Optical Snow Analysis using the 3D-Xray Transform

N. Peled[1] A. Averbuch[1] Y. Keller[2] Y. Shkolnisky[1]
[1]School of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel
[2]Department of Applied Mathematics, Yale University, New Haven 06520-8283, USA

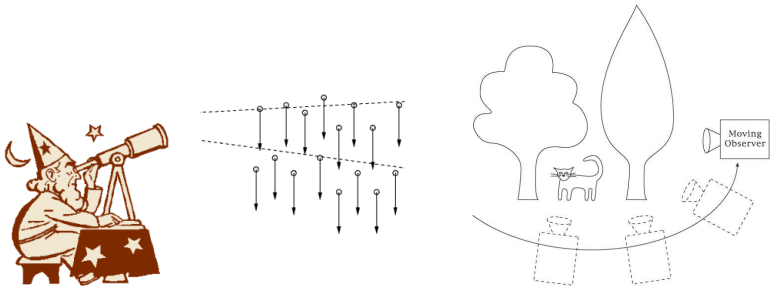
Abstract

There are many methods to analyze motion in computer vision. Most of the classical methods use optical flow, layered motion or segmentation. Optical snow is a complex motion estimation scenario which analyzes motions such as snowfall, tree movements, cars traffic and people walking. These scenes are made of many different objects, moving in different speeds in various directions. While analyzing these scenes, we have to recover both the motion and object segmentation. In this paper we detect *optical snow* motion in video by using the discrete 3D-Xray transform. The algorithm uses the discrete 3D-Xray transform, which is situated as a core algorithm in medical imaging when 3D reconstructions from projections are needed, to partition the captured data and to assemble 3D space into specific planes. The detection of *optical snow* motion with object tracking are achieved through analysis of the energy distributions on these 3D-Xray planes. The output from this analysis contains local and global directions of the movements of these objects. The algorithm identifies and preserves the path of tracked objects even when there are multiple objects, or they are partially covered (occluded) by other objects or there is a compound object of merge and split as players in a soccer game. The algorithm analyzes frame by frame on a video sequence without the need to have neither object segmentation nor clustering. The algorithm is exact and geometrically faithful as it uses summation along straight geometric lines without any interpolation schemes. In addition, it detects the direction of each object and computes its relative velocity. The algorithm has two limitations; the moving objects have to move in straight lines with constant velocity. The algorithm is computationally efficient since the computation utilizes only certain planes without the need to have an exhaustive search and computations of of the activities in all planes.

1 Introduction

The X-ray transform is an important practical tool in many scientific and industrial areas. It plays an important role in computerized tomography (CT). In this work we use the discrete 3D-Xray transform [1] to analyze the properties of *optical snow* motion [2]. *Optical snow* arises in many situations. For example, observer that moves relative to a densely cluttered 3-D scene (see Fig. 1). Snowfall is another typical example. We can assume that snowflakes fall almost in the same direction, although each snowflake has its own direction and velocity. With the Xray transform we can analyze the general direction of snowflakes while detecting the movement of each element in the scene without assuming

neither clustering nor segmentation of the image. This makes it very different from most of the classical common methods.



(a) (b)

Figure 1: Lateral observer. (a) passive observer watches snow fall. (b) The observer moves relative to a clustered 3D scene.

Optical snow is a generalization of optical flow. Most studies of optical flow assume that there is a unique velocity vector at each point in the visual field ([3, 4]). This assumption is only valid for regions in the image whose depth map is continuous. Local spatial continuity of the motion field is not needed for the analysis of *optical snow*. The speed of each snowflake varies inversely with its distance from the camera. The image may contain snowflakes at different multiple depths and the motion field can be spatial. Therefore, the motion field assumed to be discontinuous.

We can visualize the object's movements in time domain to be a line or a tube. It is analyzed by using the discrete 3D-Xray transform. Generally, the Xray transform computes a summation along straight geometric lines. The computation of the Xray transform in time domain enables to track the object in the spatial domain. Each object will have its own tracking and temporary occlusions. This does not degrade the performance of the tracked objects. We assume that the objects move in different directions and have distinct velocities. These are detected in different planes that the 3D-Xray transform generates. This capability is critical when tracking is done in a dense scenery.

The structure of this paper is as follows. In section 2 we explain the behavior of object's motion in the frequency domain. Section 3 describes portion of the theory of the discrete 3D-Xray transform for set of lines in $\mathbb{R}^3$. The *optical snow* theory is introduced in section 4 and we show there how the discrete 3D-Xray transform can help in analyzing the *optical snow* motion. Section 5 describes the algorithms that analyze movements of objects in *optical snow* scene. Experimental results are given in section 6. Section 7 describes image segmentation and detection of motion in a specific direction.

2 How motion is expressed in frequency domain ? - related work

Assume we have a frame that is being translated over time t with velocity (v_x, v_y). Then, we say that each of the 2D spatial frequency components is a shift invariant from a single frame in the frequency

domain. These shifted signals move in the same velocity (v_x, v_y) over the translated frames in time (see [2]). In other words, each object/pixel in the image produces some kind of a wave in a single image in the frequency domain. This wave travels along with the object/pixel movement in the translated sequence of images in the frequency domain. This fact relates the temporal frequency components of a single image to the frequency components in the translated images over time, where the velocity (v_x, v_y) is the connecting component (see [5] for general treatment of this analysis).

Formally, we have $I(x, y, t) = I(x + v_x dt, y + v_y dt, t + dt)$ where $I(x, y, t)$ is the image with velocity $(v_x, v_y) = (dx/dt, dy/dt)$ that is being changed over time t. By taking the derivative of the Fourier transform of the image $I(u)$ we get $\int \frac{\partial I(u)}{\partial u} e^{-2\pi i f u} du = -2\pi i f \int I(u) e^{-2\pi i f u} du$. Using the constraint of the image flow (see [6, 7]), the limit velocity becomes $v_x \frac{\partial I}{\partial x} + v_y \frac{\partial I}{\partial y} + \frac{\partial I}{\partial t} = 0$. If (v_x, v_y) is constant, then by substituting the above equations we get $-2\pi i(v_x f_x + v_y f_y + f_t)\widehat{I}(f_x, f_y, f_t) = 0$ where $\widehat{I}(f_x, f_y, f_t)$ is the Fourier transform of $I(x, y, t)$. Whenever $\widehat{I}(f_x, f_y, f_t) \neq 0$ then $(v_x f_x + v_y f_y + f_t) = 0$, which means that most of the image motion energy lies in the plane $(v_x f_x + v_y f_y + f_t)$.

