Affine Shape Adaptation of Blobs Moving in 3D Space

Jorge Sánchez and Eduardo Destefanis
Centro de Investigación en
Informática para la Ingeniería
UTN, Córdoba, Argentine
Email: {jsanchez,edestefanis}@scdt.frc.utn.edu.ar

Reinhard Klette and Sandino Morales
The *.enpeda..* Project
Tamaki campus
The University of Auckland, New Zealand
Email: {r.klette@auckland.ac.nz,pmor085@aucklanduni.ac.nz}

***Abstract*—The paper studies new constraints that characterize a 3D–motion field as observed from the relative motion of a camera. Such constraints are derived from the relative change in size of observed local image regions over time.**

To consider the image distortions that arise in a projective camera, a modified affine shape adaptation scheme is proposed for the case of blob detection, with an emphasis on robustness under important viewpoint changes and changes in lighting conditions. The resulting features and constraints are used to characterize the motion of an ego-vehicle by means of their navigation angles.

We present results on synthetic as well as on real-world image sequences.

I. Introduction

The estimation of motion fields is still a challenging task for vision-based driver assistance systems (DAS), where motion vectors are often relatively long even if sequences are taken at a frame rate of more than 30 Hz. This paper suggests a way to derive 3D directions of observed 2D motion vectors, which allows a more consistent interpretation of calculated motion fields.

Note that a 3D direction of a motion vector is not yet defining its pose, which would also require to identify its position (e.g., via stereo analysis). Algorithms for identifying the 3D pose of projected motion vectors (known as *range flow*) have been studied, for example, in [19] (and subsequent papers by the same authors). However, the applied methodology differs from the one suggested in this paper.

Our approach uses information provided by observed changes in size (scale) of local image regions over time, when a single camera moves relatively to the scene. This is known to be a very important source of information for the visual perception of motion.

We consider a first order (affine) approximation to model the projective distortions that arise during the imaging process. This is justified by assuming that surface patches in the world are locally planar.

Image regions considered in our work are *blobs*, which corresponds to locally uniform regions of the image. This kind of feature has demonstrated to be the most stable one under viewpoint changes as well as other conditions [17].

Technically, we start with the affine adaptation scheme as proposed in [4], [8] and its versions in [1], [16], characterized by some modifications proposed for the case of blob detection. The resulting affine image features are used to model some characteristics of relative 3D–motion of the camera.

The paper is organized as follows: Section II gives a brief overview of the method employed for the estimation of image deformations. Section III provides some considerations regarding the algorithm presented in [16] for the case of affine blob detection, and proposes corresponding modifications. Section IV presents new constraints for a 3D trajectory of a surface patch in terms of its projected images along this trajectory, and those are used to estimate the so-called *navigation angles*. Finally, Sections V and VI present some experiments and conclusions, respectively, regarding synthetic and real-world data.

II. Affine shape adaptation

Based on scale-space theory (see [6], [7], [22]), the affine shape adaptation algorithm was proposed in the context of *shape from texture* as a way to iteratively estimate the affine deformation of an image region (see [4], [8]), where suchs deformations are estimated using the multi–scale second moment descriptor, wich is defined as follows

$$\mu(\cdot;\Sigma_a,\Sigma_b) = g(\cdot;\Sigma_b) * \left(\nabla L(\cdot;\Sigma_a)(\nabla L(\cdot;\Sigma_a))^T\right) \quad (1)$$

with

$$g(\mathbf{x};\Sigma) = \frac{1}{2\pi\sqrt{\det(\Sigma)}} \exp\left(-\frac{\mathbf{x}^T\Sigma^{-1}\mathbf{x}}{2}\right)$$

In the above expressions, the symbol $*$ represents the convolution operator, $L:\mathbb{R}^2\times\mathcal{M}_{SPD}\to\mathbb{R}$ is the affine scale-space representation of the image,[1] and $\nabla L_f(\cdot;\Sigma_a)$ its gradient, computed at the (anisotropic) scale Σ_a. The matrix (1) can be seen as the second moment of a stochastic variable, where the scale-matrix Σ_b reflects the integration region over which the statistics is collected.

