On-line Video Motion Estimation by Invariant Receptive Inputs

Marco Gori, Marco Lippi, Marco Maggini, Stefano Melacci
Department of Information Engineering and Mathematics
Via Roma 56, 53100 - Siena, Italy
{marco,lippi,maggini,mela}@diism.unisi.it

Abstract

In this paper, we address the problem of estimating the optical flow in long-term video sequences. We devise a computational scheme that exploits the idea of receptive fields, in which the pixel flow does not only depends on the brightness level of the pixel itself, but also on neighborhood-related information. Our approach relies on the definition of receptive units that are invariant to affine transformations of the input data. This distinguishing characteristic allows us to build a video-receptive-inputs database with arbitrary detail level, that can be used to match local features and to determine their motion. We propose a parallel computational scheme, well suited for nowadays parallel architectures, to exploit motion information and invariant features from real-time video streams, for deep feature extraction, object detection, tracking, and other applications.

1. Introduction

Motion estimation is one of the basic primitives underlying tracking algorithms as well as detection approaches in dynamic scenes. Determining the trajectories of moving objects, especially in long-term sequences, crucially depends on the quality of local motion estimation schemes. A lot of effort has been and is still spent by the scientific community, in order to improve the prediction quality in different applicative contexts, and particularly in determining the *optical flow* [8] (for recent contributes, see [1, 12, 18, 21] and references therein). Optical flow estimation techniques aim at predicting the velocity field of the pixels, from which motion estimation of objects can be devised and used, for example, to improve the detection or tracking quality [4, 17]. The applicability of many powerful estimation techniques is frequently limited by their computational burden, due to execution times that are not suitable for real-time applications (e.g., see [21] for a time comparison). Recently, parallel computational schemes (mostly based on GPUs) have been applied to improve the flow estimation speed [19]. Moreover, the estimated velocity field must be frequently paired with local image features on top of which higher-level decision mechanisms can be built.

In this paper, we address the problem of estimating the optical flow in long-term video sequences, merging feature extraction and movement estimation in a unique framework. We devise a computational scheme that exploits the idea of receptive fields [9], in which the pixel flow does not only depend on the brightness (or color level) of the pixel itself, but also on information on its neighborhood. This principle is frequently adopted in local feature estimation approaches [14] and in deep convolutional networks [11], both for supervised [10] and unsupervised learning [6]. The proposed approach is based on the definition of *transformation invariant receptive inputs*, that is, receptive units that are invariant to geometric (affine) transformations of the input data. The construction of such receptive inputs allows us to build a database of visual patterns with arbitrary detail level, which can be employed to match local features and to determine their motion. The idea of feature matching for optical flow estimation with undergoing geometric transformations has been the subject of several recent studies [12, 13, 2], although the computational times are generally prohibitive for real-time applications (see [12]).

We propose a parallel computational scheme that is well suited for nowadays parallel architectures (GPU, multi-core systems, etc.), and that allows users to tune the maximum displacement of the matching procedure and the quality of the matching itself accordingly to the available hardware and the required response time. We show some preliminary examples of real-time optical flow estimation in long video sequences, comparing different settings and sizes of the generated video-feature database. We plan to apply our scheme to build a deep convolutional feature extraction system for scene parsing and object detection in video streams.

The paper is organized as follows. Section 2 introduces the receptive inputs, while Section 3 describes the transformation parameters and the video-feature database. Section 4 characterizes the local motion estimation scheme that leads to optical flow. Preliminary experiments are in 5, and conclusions are drawn in Section 6.

2. Receptive inputs

Let $\mathcal{I}$ be a video frame, and let $x \in X$ be the 2D coordinates of a pixel in $\mathcal{I}$. We model a *receptive field* of x using a set of $\mathcal{N}$ Gaussians that are located in the neighborhood of the pixel. The mean x_k of each Gaussian $g_k(x_k, \mu)$, $k = 1, \ldots, \mathcal{N}$, is expressed in the reference frame centered in x, and all the g_k share the same variance μ.