Images with translated motion were investigated in many works. Most of them use the fact that the image motion lies in the plane $(v_x f_x + v_y f_y + f_t)$ (see [2]). The goal is to efficiently find and compute the activities in these planes. Our algorithm also uses this property, but it has additional advantages. Here, we focus on the problem of analyzing motion on *optical snow* scenes. *Optical snow* is a generalization of optical flow in which the assumption of local spatial continuity of the motion field is abandoned. The pioneering work on optical snow analysis using Fourier methods was first suggested in [2]. Optical flow techniques - such as differential methods [6, 8, 9, 10], region-based matching [11, 12, 13],or energy based methods [14] - relay on local, raw estimates of the optical flow field to produce a partition of the image. The algorithm presented here finds and computes the planes via the application of the 3D-Xray transform. Past approaches use Markov Random Fields (MRF) to handle discontinuities in optical flow [15, 16, 17, 18, 19]. While these methods produce good results, they rely heavily on a proper spatial segmentation early in the algorithm, which is not realistic in many cases. Our algorithm does not need pre-segmentation. The 3D-Xray transform enables to analyze the movements of each object in the image even if global computation is used and not a clustered (local) based computation. We can investigate the motion of each object by looking at the corresponding plane generated by the 3D-Xray transform. For example, we can find all the objects that move in a specific direction by identifying the plane that represents this direction. The objects in the spatial domain are retrieved from the application of the inverse Fourier transform.

Many approaches to motion analysis are formulated as model-based tracking. These works, especially where 3D tracking is involved, include a prior knowledge about the investigated objects [20] or manual initialization [21, 22, 23, 24]. In the presented algorithm no prior knowledge of the scene is needed, although some specifications, as the motion in specific direction can help to speedup the computation by eliminating certain 3D-Xray planes from the computation.

The proposed scheme can also be used to analyze layered motions. Layered models mostly represent the motion of a small number of opaque objects and assume constant velocity field in each layer

[25, 26, 27, 28, 29, 30]. The layers are assumed to be continuous in the spatial domain and its number is usually less than 10. Thus, the Optical snow is a more difficult problem, as spatial continuity is not assumed and the number of objects can be large. Analysis of motion using the 3D-Xray transform requires constant motion velocity as well, but the discrete 3D-Xray transform can efficiently compute the velocities of multi-objects. We can compute the velocity of each object by inspecting the proper plane. Other methods ([2]) can compute the average velocity only. By expanding the use of the proposed algorithm we can even say whether the object accelerates or decelerates.

Finally, the discrete 3D-Xray transform provides an accurate computation of the energy in each plane. The transform is based on the application of the 1-D FFT. This accuracy is important since the Fourier transform becomes numerically instable if interpolation, which is used in all other methods, is involved. The Fourier transform is very sensitive to errors that come from interpolated values.

In summary, the 3D-Xray based algorithm enables to investigate and analyze multiple objects in different scenarios when *optical snow* motion is present.

3 3D-Xray theory: background

The proposed algorithm is based on the application of the discrete 3D-Xray transform. The mathematical background of the transform is based on [1]. In this section we briefly outline the theory and the main principals of the 3D-Xray transform that is needed here.

3.1 The continuous X-ray transform

The continuous X-ray transform of a 3D function $f(x, y, z)$, denoted by $\mathcal{P}f$, is defined by the set of all line integrals of f. For a line L, defined by a unit vector θ and a point x on L, we express L as $L(t) = x + t\theta \qquad t \in \mathbb{R}$. The X-ray transform of f on L is defined as

$$\mathcal{P}_\theta f(x) \triangleq \mathcal{P}f(L) \triangleq \int_{-\infty}^{\infty} f(x + t\theta)dt. \tag{3.1}$$

The X-ray transform maps each line L in $\mathbb{R}^3$ to a real value that represents the projection of the function f along the line L. The X-ray transform is closely related to the Radon transform. However, while the 3D X-ray transform is defined using line integrals of the function f, the 3D Radon transform is defined as integrals of f over all planes in $\mathbb{R}^3$. Note that in the 2D case, the X-ray transform coincides with the Radon transform. See [31, 32] for more information about the continuous Radon transform.

The Fourier slice theorem connects the continuous X-ray transform, defined by Eq. 3.1, with the Fourier transform. For a given 3D function f, it defines the relation between the 2D Fourier transform of the X-ray transform of f and the 3D Fourier transform of f. The Fourier slice theorem is summarized in the following theorem.

Theorem 3.1. *For a function $f(x, y, z)$ and a family of lines in $\mathbb{R}^3$, whose direction is given by the unit vector θ, we have $\widehat{\mathcal{P}_\theta f}(\xi) = \widehat{f}(\xi)$ where $\xi \in \theta^\perp$ and $\theta^\perp$ is the subspace perpendicular to θ.*

The discretization guidelines are given in [31, 32].