It follows that the computation of the second moment descriptor involves convolutions with non–uniform Gaussian kernels which do not obey (in general) the separability property of their uniform counterparts; thus they do not allow for efficient recursive implementations [2], [23]. In [1] and [14]

[1] $\mathcal{M}_{SPD}$ denotes the set of semi-positive definite matrices.

such convolutions were carried out in a transformed domain, obtained by *whitening* the shape matrix of the corresponding Gaussian kernels. More formally, let $\Sigma \in \mathcal{M}_{SPD}$ be the covariance matrix of a non-uniform Gaussian, defining a quadratic form as follows:

$$\mathbf{x}^T\Sigma^{-1}\mathbf{x} \;=\; \mathbf{x}^T\left(a\Sigma_0^{\frac{1}{2}}\Sigma_0^{\frac{1}{2}}\right)^{-1}\mathbf{x} = \left(\Sigma_0^{-\frac{1}{2}}\mathbf{x}\right)^T\frac{1}{a}\left(\Sigma_0^{-\frac{1}{2}}\mathbf{x}\right)$$

where $\Sigma^{\frac{1}{2}} \in \mathcal{M}_{SPD}$ denotes the square root matrix of Σ. Thus, smoothing with an anisotropic kernel, given by Σ in the original domain, is equivalent to a smoothing with a circularly symmetric kernel $\eta = \Sigma_0^{-\frac{1}{2}}\xi$ of variance a in the transformed domain. Also, in order to reduce the search space, the Σ–matrices are usually restricted to be proportional to each other, as $\Sigma_a = a\Sigma$ and $\Sigma_b = b\Sigma$ for some matrix Σ.

A. Properties of linear transformations and fixed points conditions

Let B be an invertible linear transformation of the spatial domain $\mathbb{R}^2$ and $f(\xi) = h(B\xi)$ the transformed image under B. Let L_f and L_h, both from $\mathbb{R}^2 \times \mathcal{M}_{SPD}$ into $\mathbb{R}$, be the affine-scale space representations of images f and h, respectively. It can be shown [8] that under a such (invertible) linear transformation, the second moment matrix descriptor behaves as

$$\mu(\mathbf{q}; a\Sigma_f, b\Sigma_f) = B^T \cdot \mu_h(\mathbf{p}; a\Sigma_h, b\Sigma_h) \cdot B \qquad (2)$$

where $\mathbf{p} = B\mathbf{q}$ and $\Sigma_h = B\Sigma_f B^T$. Following [8], let $\mu_f(\mathbf{q}; \Sigma_{a,f}, \Sigma_{b,f}) = M_f$ and let consider that the covariance matrices are computed in such a way that the following two conditions

$$\Sigma_{a,f} = aM_f^{-1} \quad \Sigma_{b,f} = bM_f^{-1} \qquad (3)$$

are met. Considering a linear transformation of the spatial domain, and using (2) and (3), follows that

$$\Sigma_{a,h} = B\Sigma_{a,f}B^T = aBM_f^{-1}B^T = aM_h^{-1}$$

This shows that the fixed point conditions (3) are preserved under an invertible linear transformation. The considered iterative algorithm for shape adaptation is based on such a property.

B. Transformation Properties of Corresponding Regions

Consider (as before) two images f and h. Let us suppose that they are images of a surface that can be locally approximated by a plane. Under this condition, the images can be related to each-other by an unknown affine transformation.

Let us also suppose that, for corresponding points on those images, the shape adaptation matrix was independently computed. As showed in [15], the linear transformation B can be recovered, up to an arbitrary orthogonal matrix W, from the shape adaptation matrices as follows:

$$B = \mu_h^{-\frac{1}{2}} W \mu_f^{\frac{1}{2}} \qquad (4)$$

For a given image structure, the change in *scale* of corresponding regions (under an assumed affine transformation) is then given by the determinant of the transformation matrix of Equation (4).

Algorithm 1: Affine blob detection.

1: Initialize $U^{(0)} = \mathbb{I}$.
2: Normalize the window $W(\mathbf{x}_w) = f(\mathbf{x})$ centered at $\mathbf{x}^{(k)}$, with $U^{(k-1)}\mathbf{x}_w^{(k-1)} = \mathbf{x}^{(k)}$.
3: Select the integration scale $a^{(k)}$ from the scale extrema of the normalized operator used for scale selection.
4: Select the local scale $b^{(k)} = \gamma a^{(k)}$ which maximise an isotropy measure, given in terms of the μ-matrix.
5: Refine the spatial location of the interest point and update their location on the original domain, $\mathbf{x}^{(k)}$.
6: Compute the second moment matrix for the updated $\mathbf{x}_w^{(k)}$, $a^{(k)}$ and $b^{(k)}$ and compute their inverse square root $\mu_{adapt}^{(k)} = \mu^{-\frac{1}{2}}(\mathbf{x}_w^{(k)}, a^{(k)}\mathbb{I}, b^{(k)}\mathbb{I})$.
7: Update the shape adaptation matrix $U^{(k)} = \mu_{adapt}^{(k)}U^{(k-1)}$ and normalize it in such a way that $\lambda_{max}(U^{(k)}) = 1$, ensuring that the image patch will be enlarged in the direction of $\lambda_{min}(U^{(k)})$.
8: **if** the affine shape is not yet close enough to a perfectly isotropic structure **then**
9: goto Step 2
10: **end if**

Fig. 1. Basic outline of an affine region detector [14].