A *receptive input* of x is the outcome of convolving the receptive field with $\mathcal{I}$. We indicate the receptive input with $\xi(x, \mathcal{I}) = [\xi_1(x, \mathcal{I}), \ldots, \xi_{\mathcal{N}}(x, \mathcal{I})]$, where

$$\xi_k(x, \mathcal{I}) = g_k \otimes \mathcal{I} \propto \int_X e^{-\frac{\|x_k + x - \tau\|^2}{2\mu^2}} \mathcal{I}(\tau) d(\tau) \; . \quad (1)$$

The vector $\xi(x, \mathcal{I})$ is a filtered representation of the neighborhood of x, whose detail level depends on the number of Gaussians $\mathcal{N}$ and on the variance μ. The position of the centers x_k is arbitrary, and, in this work, we select them on a uniform grid of unitary edge centered in x.

We can parametrize the receptive field by incorporating affine transformations of the Gaussian centers. Given a 2D affine map A (discarding translations, due to the convolutional nature of the fields), we can equivalently rewrite it as the composition of three 2D transformations and a scale parameter, that is, $A = \sigma R_{\varphi_1} U_{\varphi_2} R_{\varphi_3}$, where $\sigma > 0$ and R_{φ_1}, U_{φ_2}, R_{φ_3} are the following 2×2 matrices,

$$R_{\varphi_1} = \begin{bmatrix} \cos\varphi_1 & -\sin\varphi_1 \\ \sin\varphi_1 & \cos\varphi_1 \end{bmatrix}, \; U_{\varphi_2} = \begin{bmatrix} \frac{1}{\cos\varphi_2} & 0 \\ 0 & 1 \end{bmatrix},$$

$$R_{\varphi_3} = \begin{bmatrix} \cos\varphi_3 & -\sin\varphi_3 \\ \sin\varphi_3 & \cos\varphi_3 \end{bmatrix},$$

with $\varphi_1 \in [0, 2\pi]$, $\varphi_2 \in [0, \frac{\pi}{2})$, and $\varphi_3 \in [0, \pi)$ [15]. We consider these continuous intervals to be discretized into the grids Φ_1, Φ_2, Φ_3, and, similarly, we collect in Σ a set of discrete samples of σ (starting from $\sigma = 1$). The domain $\mathcal{T} = \Sigma \times \Phi_1 \times \Phi_2 \times \Phi_3$ collects all the possible *tuples of transformation parameters*.

Given a tuple $T \in \mathcal{T}$, each component of the newly parameterized receptive input $\xi(x, T, \mathcal{I})$ is then

$$\xi_k(x, T, \mathcal{I}) \propto \int_X e^{-\frac{\|\sigma R_{\varphi_1} U_{\varphi_2} R_{\varphi_3} x_k + x - \tau\|^2}{2\sigma^2\mu^2}} \mathcal{I}(\tau) d(\tau) \; , \quad (2)$$

where the effect of the scale σ is not only limited to the position of Gaussian centers, but also to the width of the Gaussian[1]. Computing the receptive input for all the pixels $x \in X$ and for all the transformations in $\mathcal{T}$ only requires to perform $|\Sigma|$ Gaussian convolutions per-pixel, independently on the number of centers $\mathcal{N}$ and on the size of the grids Φ_1, Φ_2, Φ_3. As a matter of fact, only σ affects the shape of the Gaussian functions[2]. If we memorize the result of the convolutions for all the $x \in X$, then we can compute the receptive inputs for any tuple T simply performing lookup operation.

The receptive input can be additionally improved by including invariance to local changes in brightness and contrast, that we model by normalizing $\xi(x, T, \mathcal{I})$ to zero-mean and unitary L_2 norm.

3. Transformation invariant matching

A key concept of the proposed algorithm, that makes it different from (most of) the existing optical flow estimates, is that it does not only aim at estimating the velocity vector field. As we will describe in this section, it also extracts patterns that are invariant with respect to geometric (affine) transformations of the input. This is crucial for building motion-based decision mechanisms, where the mere estimation of the velocity field must be paired with some kind of semantics on the local contents around each pixel (feature extractors, object detectors, tracking algorithms, etc.).

Given a video frame $\mathcal{I}_t$ at time t, we assign a unique transformation tuple T^{x_t} to each pixel x_t, and, as a consequence, we have a unique receptive input $\xi(x_t, T^{x_t}, \mathcal{I}_t)$ (2). The tuple is selected such that $\xi(x_t, T^{x_t}, \mathcal{I}_t)$ minimizes the mismatch to a discrete sample of the receptive inputs of the video that were processed up to the current frame-pixel. Let Q be the collection of such receptive inputs[3], and let $d(\cdot, \cdot)$ be a metric that is used to compare them. We indicate with ξ (without any arguments) the data stored in Q.