3.2 Semi-discrete transform definition

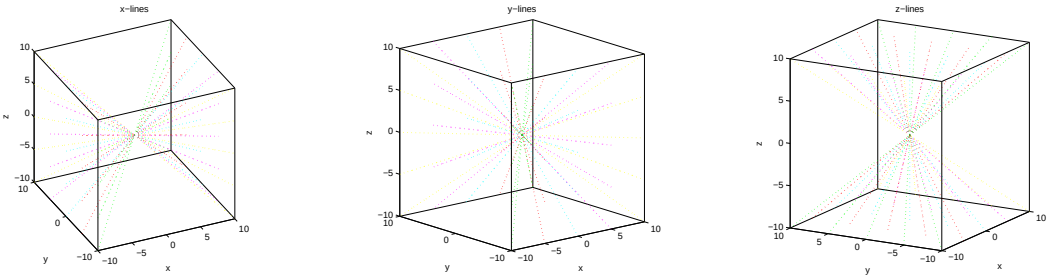
A line in $\mathbb{R}^3$ is paramterized as the intersection of two planes. Using this parametrization we define three families of lines, which we call *x-lines*, *y-lines* and *z-lines*. For convenient explanation we define our source as $n \times n \times n$ cube, but later on we will use a source with different dimensions. Formally, a x-line, y-line and z-line are defined as

$$l_x(\alpha, \beta, c_1, c_2) = \begin{cases} y = \alpha x + c_1 \\ z = \beta x + c_2 \end{cases} \quad |\alpha| \le 1,\ |\beta| \le 1, \quad c_1, c_2 \in \{-n, \dots, n\}. \tag{3.2}$$

$$l_y(\alpha, \beta, c_1, c_2) = \begin{cases} x = \alpha y + c_1 \\ z = \beta y + c_2 \end{cases} \quad |\alpha| \le 1,\ |\beta| \le 1, \quad c_1, c_2 \in \{-n, \dots, n\}. \tag{3.3}$$

$$l_z(\alpha, \beta, c_1, c_2) = \begin{cases} x = \alpha z + c_1 \\ y = \beta z + c_2 \end{cases} \quad |\alpha| \le 1,\ |\beta| \le 1, \quad c_1, c_2 \in \{-n, \dots, n\}. \tag{3.4}$$

We denote the sets of all x-lines, y-lines and z-lines in $\mathbb{R}^3$ by $\mathcal{L}_x$, $\mathcal{L}_y$ and $\mathcal{L}_z$, respectively. Also, we denote the family of lines that corresponds to a fixed direction (α, β) and variable intercepts (c_1, c_2), by $l_x(\alpha, \beta)$, $l_y(\alpha, \beta)$ and $l_z(\alpha, \beta)$ for a family of x-lines, y-lines and z-lines, respectively. See Fig. 2 for an illustration of the different families of lines for $c_1 = c_2 = 0$.



(a) (b) (c)

Figure 2: The line families $\mathcal{L}_x$, $\mathcal{L}_y$ and $\mathcal{L}_z$

It is easy to see that each line in $\mathbb{R}^3$ can be expressed as either a x-line, y-line, or z-line. In other words, each line in $\mathbb{R}^3$ belongs to $\mathcal{L}_x$, $\mathcal{L}_y$ or $\mathcal{L}_z$. The sets $\mathcal{L}_x$, $\mathcal{L}_y$ and $\mathcal{L}_z$ are not disjoint.

Generally speaking, for a given image I and a line l, we define the discrete X-ray transform of the image I for the line l as the sum of the samples of I along l. For a discrete image I of size $n \times n \times n$ we

define three continuous extensions of I, which we denote by I_x, I_y and I_z. Each of the extensions I_x, I_y and I_z is a continuous function in directions perpendicular to its index. This means, for example, that I_x is a continuous function in the y and z directions. Formally, we define these three extensions by

$$I_x(u,y,z) = \sum_{v=-n/2}^{n/2-1} \sum_{w=-n/2}^{n/2-1} I(u,v,w) D_m(y-v) D_m(z-w) \tag{3.5}$$
$$u \in \{-n/2, \ldots, n/2-1\}, \quad y,z \in \mathbb{R},$$

$$I_y(x,v,z) = \sum_{u=-n/2}^{n/2-1} \sum_{w=-n/2}^{n/2-1} I(u,v,w) D_m(x-u) D_m(z-w) \tag{3.6}$$
$$v \in \{-n/2, \ldots, n/2-1\}, \quad x,z \in \mathbb{R},$$

$$I_z(x,y,w) = \sum_{u=-n/2}^{n/2-1} \sum_{v=-n/2}^{n/2-1} I(u,v,w) D_m(x-u) D_m(y-v) \tag{3.7}$$
$$w \in \{-n/2, \ldots, n/2-1\}, \quad x,y \in \mathbb{R}$$

where $D_m(t) = \frac{\sin \pi t}{m \sin(\pi t/m)}$ is the Dirichlet kernel of length $m = 2n+1$. It performs trigonometric interpolation of length m. Therefore, the continuous functions I_x, I_y and I_z are obtained from the discrete image I by zero padding the relevant directions of I to length m and then performing trigonometric interpolation.

Next, we use I_x, I_y and I_z to define the discrete X-ray transform. For a x-line $l_x(\alpha,\beta,c_1,c_2) \in \mathcal{L}_x$, given by Eq. 3.2, we define the discrete X-ray transform $P_x I(\alpha,\beta,c_1,c_2)$ as

$$P_x I(\alpha,\beta,c_1,c_2) = \sum_{u=-n/2}^{n/2-1} I_x(u, \alpha u + c_1, \beta u + c_2), \quad |\alpha| \le 1, \quad |\beta| \le 1, \quad c_1,c_2 \in \{-n, \ldots, n\} \tag{3.8}$$

where I_x is given by Eq. 3.5. The transformation $P_x : \mathcal{L}_x \to \mathbb{R}$ is obtained by traversing the line l_x with unit steps in the x direction, and for each integer u the value of the image I at the point $(u, \alpha u + c_1, \beta u + c_2)$ is summed.

Similarly to Eq. 3.8, we define the discrete X-ray transform $P_y I(\alpha,\beta,c_1,c_2)$ for the y-line $l_y(\alpha,\beta,c_1,c_2) \in \mathcal{L}_y$ as

$$P_y I(\alpha,\beta,c_1,c_2) = \sum_{v=-n/2}^{n/2-1} I_y(\alpha v + c_1, v, \beta v + c_2) \tag{3.9}$$

where I_y is given by Eq. 3.6. Finally, we define the discrete X-ray transform $P_z I(\alpha,\beta,c_1,c_2)$ for a z-line $l_z(\alpha,\beta,c_1,c_2) \in \mathcal{L}_z$ as

$$P_z I(\alpha,\beta,c_1,c_2) = \sum_{w=-n/2}^{n/2-1} I_z(\alpha w + c_1, \beta w + c_2, w) \tag{3.10}$$

where I_z is given by Eq. 3.7.