III. Affine blob detection

Given an image $f : \mathbb{R}^2 \to \mathbb{R}$ and an initial point with detected location $\mathbf{x}$ and scale a, the algorithm in Figure 1 provides a basic outline of the *Harris–affine* and *Hessian–affine* region detectors (for a detailed description, see [14]); those will serve as our starting point for considerations about blob detection. These detectors use the normalized Laplacian operator for scale selection ([7], [10]) and – as the name suggests – the Harris measure [16] and the determinant of the Hessian matrix, respectively, for the spatial location of feature points.

There exist experimental results which show that the normalized Laplacian provides a maximum number of scale extrema compared to other choices [15]. This operator is also the base of the popular *Difference of Gaussians* (DoG) pyramid used by the SIFT detector [11]. The affine region detector, developed in the presented work, uses the normalized Laplacian both for scale selection and spatial localization, and is therefore referred to as *LoG-affine* detector.

As already said, this adaptation algorithm works in the transformed image domain. The final adaptation matrix U is defined by the concatenation of those $U^{(k)}$-matrices, estimated in all the previous iteration runs, and is initially set equal to an identity matrix.

In order to reduce the number of iterations needed for convergence, the local scale is selected as the one which maximizes the eigenvalue ratio $\mathcal{Q} = \lambda_{min}(\mu)/\lambda_{max}(\mu)$. It is easy to see that $\mathcal{Q} \in [0, 1]$ where $\mathcal{Q} = 1$ corresponds to the perfect isotropic case.

A. Considerations for the case of blob detection

1) Scale selection step: The scale selection step of the shape adaptation algorithm re-estimates (for the current iteration run) the characteristic scale of the brightness pattern associated to the given point with coordinates $\mathbf{x}^{(k)}$. This is

justified by the fact that during each iteration run, the image pattern is *deformed* in order to compensate for the (current estimate of) the affine distortion, changing the relative size of the pattern, thus making a re-detection necessary.

Such an estimation is done by looking for local extrema on the response of the scale-space operator used for scale selection, in a local neighborhood (in scale) of the current estimate $a^{(k)}$. [14] used a range of $a^{(k)} = b^2 a^{(k-1)}$, with $b = 0.7, \ldots, 1.4$.

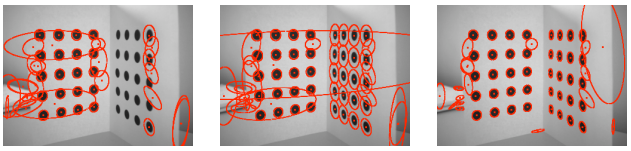
At each step of the algorithm, the warping matrix $U^{(k)}$ is normalized by ensuring $\lambda_{max}(U^{(k)}) = 1$. As pointed out in [15], this has the effect of expanding the image pattern in the direction of the minimum eigenvalue of the U-matrix. This expansion increases the area of the uniform region that represents the actual blob, increasing also their associated characteristic scale.

In the case where the initially detected blob undergoes a significant affine deformation, the corresponding warping of the region may have the impact that scale extrema fall outside of the scale search interval (i.e., we are loosing the point due to a 'no detection of integration scale'). At this moment, and after the $U^{(k)}$-matrix is estimated, it is important to correct the actual estimate of $a^{(k)}$ for the mentioned change in size, before a further estimation of characteristic scale takes place.

One way to introduce such a correction factor is by means of the area ratio of the ellipses associated to the quadratic form $\mathbf{x}^T \mu^{(\nu)} \mathbf{x} = 1$, with $\nu \in \{k, k-1\}$. Thus, the integration scale should be affected by the factor

$$c_k = \frac{\lambda_{min}(\mu^{(k-1)}) \cdot \lambda_{max}(\mu^{(k-1)})}{\lambda_{min}(\mu^{(k)}) \cdot \lambda_{max}(\mu^{(k)})}$$

prior to the normalization step.



(a) (b) (c)

Fig. 2. LoG-affine detector. a) standard algorithm, b) correction of the integration scale prior to the window normalization step and c) normalization of the U-matrix to enforce constant area.