Formally, we associate T^{x_t} to x_t of $\mathcal{I}_t$ such that

$$(T^{x_t}, \; \xi^{x_t}) = \arg\min_{T \in \mathcal{T}, \; \xi \in Q} d(\xi, \xi(x_t, T, \mathcal{I}_t)) \; , \quad (3)$$

where $\xi^{x_t} \in Q$ is the closest element to $\xi(x_t, T^{x_t}, \mathcal{I}_t)$. As a result, each pixel is "described" by its transformation parameters T^{x_t} and by its closest representative $\xi^{x_t} \in Q$. Any learning scheme can be applied to the data in Q to extract higher-level information, while the matching criterion (3) allows us to link each pixel to the predictions on the data belonging to Q.

Duplicates are not stored in Q, and a tolerance ϵ is introduced to define the detail level of the sampling. More precisely, after having solved (3), if $d(\xi^{x_t}, \xi(x_t, T^{x_t}, \mathcal{I}_t)) > \epsilon$, then $\xi(x_t, T^{x_t}, \mathcal{I}_t)$ is added to Q, otherwise it is associated with ξ^{x_t} (3). We end up with a well-spaced set of points, in which each $\xi_i \in Q$ is the centre of a (closed) ball of radius ϵ, $B(\xi_i, \epsilon)$, and no other $\xi_j \in Q$ falls in the ball that is centered in $\xi_i \neq \xi_j$. This mechanism allows us to filter out noise on the data captured by the receptive fields and to

[1] See [15] for additional details.

[2] The non-uniform scaling of U_{φ_2} should generate anisotropic Gaussians (see [15]), that we do not consider here both for simplicity and to reduce the computational burden.

[3] We do not explicitly indicate the dependance of Q on the frame and pixel indices to keep the notation simple.

not consider slight local changes in the video frames. The value of ϵ can be selected to tune the size of Q with respect to the available memory budget. The set Q is a *compressed affine-invariant* representation of the video data.

We set $Q = \emptyset$ at the beginning of the video, and then we progressively add data to it as long as the video is processed. The resulting Q depends on the characteristics of the video stream, and on the order in which pixels are elaborated. We devise a *blurring scheme* to reduce the latter dependency, in which the variance μ (2) is initialized to a large value and progressively decreased with an exponential decay that depends on t. The first frames are strongly blurred, and only a few ξ's are added to Q for each $\mathcal{I}_t$ (even just one or none). The progressive decrement of μ allows us to reduce the number of additions to Q for each following $\mathcal{I}_t$ (the tuple assigned to the first addition to Q is arbitrary).

The data in Q are distributed on the surface of a sphere $\mathcal{S}^{\mathcal{N}-2}$ of radius 1, since one of the dimensions is lost due to the L_2 normalization, and the other due to the data centering. If we consider $d(\cdot,\cdot)$ to be the classical Euclidean distance, one can equivalently use a similarity measure based on the scalar product $\langle\cdot,\cdot\rangle$ to compare receptive inputs, and then the constraint $d(\xi_i,\xi_j) > \epsilon$ translates to $\langle\xi_i,\xi_j\rangle < \gamma_\epsilon$.

Due to the role of the tolerance ϵ, the set Q is an ϵ-net of the subspace of $\mathbb{R}^{\mathcal{N}}$ that contains all the observed receptive inputs ($\hat{Q}$). Such nets are standard tools in the field of metric spaces, frequently exploited in many search problems [7]. As a consequence, Q is both ϵ-*covering* and $\frac{\epsilon}{2}$-*packing*,

1. $d(\xi_i,\xi_j) > \epsilon, \ \ \forall \xi_i,\xi_j \in Q$ ($\frac{\epsilon}{2}$-*packing*)
2. $\hat{Q} \subseteq \cup_{\xi_i \in Q} B(\xi_i,\epsilon)$ (ϵ-*covering*)

where the second condition tells us that all the observed inputs in $\hat{Q}$ are either elements of Q or duplicates of them, up to tolerance ϵ.