The parameters c_1, c_2 and m in Eqs. 3.2 and 3.5 are carefully chosen to ensure that the transform, defined by Eq. 3.8, is invertible and geometrically faithful. We choose $m \geq 2n+1$ as the length of the interpolation kernel D_m to avoid line wraparound due to the periodic nature of the Dirichlet kernel. See [32] for a rigorous derivation of the length of the shortest valid kernel.

Equations 3.8 – 3.10 define the X-ray transform for x-lines, y-lines and z-lines, respectively. Since each line in $\mathbb{R}^3$ can be expressed as either a x-line, y-line or z-line, then, given a line l, we express it as either a x-line, y-line or z-line and apply on it our definition of the X-ray transform.

In section 3.4 we show how to discretize the set (α, β) to have a fully discrete definition of the X-ray transform, which is rapidly computable. In the *"optical-snow"* algorithm we will use only the computation of *z-lines*, where z is the time axis.

3.3 Discrete Fourier slice theorem for the discrete X-ray transform

As we showed in theorem 3.1, the continuous X-ray transform satisfies the Fourier slice theorem, which associates the continuous X-ray transform of a function f with the Fourier transform of f. This relation is very useful for both the computation and analysis of the continuous X-ray transform. We will therefore derive a similar relation for the discrete X-ray transform.

Theorem 3.2 (Fourier slice theorem for the x-lines ([1])). *For a given family of x-lines $l_x(\alpha, \beta)$ with fixed slopes (α, β) and variable intercepts (c_1, c_2), we take the 2D array of projections $P_{x(\alpha,\beta)}I$, where $P_{x(\alpha,\beta)}$ is the x-lines in angles α and β of the directional vector. Then,*

$$\widehat{P}_{x(\alpha,\beta)}I(k,l) = \widehat{I}(-\alpha k - \beta l, k, l), \quad k,l = -n, \ldots, n \tag{3.11}$$

where $\widehat{I}$ is given by

$$\widehat{I}(\xi_1,\xi_2,\xi_3) = \sum_{u=-n/2}^{n/2-1} \sum_{v=-n/2}^{n/2-1} \sum_{w=-n/2}^{n/2-1} I(u,v,w) e^{-2\pi\imath\xi_1 u/m} e^{-2\pi\imath\xi_2 v/m} e^{-2\pi\imath\xi_3 w/m}. \tag{3.12}$$

and $\widehat{P}_{x(\alpha,\beta)}I(k,l)$ is the 2D DFT of the array $P_{x(\alpha,\beta)}I$.

Similar theorems hold for the y-lines and z-lines.

Geometrically, theorem 3.2 states that the 2D DFT of the discrete X-ray transform over a family of x-lines with fixed slopes (α, β) is equal to the samples of the trigonometric polynomial $\widehat{I}$ on the plane defined by the points $(-\alpha k - \beta l, k, l)$. Explicitly, we need to sample $\widehat{I}$ on the plane given by the equation $x = -\alpha y - \beta z$. Figure 3 depicts these x-planes for various values of α and β. Theorem 3.2 is very important for the rapid computation of the discrete X-ray transform, as it relates the discrete X-ray transform of the image I to the 3D DFT of I.

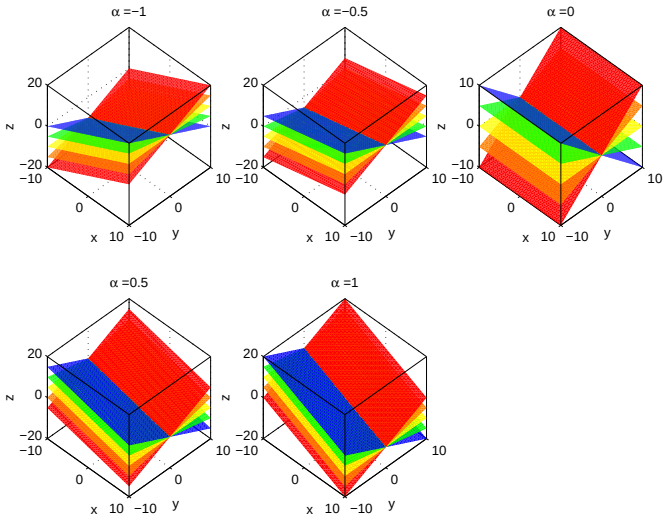


Figure 3: x-planes for various values of α and β

3.4 Discretization of the X-ray transform

So far the X-ray transform is defined over the continuous line sets $\mathcal{L}_x$, $\mathcal{L}_y$ and $\mathcal{L}_z$. These line sets are comprised from lines that have discrete intercepts and continuous slopes. In this section we define the discrete X-ray transform for a discrete set of lines, which are discrete both in their slopes and intercepts. Consider the set $S = \{2p/n\}, \quad p = -n/2, \ldots, n/2$. We define the *discrete X-ray transform* as a restriction to the set of slopes $S \times S$. Formally, we define three discrete sets of lines:

- discrete x-lines
$$\mathcal{L}_x^d = \{l_x(\alpha, \beta, c_1, c_2) \in \mathcal{L}_x \mid \alpha \in S,\ \beta \in S\} \tag{3.13}$$
- discrete y-lines
$$\mathcal{L}_y^d = \{l_y(\alpha, \beta, c_1, c_2) \in \mathcal{L}_y \mid \alpha \in S,\ \beta \in S\} \tag{3.14}$$
- discrete z-lines
$$\mathcal{L}_z^d = \{l_z(\alpha, \beta, c_1, c_2) \in \mathcal{L}_z \mid \alpha \in S,\ \beta \in S\}. \tag{3.15}$$

Let $\mathcal{L}^d \triangleq \mathcal{L}_x^d \cup \mathcal{L}_y^d \cup \mathcal{L}_z^d$. By using the lines in $\mathcal{L}^d$ we define the discrete X-ray transform for discrete images as

Definition 3.1. For an image I and a line $l(\alpha, \beta, c_1, c_2) \in \mathcal{L}^d$ the discrete X-ray transform is given by

$$PI(l) = \begin{cases} P_x I(l) & l \in \mathcal{L}_x^d \\ P_y I(l) & l \in \mathcal{L}_y^d \\ P_z I(l) & l \in \mathcal{L}_z^d \end{cases} \tag{3.16}$$

where P_x, P_y and P_z are defined by Eqs. 3.8 – 3.10, respectively.