Figure 2 shows a real example of the inclusion of such a correction into the affine adaptation procedure. Figure 2(a) illustrates a detection obtained from the 'standard' algorithm, as described in Section III. The test pattern corresponds to two planes with circular black blobs. These two planes are approximately orthogonal to each other. It can be seen that for most of the blobs on the right plane, the algorithm fails to converge. The shown ellipses correspond to detected blobs, and are displayed in such a way that their area is $\sqrt{2}$ times the detected scale, and their shape is given by the matrix $U^T U$.

Figure 2(b) illustrates a detection obtained by correcting the integration scale in each iteration run of the algorithm, as described above. Observe that, as expected, the number of regions lost by non-detections of characteristic scales were reduced.

2) U-Normalization at important viewpoint changes: As mentioned before, the normalization of the U-matrix by $\lambda_{max}(U^k)$ produces a displacement of the characteristic scales towards larger values. This effect can be appreciated on the right plane of Figure 2(b). With patterns undergoing important viewpoint changes, that could be a serious problem if one relies on the detected scale for further processing.

In order to overcome this, a possible alternative is to normalize the $U^{(k)}$ matrix in such a way that the *area* of blobs remains constant during the iterations. This corresponds to a normalization of the U matrix by the square root of the product of their eigenvalues, $U^{(k)}_{norm} = U^{(k)} / \sqrt{\lambda_{min}(U^{(k)}) \lambda_{max}(U^{(k)})}$.

Figure 2(c) shows results when applying such a U-normalization in the affine adaptation procedure. Note that resulting scales and shapes of the final estimates correspond now to the actual (physical) blobs, even if such blobs are imaged under quite different viewpoints. It is important to note that, with this normalization of the U-matrix, there is no need to correct the integration scale, and we may use factor $c_k = 1$ during all the iteration runs.

3) Lighting Conditions: Besides some modifications in the adaptation scheme, some degree of robustness against changes in lighting conditions is also desirable. Regarding the shape adaptation matrix, [9] proposed the so-called *centered second moment matrix*, defined as follows:

$$\nu(\cdot; \Sigma_a, \Sigma_b) = g(\cdot; \Sigma_b) * \left(\nabla L(\cdot; \Sigma_a) (\nabla L(\cdot; \Sigma_a))^T \right) - (\overline{\nabla L})(\overline{\nabla L})^T$$

where $\overline{\nabla L} = g(\cdot; \Sigma_b) * \nabla L(\cdot; \Sigma_a)$. The centered second moment matrix has the same behavior as the non-centered counterpart, as both behave as in Equation (2) for a linear transformation of the spatial domain. It's also easy to see, that this centering makes shape adaptation invariant with respect to linear changes in image brightness.

IV. Derivation of Motion Characteristics

Ideas presented so far may be used (like a guide) to infer some additional motion characteristics, usually not taken into account in standard optic-flow techniques.

A. Moving Surface Patch

Let us consider a projective camera of focal length l. A point $\mathbf{X} = (X, Y, Z)^T$ in the 3D space is projected onto a point $\mathbf{x} = (x, y)^T = \left(l\frac{X}{Z}, l\frac{Y}{Z}\right)^T$ on the 2D image plane.

As in [5], be Ω a surface patch that moves relatively to the camera, and Ω_t and Ω_{t+1} their projections onto the image plane at times t and $t+1$, respectively; see Figure 3. Let us suppose that Ω can be locally approximated to be incident with the plane $\mathbf{E} : Z = pX + qY + d$ located in the three-dimensional space. Let be A_t and A_{t+1} the area measures of Ω_t and Ω_{t+1} respectively. It follows that

$$A_t = \int_{\Omega_t} \mathrm{d}x \; \mathrm{d}y = \int_{\Omega} \det(J_t) \; \mathrm{d}X \; \mathrm{d}Y \qquad (5)$$

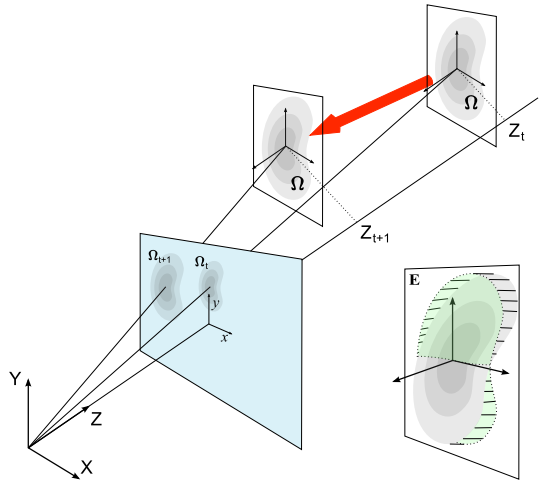


Fig. 3. A moving surface patch Ω, projected into Ω_t.