Theorem 3.1 *There exists a finite set Q for any processed video stream.*

Proof: The components of the zero-mean and L_2 normalized receptive inputs (2) are bounded in $[-1,1]$, so the set $\hat{Q}$ is bounded, and the metric space $(\hat{Q},d)$ is closed and bounded. $\hat{Q}$ is compact, that, in turn, implies that it is complete and totally bounded. If a set is totally bounded then there exists a *finite* ϵ-net of it for all $\epsilon > 0$ [20]. □

The cardinality of Q can be estimated by using some theoretical results on the metric space $(\hat{Q},d)$. From [3], and considering the spherical data distribution, we have that $|Q|$ is $O(1/\epsilon^{\mathcal{N}-1})$. Notice that this estimate is only due to the properties of the selected metric and on the dimensionality of the metric space itself, and not due to the intrinsic dimensionality of the data. As a matter of fact, the data may lie only in a lower dimensional manifold over the surface of $\mathcal{S}^{\mathcal{N}-2}$, and then the estimate could be further reduced. Another useful descriptive index is related to the *covering density* of Q, given that we can cover the spherical data in $\hat{Q}$ with differently distributed ϵ-nets. Such density is the average number of spherical caps covering a point of $\hat{Q}$, i.e.

$$\nu(\hat{Q}) = \nu(\mathcal{S}^{\mathcal{N}-2}) = \sum_{\xi_i \in Q} \frac{vol\left(\hat{Q} \cap C(\xi_i, r_\epsilon)\right)}{vol(\hat{Q})} ,$$

where $C(\xi_i, r_\epsilon)$ is the cap centered in ξ_i and with base radius r_ϵ, generated by the intersection of $B(\xi_i,\epsilon)$ with the surface of $\mathcal{S}^{\mathcal{N}-2}$, and $vol(\cdot)$ indicates the volume of a set [5, 16]. The cardinality of the ϵ-net stored in Q is directly proportional to $\nu(\hat{Q})$. Following Theorem 1 in [5], we can devise a lower bound on $\nu(\mathcal{S}^{\mathcal{N}-2})$.

Theorem 3.2 *([5]) For any $\eta > \frac{1}{2}$, a sphere $\mathcal{S}^{\mathcal{N}-2}$ of any radius and growing dimension $\mathcal{N} \to \infty$ can be covered with spherical caps of half-chord $r_\epsilon = 1$ with density*

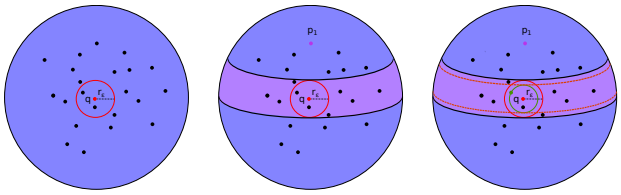
$$\nu(\hat{Q}) = \nu(\mathcal{S}^{\mathcal{N}-2}) \leq \eta |Q| \ln |Q| .$$

The result presented for $r_\epsilon = 1$ can be generalized to any r_ϵ by rescaling the radius of $\mathcal{S}^{\mathcal{N}-2}$ accordingly.

Solving (3) requires to perform $|\mathcal{T}|$ full nearest-neighbor (NN) searches in Q. Theoretical results proved that a NN search on an ϵ-net can be performed in constant time, but the memory requirements for the data structures are exponential in the dimensionality $\mathcal{N}$ [7]. We propose a pivot-based strategy for exact NN search, that is both feasible and effective, exploring the relationships among different instances of the same receptive input, generated with different transformation tuples $T_1,\ldots,T_{|\mathcal{T}|}$.

Starting from the first tuple T_1, we have to search for the NN in Q of the query receptive input $q := \xi(x,T_1,\mathcal{I})$. Let α_ϵ be the angle corresponding to a Euclidean spacing of ϵ between two receptive inputs. We select a random pivot element p on the surface of $\mathcal{S}^{\mathcal{N}-2}$, and compute the angle $\alpha_{(p,q)}$ between p and q as the inverse cosine of dot product $\langle p,q\rangle$. If the NN to q is closer than ϵ, then it is a duplicate of an element in Q, and it is guaranteed to lie on the surface of the *spherical segment* defined by the points whose angles from p belong to $[\alpha_{(p,q)} - \alpha_\epsilon \ , \ \alpha_{(p,q)} + \alpha_\epsilon]$[4], as depicted in Figure 1, therefore strongly reducing the search space. Moreover, a full search can be early stopped (without any loss) once we find ξ_i for which $d(q,\xi_i) \leq \frac{\epsilon}{2}$, due to the properties of ϵ-nets. Finding the subset of Q that belongs to the spherical segment can be efficiently performed by keeping the data in Q sorted w.r.t. the pivot-angles, $\alpha_{(p,\xi_i)}$.
We can reduce the spherical segment whenever we find a point whose angle with q is smaller than α_ϵ, and we can always consider the smallest segment size when processing the following transformation tuples, $T_2,\ldots,T_{|\mathcal{T}|}$, reducing the burden of the NN searches. See Figure 1 (a-c). If the