Definition 3.1 defines the discrete X-ray transform for discrete images by using a discrete set of lines. This transform is not defined for all lines in $\mathbb{R}^3$, but only for lines in $\mathcal{L}^d$. We will show in section 5 our algorithm for computing the set of lines $\mathcal{L}^d_z$ to analyze the *optical snow* phenomena.

Since the Fourier slice theorem 3.2 holds for continuous slopes (α, β), it holds in particular for the discrete set of slopes defined by $S = \{2p/n\}, \quad p = -n/2, \ldots, n/2$. Substituting the discrete set of slopes, $S = \{2p/n\}, \quad p = -n/2, \ldots, n/2$, into theorem 3.2 gives the discrete Fourier slice theorem, which is defined for both discrete images and a discrete set of directions.

Corollary 3.1. *([1])[Discrete Fourier slice theorem for the z-lines]*
Let $S = \{2p/n\}, \quad p = -n/2, \ldots, n/2$ *and let* $\widehat{I}$ *be the trigonometric polynomial defined in Eq. 3.12. Then, for a given family of z-lines* $l_z(\alpha, \beta)$, $\alpha = 2p/n \in S_1$, $\beta = 2q/n \in S_2$

$$\widehat{P}_{z(2p/n,2q/n)} I(k,l) = \widehat{I}(k,l,-2pk/n - 2ql/n). \tag{3.17}$$

4 Motion analysis in optical snow

As illustrated in Fig. 1, *optical snow* can be viewed by either an observer that moves laterally in a static 3D clustered scene or by a static observer that watches the same scene that propagates in a uniform direction. Each object in the observed scene can move in its own direction (d_x, d_y) with velocity (v_x, v_y) where the general direction stays uniform. For example, each snowflake in a snowfall scene has its own direction and its own velocity in the spatial domain, but the majority of their movements are towards earth due to gravity. When all the velocities are in the same direction, it is called *parallel optical snow*.

4.1 Object's Motion

Assume an object moves in front of a static camera, and let $d = (d_x, d_y, d_z)$ be the distance the object moves in each frame. The distance d can vary from frame to frame due to object's velocity and direction. Assume that generally the object moves in one direction with constant velocity. In other words, the distance vector stays constant. Under this assumption, this movement generates a straight line/tube l in the time axis (see Fig. 4). The 3D-Xray transform analyzes these lines.

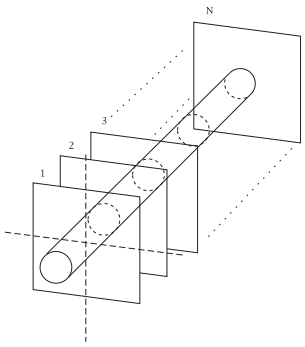


Figure 4: The motion of an object. It generates a "tube" (trace) like view in the 3D image.

To simplify the explanation we use a confined graphic case (images in Fig. 5). Let P be our object, which we treat as a single point $P = (x, y)$ in a 2D plane. It moves vertically along the Y-axis. We consider the Z-axis as the time axis. Therefore, the coordinates of the object's movements are $\{(0,0,0),(0,1,1),(0,2,2),.....,(0,n,n)\}$ which it is illustrated in the leftmost image in Fig. 5.

Formally, a $z-line$ in the 3D volume is defined (see Eq. 3.4) as

$$l_z(\alpha,\beta,c_1,c_2) = \begin{cases} x &= \alpha z + c_1 \\ y &= \beta z + c_2 \end{cases} . \tag{4.1}$$

In our case, we can write the equation as $l_z(0,\beta,0,0) = y = \beta z + 0 = \beta z$ where β is the angle between the movement of vector V and the time axis Z. β is the relative velocity of the object. Fast object generates a bigger β angle, while slow object generates a smaller β angle. No movement generates a vector V, which is parallel to the Z-axis (see the right image in Fig. 5). By knowing the range of speeds that we measure together with our sampling rate (frame rate) we can determine the speed of the object.

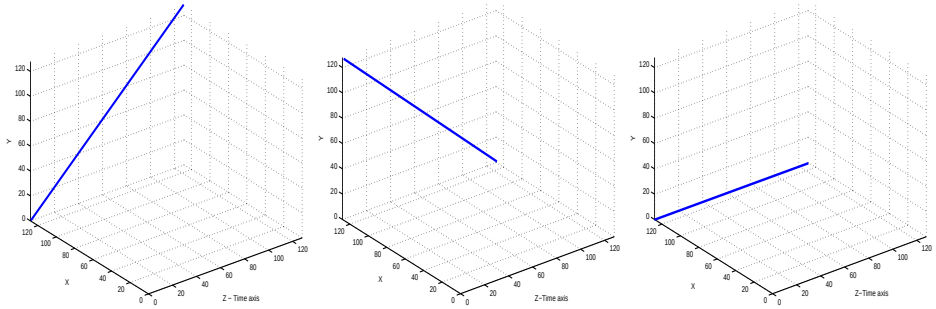


Figure 5: Left: The object's motion vector from $(0,0,0)$ to $(0,n,n)$ in the 3D-image is marked by blue. Middle: The object's motion vector from $(0,n,0)$ to $(0,0,n)$ in the 3D-image is marked by blue. Right: the static object creates a line/tube which is parallel to the Z-axis.