with J_t the Jacobi matrix of the given transformation at time t. Using the expressions for the imaged point $\mathbf{x}_t$ and the plane $\mathbf{E}$, it follows that

$$\det(J) = l^2 \frac{d}{Z^3} \tag{6}$$

and thus for the area measure (for a fixed focal length l) that

$$A_t = l^2 \int_\Omega \frac{d}{Z^3}\, \mathrm{d}X\, \mathrm{d}Y \tag{7}$$

Let us suppose that the planar patch is located parallel to the image plane or, in a similar way to the case of a *weak-perspective* projection, consider a projection of the surface patch onto a plane which is parallel to the image plane, and incident with the centroid of the patch; see the green region at the bottom right of Figure 3. Let $\tilde{Z}$ be the mean of Z-coordinates of the patch. In that case, $d = \tilde{Z}$ and the relation between the projected area and the Z coordinate of the plane is of the form $A_t \sim 1/\tilde{Z}^2$.

Let λ_z be the ratio between area measures at times t and $t+1$. It follows that

$$\lambda_z = \frac{\sqrt{A_t}}{\sqrt{A_{t+1}}} = \frac{\tilde{Z}_{t+1}}{\tilde{Z}_t} \tag{8}$$

Now consider two subsequent images f_t and f_{t+1} of a recorded sequence, at times t and $t+1$, respectively, and, as before, a surface patch that moves in 3D relatively to the camera between the time slots when frames t and $t+1$ were taken. Assume that this patch is visible in both frames, say with centroids at $\mathbf{x}_t$ and $\mathbf{x}_{t+1}$. For the ratio between the other two coordinate components of a moving point we have that

$$\lambda_x = \frac{X_{t+1}}{X_t} = \frac{Z_{t+1}}{Z_t} \cdot \frac{x_{t+1}}{x_t} = \lambda_z \frac{x_{t+1}}{x_t} \tag{9}$$

$$\lambda_y = \frac{Y_{t+1}}{Y_t} = \frac{Z_{t+1}}{Z_t} \cdot \frac{y_{t+1}}{y_t} = \lambda_z \frac{y_{t+1}}{y_t} \tag{10}$$

In a more compact form, this may be expressed as $\mathbf{X}_{t+1} = \Lambda \mathbf{X}_t$, where $\Lambda = \mathrm{diag}(\lambda_x, \lambda_y, \lambda_z)$, and

$$\Delta\mathbf{X} = \mathbf{X}_{t+1} - \mathbf{X}_t = (\Lambda - \mathbb{I})\mathbf{X}_t \tag{11}$$

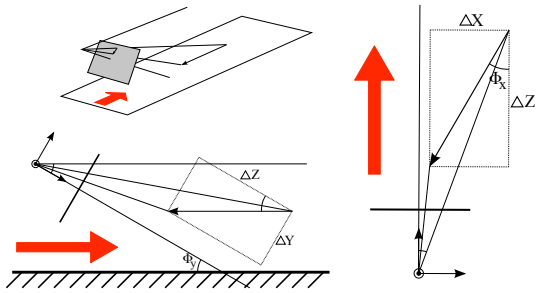


Fig. 4. A tilted camera translating along a plane (top left), motion angles (tilt) on the ZY-plane (bottom left), and on the ZX-plane (yaw; right).

is the absolute spatial displacement, specifying a *scene flow vector*, positioned at $\mathbf{X}_t$.

B. Navigation Angles

Consider a mobile platform (e.g., a car) moving on a planar surface (e.g., a patch of a road surface of limited area), as illustrated in Figure 4. From (11) and letting $\Delta\mathbf{X} = (\Delta X, \Delta Y, \Delta Z)$, it follows that the ratios

$$\frac{\Delta X}{\Delta Z} = \left(\frac{\lambda_x - 1}{\lambda_z - 1}\right) \frac{X_t}{Z_t} = \left(\frac{\lambda_x - 1}{\lambda_z - 1}\right) \frac{x_t}{f} \tag{12}$$

$$\frac{\Delta Y}{\Delta Z} = \left(\frac{\lambda_y - 1}{\lambda_z - 1}\right) \frac{Y_t}{Z_t} = \left(\frac{\lambda_y - 1}{\lambda_z - 1}\right) \frac{y_t}{f} \tag{13}$$

are the tangents of the *navigation angles* Φ_{zx} and Φ_{zy} (see Figure 4). Those angles represent the *3D direction of motion* (between two subsequent frames) for a tracked 3D point.