[4]We always assume that the result of adding or subtracting angles is bounded in $[0,\pi]$.



(a) (b) (c)

Figure 1. The (dark-purple) spherical surface on which the data (black dots) lie. (a) A query point (red dot) q defines an ϵ-ball that corresponds to the basis of a spherical cap of radius r_ϵ. (b) Given a pivot p_1, the surface of the target spherical segment is depicted in light purple. (c) Whenever we find a point whose angle with q is smaller that α_ϵ, we can reduce the segment and restrict the search space to the area bounded by the dotted-orange lines.

NN is farther than ϵ, then we can both perform a full search on the discarded data, or keep the best solution found so far (if any) to devise an approximation of the global solution. As long as the number of points and the dimensionality increase, a single pivot p may not be enough to significantly restrict the search space. In those cases, more pivots can be used, leading to consider only the data that belong to the intersection of the multiple spherical segments.

4. Local motion estimation

Solving (3) for all x_t of frame $\mathcal{I}_t$, in order to determine the pairs (T^{x_t}, ξ^{x_t}), is an unfeasible operation in online video processing, even using the described pivot-based scheme. In order to reduce the computational burden, we define a local motion estimation procedure that exploits the pairs $(T^{x_{t-1}}, \xi^{x_{t-1}})$, computed for all x_{t-1} of frame $\mathcal{I}_{t-1}$. We assume that the scene smoothly changes over time (camera movements, object movements, etc.), and we introduce $\delta(x_t)$, that is the map that associates x_t with x_{t-1}, i.e. the (reverse) *optical flow* between the pair of frames.

We define the following problem

$$\min_{\delta} \quad R[\delta] \tag{4}$$

$$s.t. \quad \xi^{x_t} = \xi^{\delta(x_t)} \tag{5}$$

$$T^{x_t} \approx T^{\delta(x_t)} \tag{6}$$

$$d(\xi^{x_t}, \xi(x_t, T^{x_t}, \mathcal{I}_t)) \leq \epsilon \tag{7}$$

$$d(\xi^{x_t}, \xi(x_t, T^{x_t}, \mathcal{I}_t)) \leq \zeta \cdot d_{t-1}^{\xi^{\delta(x_t)}} \tag{8}$$

in which $R[\cdot]$ is a regularizer that penalizes large displacements in the map δ (i.e. we aim at associating pixel coordinates that are close). Constraint (5) enforces the map to assign the same receptive input of Q to the mapped pixels, thus enforcing visual coherence. Notice that this is a stricter requirement than what is commonly done in optical flow estimation, in which only the brightness of the pixel is considered in constraining schemes. The receptive input has a wider coverage of the neighborhood of the pixel, hence allowing more informed associations. Constraint (6) is fulfilled whenever T^{x_t} is equivalent to $T^{\delta(x_t)}$, or a slight perturbation of it, formalized by the $\approx$ operator[5]. This allows the map to tolerate small changes in the geometry of the receptive field, that may be due to the assumed smooth changes that incur between $\mathcal{I}_{t-1}$ and $\mathcal{I}_t$. The inequality of (7) is required to keep a valid association with respect to the ϵ-net of Section 3. The last constraint (8) introduces $d_{t-1}^{\xi^{\delta(x_t)}}$, that is the distance between the receptive input at coordinates $\delta(x_t)$ and its associated $\xi^{\delta(x_t)}$, computed while processing the previous frame. The constant $\zeta > 1$ enforces an improvement in the matching distance (left hand side), or it limits the eventual worsening to a fixed factor.