In addition, we can measure the movement's direction using this vector. The direction of the vector determines the direction of the movement. For example, if we take the same object that moves in the

opposite direction from $(0, n, 0)$ to $(0, 0, n)$, we get a vector that points down instead of pointing up in the middle image in Fig. 5. In the same way, we can expand these cases to all movements in 3D.

4.2 Movements analysis of multiple objects

In this section we expand the case that handled a single object's motion as was explained in section 4.1. We show that the algorithm can be extended to handle various scenarios when multiple objects are present. We assume the following setup: Each frame is of size $n \times n$ that contains all the objects. We track the movements of the objects in m frames. This assumption creates a box of size $n \times n \times m$, which is considered to be the input. The coordinates of the movements are X, Y and Z, which are the horizontal, vertical and time axes of the image, respectively.

As we saw in section 4.1, in the time axis Z we get a line or tube that describes the object's movements. Each static object generates a tube, which is parallel to Z. Dynamic object generates a tube which is not parallel to Z. Constant velocity is a prerequisite for the coming analysis. Therefore, we analyze short time movements in which we assume that the velocity is constant.

As noted in the introduction, an object whose velocity is given by the vector (v_x, v_y) induces positive energy on the plane $v_x f_x + v_y f_y + t_t = 0$ in the Fourier domain. The energy is a function of the number of objects with this velocity vector. Thus, if there exists dominant motion in the scene with common velocity vector (v_x, v_y), then, most of the energy in the Fourier domain lies on the plane $v_x f_x + v_y f_y + t_t = 0$. This suggests the following algorithm: Compute the Fourier transform of the scene; Find the plane on which most the energy lies; (v_x, v_y) is given by the normal to this plane.

Since we investigate an *optical snow* scene, we can expect that the general movements of the object will be only in the X, Y plane. Movement in the Z axis (the depth) is assumed to be small in comparison to the movements on X, Y, therefore they are ignored. Formally, if we denote the movement as (T_x, T_y, T_z), then we can assume that $T_z \ll (T_x, T_y)$. This assumption allows us to easily find the general direction of all objects in the scene. This global direction is detected on the (f_x,f_y) plane, which means that the vector's direction is (x,y,0). By finding specific α and β (Eq. 3.17), which generate the surface with maximum l_1 sum, we can determine the angle of this surface in the 3D domain. α and β are actually the pan and tilt of the surface with the maximum energy (see Fig. 3) and the vector of the global movement's direction will be perpendicular to this surface.

The computation is done as follows. If the surface with the maximum energy, which is defined by Eq. 3.11 to be $(k, l, -\alpha k - \beta l)$ for a given family of $z - line$, then we get:

$$x = k, \quad y = l, \quad z = -\alpha k - \beta l. \tag{4.2}$$

The perpendicular vector to this surface is computed in the following way: From Eq. 4.2 we have $z + \alpha k + \beta l = 0$. Then, $\alpha x + \beta y + z = 0$, which can be rewritten as $(\alpha, \beta, 1)(x, y, z) = 0$, which means that the requested perpendicular vector is $(\alpha, \beta, 1)$.

5 Determination of the movement direction in optical snow: algorithmic description

We show how to compute the movement's direction in an *optical snow* scene. This algorithm has four major steps:

Step 1: Input generation. The output is denoted by $3D\text{-}inputIm$ (section 5.1).

Step 2: Application of the 3D-Xray transform in the Z direction on the $3D\text{-}inputIm$ from step 1. The output is denoted by $3D\text{-}XrayImage_z$ (section 5.2).

Step 3: The $3D\text{-}XrayImage_z$ surfaces from step 2 are scanned. The output that is denoted by *Energy-map* (section 5.3).

Step 4: Finding the direction of the movement from the *Energy-map* in step 3. The output is an *angle* (section 5.4).

Here is a detailed algorithmic description of each step.

5.1 Step 1: Input construction

The input is a 3D video file which captures the *optical snow* scene. m frames from this video file are processed. To simplify the presentation we assume that each frame is of size $n \times n$ pixels. We construct a box of dimensions $n \times n \times m$ that corresponds to the X, Y, Z axes, respectively. The input is a sequence of video frames $f_1, f_2, \ldots, f_m$. It outputs $I \triangleq 3D - inputIm$.

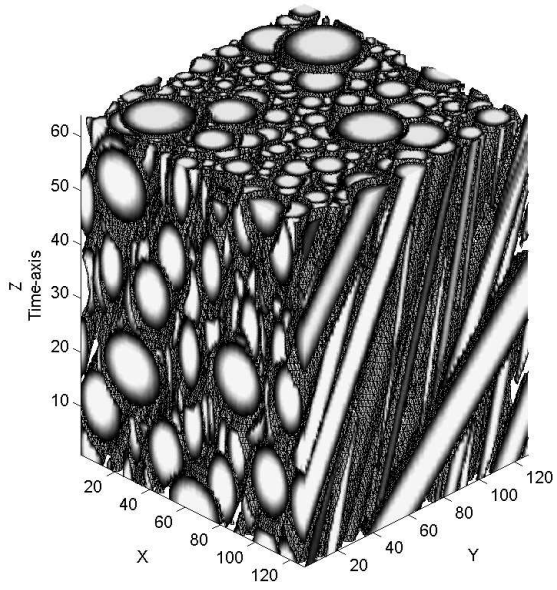


Figure 6: Construction of the input to Step 2. This $3D\text{-}inputIm$ is constructed from a sequence of 64 frames that contain snow-fall scenes. The images are placed one above the other and each object (snow flake) creates a "tube" like form in the Z-axis

5.2 Step 2: Application of the 3D-Xray transform using equation 3.16

We apply the 3D-Xray transform ([1]) from section 3 (Eq. 3.16) on $3D\text{-}inputIm$ that was constructed in Step 1. We compute the 3D-Xray transform only in the $z - direction$. To speed the computation we introduce the *skip* parameter to reduce the number of computed slopes. *skip* can take any value between 0 to n. For example, in our case, where the input dimensions are $n \times n \times m$, we get $n + 1$ slopes for α and $n + 1$ slopes for β in the regular case. By introducing *skip*, the number of slopes $(n/skip)+1$ is smaller and the computation is reduced.