Now suppose that two shape adaptation matrices were obtained for two corresponding points $\mathbf{x}_t$ and $\mathbf{x}_{t+1}$, in images at times t and $t+1$ (as described in Section II-B for a single point in one image only).[2] Let us call these two matrices μ_t and μ_{t+1}. Recalling Section II-B it follows that

$$\lambda_z = \sqrt{\det(B)} = \sqrt{\frac{\det(\mu_t)}{\det(\mu_{t+1})}}$$

given an estimation of λ_z, the navigation angles, θ_{zx} and θ_{zy}, are given as arcus tangent of the ratios provided by Equations (12) and (13). This estimation is done independently for each of the corresponding features, obtained from frames at times t and $t+1$.

After the estimation of both navigation angles (for all pairs of tracked points), histograms of these two values are computed. This allows to calculate one *summarizing 3D direction*, for all the detected 3D directions between images f_t and f_{t+1}. For this summarizing 3D direction, we decided for

$$\begin{aligned} \hat{\phi}_{zx} &= \arg\max_\theta h(\theta_{zx}) \\ \hat{\phi}_{zy} &= \arg\max_\theta h(\theta_{zy}) \end{aligned}$$

where $h(\theta_{zx})$ and $h(\theta_{zy})$ are the histograms of the navigation angles.

[2]Note that, if the shape adaptation matrix has been obtained with the full iterative algorithm, then it can be used as an initial estimate for the corresponding point, in order to reduce the number of iterations.

To summarize, given a pair of consecutive images, f_t and f_{t+1}, the estimation of the navigation angles, ϕ_{zx} and ϕ_{zy}, is accomplished by the algorithm of Figure 5.

Algorithm 2: Estimation of navigation angles.

1: Extract from image f_t affine blobs, as discussed in Section III.
2: **for** each adapted blob obtained from the previous step **do**
3: Set the size of the tracking window proportional to the characteristic scale of the corresponding blob.
4: Track them from image f_t to f_{t+1} as described in Section IV-C
5: Affine-adapt the tracked point in order to refine their shape/scale estimate.
6: Estimate the λ-ratios, as described in Section IV.
7: Estimate the navigation angles for the current blob, as described in Section IV-B, and update their histograms.
8: **end for**
9: Obtain a final estimate of ϕ_{zx} and ϕ_{zy} by means of the peak on the constructed histograms.

Fig. 5. Algorithm for the estimation of the navigation angles.

The estimation of navigation angles relies on the scale ratio of the tracked blobs; thus, an estimation of such characteristic scales at *sub-scale* level is desirable.

One possibility is then to fit a parabola to three points that determine a local maxima estimated on the scale selection step of the algorithm of Figure 1, and to refine such an extrema at sub-scale resolution. Here, one must take into account that the scale-axis of the scale-space representation is not sampled linearly. Instead, an exponential sampling is used, where the scale of the Gaussian kernel that generates the scale level (n) is obtained from the scale at scale level $(n-1)$ as follows:

$$a_n = ka_{n-1} = k^n a_0 \quad \text{for} \quad n = 1, 2, \ldots, N-1 \qquad (14)$$

Obviously, logarithms of those a values must be used for the computation of an interpolating parabola.

C. Multiscale Tracking

The determination of those navigation angles depends upon the detection of projected motion (of tracked points) in the image plane. A popular approach for feature tracking is the one proposed in [12], for which pyramidal implementations are available at [18]. Here, one parameter to be selected is the size of the tracking window, which is usually treated as a value to be manually adjusted. In the case of blob tracking, this parameter cannot be kept constant for every feature because of the variable size of the image blobs, and, more important, because that image blobs correspond in the general to uniform regions of the brightness pattern; places where the *information content* is low (i.e., textureless regions) leading to *aperture-like* problems.

For the presented model, the relative size of image features are selected automatically. Thus, it is reasonable to set a tracking window proportional to a detected characteristic scale (of the feature to be tracked). This specifies a *multiscale affine blob tracking* method, which proofed to be well suited for this task (and possibly others).

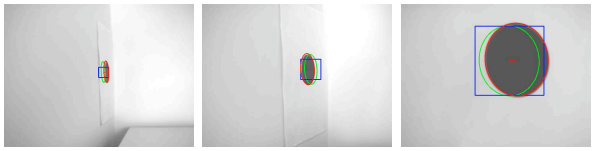


Fig. 6. Detection of an affine blob (black disk), and tracking results for viewpoint and scale changes.