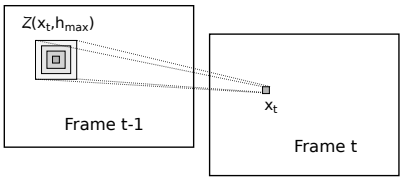


Figure 2. For each x_t, we look for a valid mapping $\delta(x_t)$ over a set of coordinates centered around x_t in $\mathcal{I}_{t-1}$. Incrementally larger contours $\mathcal{Z}(x_t, h)$ are considered (depicted with different intensities) for different values of $h \leq h_{max}$.

We minimize (4) in a greedy way, by performing a reduced search. Let h_{max} be the maximum allowed displacement of the map $\delta(\cdot)$. We use the notation $\mathcal{Z}(x, h)$ to refer to the set of coordinates belonging to the $h \times h$ squared contour centered in x, with $h \in [1, h_{max}]$ (Figure 2). For each x_t, we search for a feasible solution by restricting the codomain of $\delta(x_t)$ to iteratively increasing contours $\mathcal{Z}(x_t, h)$, from $h = 1$ to $h = h_{max}$, in order to minimize $R[\delta]$ (4). Whenever we find a solution that satisfies (5,6,7,8) for a given h, we stop the search procedure, avoiding any additional increment of $\mathcal{Z}(x_t, h)$. Moreover, the codomain element that corresponds to such found solutions (that is $\delta(x_t)$), is excluded from any other instances of the codomain of $\delta(\cdot)$, i.e. it cannot be associated to any other pixel different from x_t. We also have to update the distance-to-Q value $d_{t-1}^{\xi^{\delta(x_t)}}$, that will be used in (8) when estimating the optical flow for the next pair $\mathcal{I}_t, \mathcal{I}_{t+1}$. We set it to $\min\left(d(\xi^{x_t}, \xi(x_t, T^{x_t}, \mathcal{I}_t)), d_{t-1}^{\xi^{\delta(x_t)}}\right)$ in order to avoid a progressive relaxation of the ζ-based bound (8). In other words, we track a receptive input among several consecutive frames, geometrically transforming it, until the matching quality is good enough (and ζ tunes the quality level).

For each value of h, the proposed greedy search is performed over all the still-unsolved coordinates $x_t \in X$. This

[5] We limit perturbations to a single transformation parameter (out of 4) of the tuple T^{x_t}, that can be substituted with one of its nearest neighbors in the selected discrete grids.

approach strongly reduces the number of conflicting solutions with respect to a complete search for all the contours $h = 1, \ldots, h_{max}$ that is sequential on the coordinate x_t. Consider that for a given value of h we can have multiple coordinates x_t for which $\delta(x_t)$ maps to the same x_{t-1}, and we tolerate this due to the assumption of δ not being a strictly-speaking bijective function (border effects, scene changes, etc.).

Evaluating the feasibility of a solution requires to compute a receptive input for the considered T^{x_t} (2) and to evaluate its distance from the candidate ξ^{x_t} (7,8). Some candidates may be repeated along $\mathcal{Z}(x_t, h)$ for the same T^{x_t}, so that we can reduce the number of computations by keeping track of the already computed distances. As long as we increase h, the number of still-unsolved coordinates $x_t \in X$ is reduced, so that tracking large displacements will be usually performed only on a few pixels.

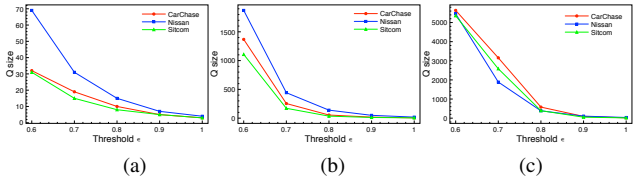
The pixels x_t for which no solution was found up to $\mathcal{Z}(x_t, h_{max})$, are processed with the full searches of Section 3, and their displacement field is discarded. The same holds for large constant regions, that give rise to constant receptive inputs that cannot be normalized to unitary L_2 norm.