$O(n^7)$ operations are required to compute the Xray transform in a straightforward way. As shown in [1], we can compute the transform in $O(n^4 \log n)$ operations using a fast algorithm. The pseudo-code given in this section implements the fast 3D-Xray transform according to Eq. 3.17 in the Fourier slice theorem.

Algorithm 1: Application of the Xray transform (Eq. 3.17)

input: $(I, skip)$

output: The output is stored in ***3D-XrayImage***$_z$, a 4-D matrix of size $(n+1)\times(n+1)\times d\times d$, $(d=2n+1)$, which contains the computed 3D-Xray transform

/**********

$n = n/skip$

E_x, E_y – Extension operators, which symmetrically zero-pad the image I to length $2n+1$ along the x, and y directions, respectively.

F_x, F_y – 1D discrete Fourier transform (FFT) along the specified direction. For example, F_xI takes all the vectors $I(\cdot, y, z)$ and applies on them the 1D FFT.

$CZT(z, \omega_0, \Delta\omega)$ - The chirp Z-transform with parameters ω_0 and $\Delta\omega$. Specifically, $CZT(z, \omega_0, \Delta\omega)$ is defined by

$$CZT(z,\omega_0,\Delta\omega)_k = \sum_{j=-n/2}^{n/2-1} z(j)e^{-2\pi\imath j\omega_k}$$

$$\omega_k = \omega_0 + k\Delta\omega, \quad k = -n/2,\ldots,n/2.$$

**********/

begin

$\dot{I} = E_xE_yI$

$\widetilde{I} = F_xF_y\dot{I}$

foreach p ***in*** $-n/2$ ***to*** $n/2$ ***step*** *skip* **do**

foreach k,l ***in*** $-n$ ***to*** n **do**

$z_{k,l} \leftarrow \widetilde{I}(\cdot,k,l)$ ($z_{k,l}$ is a sequence of length n)

$\omega_0 \leftarrow -2pk/(n\cdot d)$

$\Delta\omega \leftarrow -2l/(n\cdot d)$

3D-XrayImage$_z(p,\cdot,k,l) = CZT(z_{k,l},\omega_0,\Delta\omega)$

end

end

return *3D-XrayImage*$_z$;

end

5.3 Step 3: Construction and computation of the *Energy-map*

The *Energy-map* is a 2D matrix of size $(n+1)\times(n+1)$ where each entry is denoted by $Energy{-}map(i,j)$ that corresponds to the energy of the surface i,j in the 3D-Xray image. The input was generated in Step 2. The output is: $Energy-map(i,j) = \sum_{k,l=0}^{2n+1} 3D\text{-}XrayImage_z^2(i,j,k,l)$ and the total energy is $\sum_{i,j=0}^{n+1} Energy-map(i,j)$.

5.4 Step 4: Identification of the direction

From the *Energy-map* we identify the surface that has the maximum energy and compute the direction of the movement. The output of this function is an angle that is ranged between 0° to 360°, where 0° is the $X - axis$ (see Fig. 7).

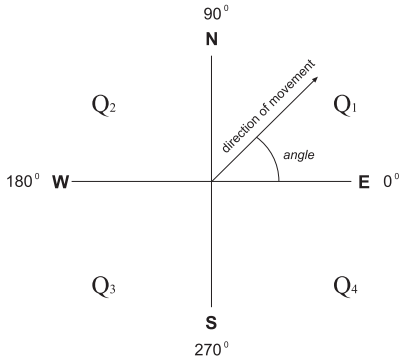


Figure 7: The output from the application of Steps 1-4 is the *angle*.

The angle is computed from the input (x, y) using Fig. 7. The output is the *angle* between the direction vector and the X-axis for a given (x, y).

6 Experimental results

From the assumption $T_z \ll (T_x, T_y)$ we know that the general direction of the movement should lie in the (f_x, f_y) plane, where f_x and f_y correspond to the horizontal and vertical movements, respectively. This enables to analyze the general object's movement as we see in the following examples.

6.1 Example 1: Snow fall

We used here a synthetic image that contains a falling spheres as shown in Fig. 8 (substitute for snow flakes). This video sequence resembles the video sequence in [2].

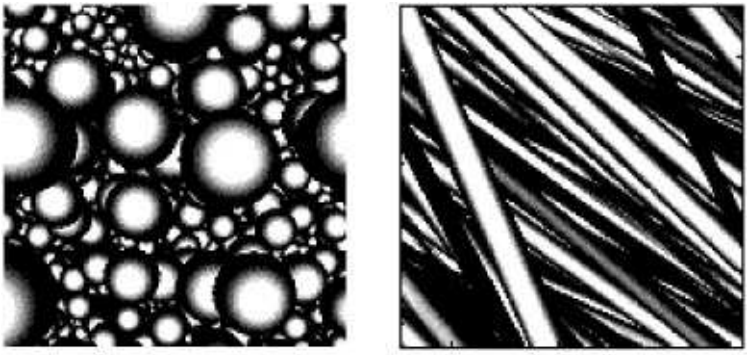


Figure 8: Synthetic illustration of falling spheres. Left: First frame in the sequence - XY slice. Right: YZ slice of the sequence taken from the middle of the box. The spheres have different velocities. The size of the box is $128 \times 128 \times 128$.

We now describe in detail the sequential application of steps 1-4 from section 5 on this example.

6.1.1 Step 1: Construction of the input

The spheres in Fig. 8, are in different distances from the camera. Therefore, each sphere has a different size. The spheres are moving downward. The algorithms were tested on several different time periods. It was tested on 10, 20 and 30 frames. Therefore, the input box is of size $128 \times 128 \times 10$, $128 \times 128 \times 20$ and $128 \times 128 \times 30$, respectively. Figure 8 illustrates the XY and the YZ slices of the input sequence that has different velocities. We can detect the general direction of the spheres by looking at the YZ slice in Fig. 8 (right). Each object creates a line along the time axis as was explained in sections 4.1 and 4.2.

