Man Cybern.*, vol. 4. Oct. 2004, pp. 3116–3121.

[16] A. Bevilacqua and P. Azzari, "A fast and reliable image mosaicing technique with application to wide area motion detection," in *Proc. 4th Int. Conf. Image Anal. Reconstruct.*, 2007, pp. 501–512.

[17] I. Matthews, T. Ishikawa, and S. Baker, "The template update problem," *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. 26, no. 6, pp. 810–815, Jun. 2004.

[18] J. P. Lewis, "Fast normalized cross-correlation," *Vision Interface*, pp. 120–123, 1995.

[19] S. Baker and I. Matthews, "Lucas-Kanade 20 years on: A unifying framework," *Int. J. Comput. Vision*, vol. 56, no. 3, pp. 221–255, 2004.

[20] G. D. Hager, M. Dewan, and C. V. Stewart, "Multiple kernel tracking with SSD," in *Proc. IEEE Conf. Comput. Vision Patt. Recog.*, Jun.–Jul. 2004, pp. 790–797.

[21] S. Lovegrove and A. J. Davison, "Real-time spherical mosaicing using whole image alignment," in *Proc. Eur. Conf. Comput. Vision*, 2010, pp. 73–86.

Marc Vivet received the B.Sc. degree in computer science in 2007 and the M.Sc. degree in computer vision and artificial intelligence in 2008, both from the Universitat Autónoma de Barcelona, Barcelona, Spain, where he is currently a Ph.D. student under the supervision of Dr. Binefa.

Brais Martinez (S'10) received the B.Sc. degree in mathematics from the Universidade de Santiago de Compostela, Santiago de Compostela, Spain, in 2003, and the M.Sc. degree in computer science from the Universitat Autónoma de Barcelona, Barcelona, Spain, in 2006, where he is currently a Ph.D. student under the supervision of Dr. Binefa.

Currently, he is a Research Assistant with the Universitat Pompeu Fabra, Barcelona, in the CMTech Group led by Dr. Binefa.

Xavier Binefa received the B.Sc. degrees in mathematics from the University of Barcelona, Barcelona, Spain, and in computer engineering from the Universitat Autónoma de Barcelona, Barcelona, and the Ph.D. degree in computer vision from the Universitat Autonoma de Barcelona in 1996.

He was an Associate Professor with the Department of Computer Science until 2009, when he was contracted by the Universitat Pompeu Fabra, Barcelona, as an Associate Professor with the Department of Information and Communication Technologies, where he currently leads the CMTeach Research Group.