4.1. Parallel computational scheme

The optical flow estimation described in Section 4 can be easily implemented on parallel hardware architectures, such as GPUs and multi-core CPUs. For each contour size h, the search procedure can be parallelized on the pixel coordinates, i.e. each computational unit (core) can focus on searching a valid solution for a single pixel or for a small batch of still-unsolved pixels. Each value of h defines a computational stage, while full searches (Section 3) on the unmatched pixels represent an additional stage. There are no data races to check, and write operations are independently performed for each pixel. During the first stages (in which most of the pixels have to be processed) the average speedup is almost linear in the number of computational units. Moreover, parallel architectures are popular for their efficiency in convolution operations such as (2). We implemented the proposed algorithm by exploiting a multi-thread scheme that can efficiently run on multi-core CPUs, allowing scalability to large resolution images.

5. Experiments

We present some preliminary experiments on three different videos taken from the LTDT workshop data set: Sitcom, 09_carchase and NissanSkylineChaseCropped, processed at their original resolution. If not otherwise stated, we employed a discrete grid of 16 angles for the in-plane rotation quantization (φ_1), 3 values for tilt angles (φ_2), up to 6 values for φ_3 (whose quantization depends on the value assumed by φ_2 [15]) and 5 values for the scale σ. These parameters produce $|\mathcal{T}| = 880$ tuples.



(a) (b) (c)

Figure 3. Q size as a function of the tolerance ϵ: we report results for 3×3 (a), 5×5 (b) and 7×7 (c) receptive fields, respectively.

A first experiment was designed o analyze the number of stored templates $|Q|$ as a function of tolerance ϵ. We considered five different values for ϵ (0.6, 0.7, 0.8, 0.9, 1.0) and three different receptive field dimensions $\mathcal{N}$ (3×3, 5×5 and 7×7 grids), running our algorithm on the three selected videos. Results in Figure 3 show that, with the same threshold ϵ, larger receptive fields produce larger Q sets, as more different details are captured by wider grids.

A second experiment was conceived so as to measure the impact of transformation invariance on the number of stored templates. To this aim, we tested the algorithm by selecting three additional different sets of transformation tuples: in the first case we only incorporate in-plane rotations (φ_1 angle), in the second case, we consider in-plane rotations and scale changes (φ_1 and σ), while in the third case we embed all three rotations (φ_1, φ_2, φ_3). Figure 4 shows the results obtained on the three different videos, highlighting the crucial impact of using invariances for reducing $|Q|$. Note that we set a memory budget of 512MB, therefore enforcing a maximum available size for Q: with only in-plane rotations, such maximum number of templates is always stored.

Figure 5 shows some examples of the produced optical flow for three frames extracted from each of the considered videos (with $\epsilon = 0.7$ and 5×5 grids). Each pixel is colored with the angle of the predicted motion, using the hue angle color wheel (same color indicate same motion direction, while similar colors indicate similar angles).

The computational cost of the algorithm greatly depends both on the receptive fields size, and especially on Q set size. In the experiments reported in Figure 5, our system can process around 3-4 fps at their original resolution using 6 threads of a i7-3970X cpu at 3.50GHz. While processing videos at 320×240 resolution, real-time performance can be achieved even on a two years-old Intel Core-i7 laptop.

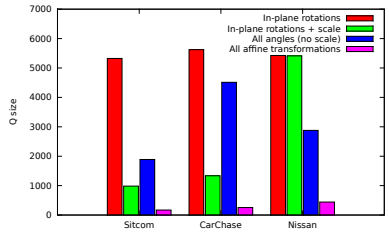
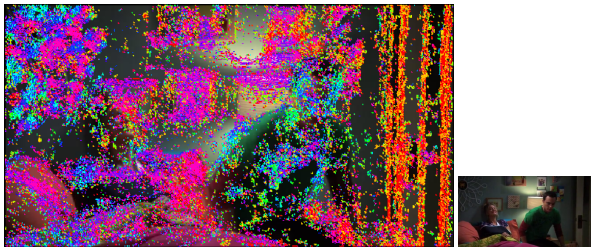
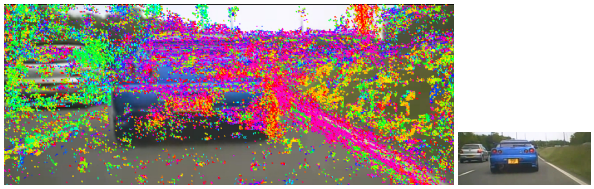


Figure 4. Size of Q when not exploiting all invariances.