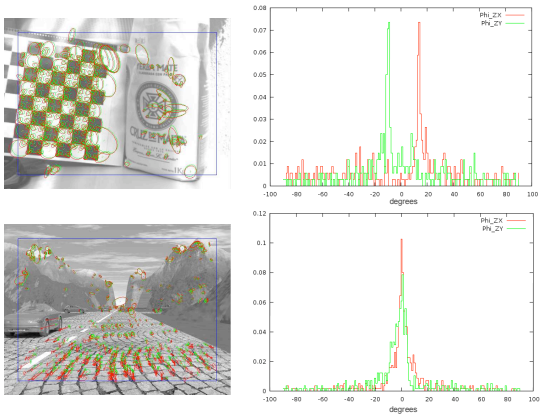


Fig. 7. Images showing tracked blobs and histograms computed for the estimation of angles ϕ_{zx} (red) and ϕ_{zy} (green).

V. Experiments

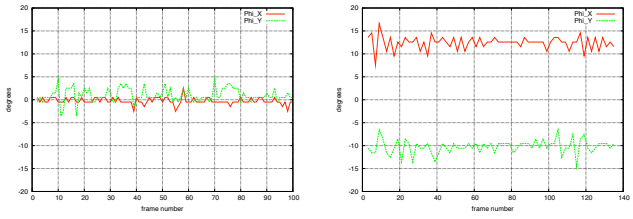
Figure 6 illustrates one particular experiment for affine blob tracking, following the approach as described above. A camera moves freely away from a black disk. In this figure, green and red ellipses indicate shape (scale) estimations at times t and $t+1$, respectively, while the blue box shows the window used for tracking.

Figure 7 shows two generated histograms obtained for the synthetic sequence (Set 2 of [3]) and the desktop sequence.

A square window of size $(2N+1) \times (2N+1)$ was used for tracking, with N set as twice the characteristic scale of the blob detected at time t.

Such estimates at time t can be visualized by diagrams for the whole image sequence. Figure 8 shows such results along the used test sequences. (Note that this allows a new quality of analysis compared to short sequences, such as, for example, currently available on [13].)

The accuracy of estimated values depends in some way on the magnitude of the relative motion. For points that remain



(a) (b)

Fig. 8. Estimation of navigation angles for the (a) synthetic and (b) desktop sequence.

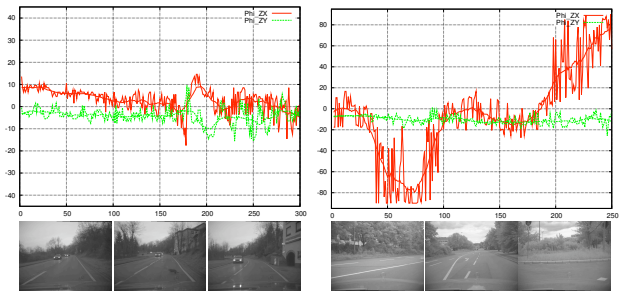


Fig. 9. Results on Sequences 3 (left) and 7 (right) of Set 1 on [3].

almost static, and where the factor λ_z is close to 1, the quotients in Equations (12) and (13) are not well defined, resulting in noisy measurements.

The synthetic sequence was generated with corresponding navigation angles of $\phi_{zy} = 0°$ and $\phi_{zy} = 0°$. The desktop sequence was generated with a calibrated camera, with translational motion on a rail with fixed navigation angles of approximately $\phi_{zx} = 12°$ and $\phi_{zy} = -10°$.

The experiments carried out over the synthetic sequence gave a mean value of $0.13°$ and a standard deviation of $0.57°$ for ϕ_{zx}. For ϕ_Z, those values were $0.15°$ and $0.93°$, respectively.

In the case of the desktop sequence, mean and standard deviation of ϕ_{zx} are equal to $12.42°$ and $1.43°$, respectively. For ϕ_{zy}, those values are equal to $-10.243°$ and $1.52°$, respectively, taken over the entire sequence.

Figure 9 shows the estimation of the navigation angles for two real-world sequences (Daimler sequences 3 and 7, available in Set 1 on [3]). For these sequences, there is no ground truth available for navigation angles, and they are only used as a qualitative (visual) reference. The figure illustrates the instantaneous estimation of the navigation angles and, superimposed, results after applying a sliding mean (5 backward, current, and 5 forward values) filter.[3] It also shows some images of the corresponding sequences.

In the case of Sequence 7 (bottom), the estimated directions Φ_{zx} correspond to the steering of the car over the sequence. In Sequence 3 (top), we observed a low frequency oscillation in the value of Φ_{zy}, starting about at frame 180, when the 'squirrel' (actually, a cat) crossed the street and the car made a breaking maneuver.

The proposed (non-run-time-optimized) algorithm runs at approximately 0.1 fps on a 3.0 GHz Intel© Core 2 Duo CPU.