(a)

(b)



(c)

Figure 5. Optical flow directions depicted over three sample frames (rightmost images). Same color indicate same motion direction, while similar colors indicate similar angles. In (a), the boy is sitting and his contour is mostly green/cyan, while the camera is zooming, hence the border objects look as if moving towards outside the image (pink/left or orange/right). In (b), the car is indentified by a red/yellow pattern, while the street is mostly pink and violet, due to camera movements. In (c), the blue car is overtaking the grey one, the optical effect being that of two different movement directions.

6. Conclusions and future work

We presented an optical-flow estimation algorithm for online video streams, based on the idea of affine-invariant receptive inputs. Information coming from the neighborhood of each pixel is exploited by enforcing coherence between pairs of receptive inputs. The extracted affine-invariant patterns can be used to build higher-level decision mechanisms, hence exploiting motion in feature extractors, object detectors, and trackers. The algorithm can be easily parallelized. Additional noise filtering and informative spatial regularization are planned in future work.

References

[1] S. Baker, D. Scharstein, J. P. Lewis, S. Roth, M. J. Black, and R. Szeliski. A database and evaluation methodology for optical flow. *Int. Journ. of Comp. Vis.*, 92(1):1–31, 2011.

[2] C. Barnes, E. Shechtman, D. B. Goldman, and A. Finkelstein. The generalized patchmatch correspondence algorithm. In *Eur. Conf. on Comp. Vis.*, pages 29–43, 2010.

[3] K. L. Clarkson. Nearest-neighbor searching and metric space dimensions. *Nearest-Neighbor Methods for Learning and Vision: Theory and Practice*, pages 15–59, 2006.

[4] D. Decarlo and D. Metaxas. Optical flow constraints on deformable models with applications to face tracking. *Int. Journ. of Comp. Vis.*, 38(2):99–127, 2000.

[5] I. Dumer. Covering spheres with spheres. *Discrete & Computational Geometry*, 38(4):665–679, 2007.

[6] M. Gori, S. Melacci, M. Lippi, and M. Maggini. Information theoretic learning for pixel-based visual agents. In *Eur. Conf. on Comp. Vis.*, pages 864–875, 2012.

[7] S. Har-Peled and B. A. Raichel. Net and prune: A linear time algorithm for euclidean distance problems. In *ACM Sympos. on theory of computing*, pages 605–614. ACM, 2013.

[8] B. K. P. Horn and B. G. Schunck. Determining optical flow. *Art. Int.*, 17(1-3):185–203, 1981.

[9] D. H. Hubel and T. N. Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of Physiology*, 160(1):106, 1962.

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1106–1114, 2012.

[11] Y. LeCun and Y. Bengio. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361, 1995.

[12] M. Leordeanu, A. Zanfir, and C. Sminchisescu. Locally affine sparse-to-dense matching for motion and occlusion estimation. In *ICCV*, pages 1721–1728, 2013.

[13] C. Liu, J. Yuen, and A. Torralba. Sift flow: Dense correspondence across scenes and its applications. *IEEE Trans. on Patt. Anal. and Mach. Int.*, 33(5):978–994, 2011.

[14] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. Journ. of Comp. Vis.*, 60(2):91–110, 2004.

[15] J.-M. Morel and G. Yu. Asift: A new framework for fully affine invariant image comparison. *SIAM Journal on Imaging Sciences*, 2(2):438–469, 2009.

[16] Y. Rabani and A. Shpilka. Explicit construction of a small ϵ-net for linear threshold functions. *SIAM Journal on Computing*, 39(8):3501–3520, 2010.

[17] P. Sand and S. J. Teller. Particle video: Long-range motion estimation using point trajectories. *Int. Journ. of Comp. Vis.*, 80(1):72–91, 2008.

[18] D. Sun, S. Roth, and M. J. Black. Secrets of optical flow estimation and their principles. In *Int. Conf. on Comp. Vis. and Patt. Rec., CVPR*, pages 2432–2439, 2010.

[19] N. Sundaram, T. Brox, and K. Keutzer. Dense point trajectories by gpu-accelerated large displacement optical flow. In *Eur. Conf. on Comp. Vis.*, pages 438–451, 2010.

[20] W. Sutherland. *Introduction to metric and topological spaces*. Oxford University Press, 1975.

[21] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid. Deepflow: Large displacement optical flow with deep matching. In *Int. Conf. on Comp. Vis.*, pages 1385–1392, 2013.