VI. Conclusions

This work proposes a method for the instantaneous (frame to frame) estimation of the 3D direction of motion; the method was studied based on the determination of scale ratios between tracked blobs. Results may be used for ego-motion correction of video sequences recorded by the ego-vehicle (by compensating for estimated tilt and roll angles).

[3]This sliding mean filter is certainly only a demonstration of filter opportunities; the design of an appropriate Kalman filter would be an option to ensure real-time analysis.

Besides some poor estimations for some frames, the proposed method may be recommended as a possible approach for the use of perceptually very important spatio-temporal cues, induced on images as an observer (camera) moves relatively to the scene. The extracted information has the advantage of being local, allowing robustness in the case of multiple moving objects. The same principle of scale-ratio estimation could also be used for motion segmentation, or to add new constraints to multiple-view approaches of 3D motion estimation, thus further contributing to the already known coherence between optic flow vectors and image disparities.

The overall run-time of the algorithm can be significantly improved by the use of dedicated hardware (FPGA or ASICs).

References

[1] Baumberg, A.: Reliable feature matching across widely separated views. In Proc. *Conf. Computer Vision Pattern Recognition*, volume I, pages 774–781 (2000)

[2] R. Deriche: Recursively implementing the Gaussian and its derivatives. In Proc. *Int. Conf. Image Processing*, pages 236–267 (1992)

[3] *.enpeda..*: Image Sequence Analysis Test Site: www.mi.auckland.ac.nz/EISATS

[4] Garding, J., Lindeberg, T., Direct computation of shape cues using scale-adapted spatial derivative operators. *Intl. J. of Computer Vision*, 17:163–191 (1996)

[5] Klette, R.: Shape from area and centroid, In Proc. *Int. Conf. Artificial Intelligence Information-Control Systems Robots*, pages 309–314 (1995)

[6] Koenderink, J.J.: The structure of images. *Biological Cybernetics*, 50:363–370 (1984)

[7] Lindeberg, T. : *Scale-Space Theory in Computer Vision*. Kluwer Academic Publishers, Norwell, MA, USA (1994)

[8] Lindeberg, T., Garding, J.: Shape-adapted smoothing in estimation of 3-d depth cues from affine distortions of local 2-d brightness structure. In Proc. *European Conf. Computer Vision*, LNCS 800, pages 389–400 (1994)

[9] Lindeberg, T.: A scale selection principle for estimating image deformations. In Proc. *Int. Conf. Computer Vision*, pages 134–141 (1995)

[10] Lindeberg, T.: Feature detection with automatic scale selection. *Int. J. Computer Vision*, 30:77–116 (1998)

[11] Lowe, D.: Distinctive image features from scale-invariant keypoints. *Int. J. Computer Vision*, 60:91–210 (2004)

[12] Lucas, B., Kanade, T.: An iterative image registration technique with an application to stereo vision. In Proc. *IJCAI*, pages 674–679 (1981)

[13] Middlebury Optical Flow website: //vision.middlebury.edu/flow/

[14] Mikolajczyk , K., Schmid, C.: An affine invariant interest point detector. In Proc. *European Conf. Computer Vision*, volume I, pages 128–142 (2002)

[15] Mikolajczyk ,K.: Detection of local features invariant to affine transformations. *PhD thesis* , INRIA (2002)

[16] Mikolajczyk , K., Schmid, C.: Scale and affine invariant interest point detectors. *Int. J. Computer Vision*, 60:63–86 (2004)

[17] K. Mikolajczyk and T. Tuytelaars and C. Schmid and A. Zisserman and J. Matas and F. Schaffalitzky and T. Kadir and L. Van Gool: A comparison of affine region detectors. *Int. J. Computer Vision*, 65(1/2):43–72 (2005)

[18] Open Source Computer Vision Library: www.intel.com/research/mrl/research/opencv/

[19] Spies, H., Haußecker, H., Jähne, B., Barron, J. L.: Differential range flow estimation. In Proc. *DAGM* (1999) 309–316

[20] Tomasi, C. and Kanade, T.: Shape and Motion from Image Streams under Orthography: a Factorization Method, In: *International Journal of Computer Vision*, volume 9 (1992) 137–154

[21] Wedel, A., Vaudrey, T., Rabe, C., Brox, T., Franke, U., Cremers, D.: Decoupling motion and position of image flow with an evaluation approach to scene flow. Chapter XX in this book.

[22] Witkin, A.P.: Scale-space filtering. In Proc. *Int. Joint Conf. Art. Intell.*, pages 1019–1022 (1983).

[23] I. T. Young, L. J. van Vliet, Recursive implementation of the Gaussian filter. *Signal Processing*, 44:139–151 (1995)