6.1.2 Step 2: Computation of the Xray transform

The Xray transform is applied on the $z-lines$ only. The sought after data lies on the $z-line$, denoted by $\mathcal{L}_z^d$ (Eq. 3.15).

The algorithm was tested with $skip = 2$, which means that we skip the second value of each of the α and β slopes. This parameter speeds up the running time and memory usage. It also reduces the accuracy by factor 2. In our case, when the input box is of size 128×128 and the $skip = 2$, only 65 α and β slopes are computed. Therefore, the achieved accuracy in the X and Y directions is $90°/65 \approx 1.38°$ in contrast to the maximum expected accuracy of $90°/129 \approx 0.69°$ with $skip = 1$.

6.1.3 Step 3: Construction of *Energy-map*

Figure 9 shows all the lines when $skip = 1$ is used. The solid lines represent the ignored slopes when $skip = 2$ is used. From the constructed $3D\text{-}XrayImage_z$ (Step 2), we compute the energy of each surface that is represented by a slope. Thus, we have $\lceil \frac{129}{2} \rceil \times \lceil \frac{129}{2} \rceil = 4225$ surfaces of size $(2 \times 128 + 1) \times (2 \times 128 + 1) = 257 \times 257$. The energy of each surface is computed and these energy

values are summed. Then, each surface from the 4225 available surfaces is associated with an *energy value* from the *Energy-map*. These values are stored in a 65×65 matrix, where the X axis represents the α slopes and the Y axis represents the β slopes (see Eq. 3.11). This matrix is called *"Energy-map"*.

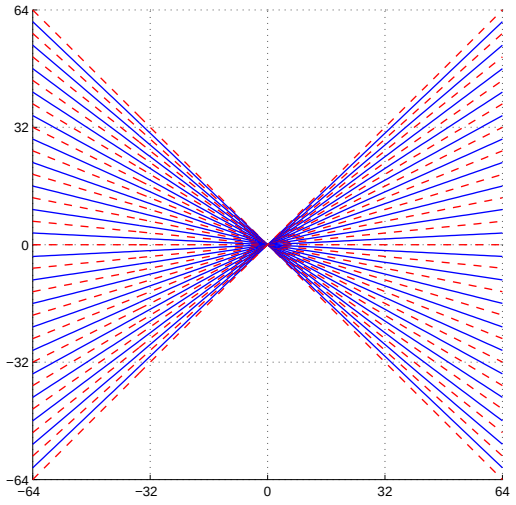


Figure 9: 129 slopes with $skip = 1$. The solid lines represent the skipped slopes when $skip = 2$.

Figure 10 shows the *Energy-map* distribution. We see that higher energy values are detected along the route of the object's movements.

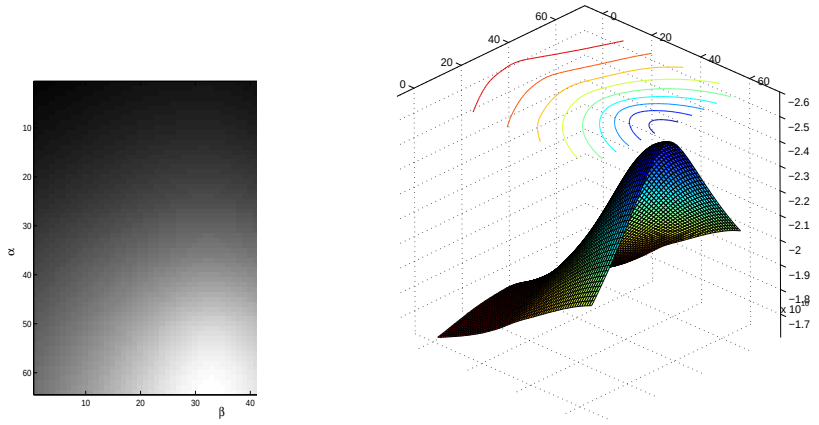


Figure 10: Distribution of the *Energy-map* of Fig. 8 as 2D (Left) and 3D (Right) where the axes are α and β. Higher energy values (colored blue) are located around the object's route.

6.1.4 Step 4: Computation of the direction of the movements

We are looking for the highest value in the *Energy-map*. The values $\alpha = \beta = 0$, which represent static objects in the sequence (see the rightmost image in Fig. 5), are ignored. The final result is denoted by *angle*. For convenience we define 0° to be the X positive axis, as illustrated in Fig. 7. By looking at the *Energy-map* shown in Fig. 10, we can immediately see that the direction of movement

is downward.

The surface with maximum energy value lies on $\alpha = 64$ and $\beta = 33$. Substituting these values we get the angle to be 270°, which is correct.

6.1.5 Rotated sequence

The sequence in Fig. 8 is counterclockwise rotated by 23°. The result is classified as a snow fall type where spheres move from north-west to south-east direction as it is shown in Fig. 11. We expect the angle to be approximately $270° + 23° = 293°$.

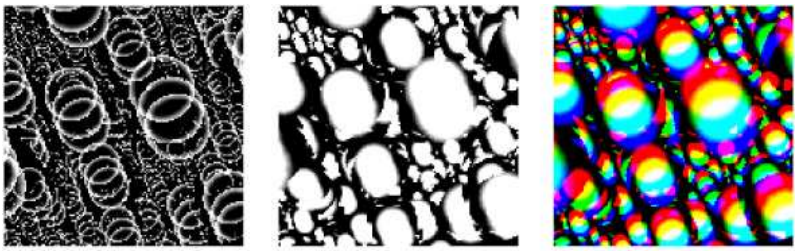


Figure 11: Rotated falling spheres. Three images from the rotated sequence illustrate the spheres movements. Left: Sum of three contour images. Middle: Sum of three images. Right: Colorize three images - red, yellow and blue for the first, second and third images, respectively.

Steps 1-4 output 294.3° which is correct if we take into consideration the accuracy of the algorithms to be $90°/65 \approx 1.4°$.

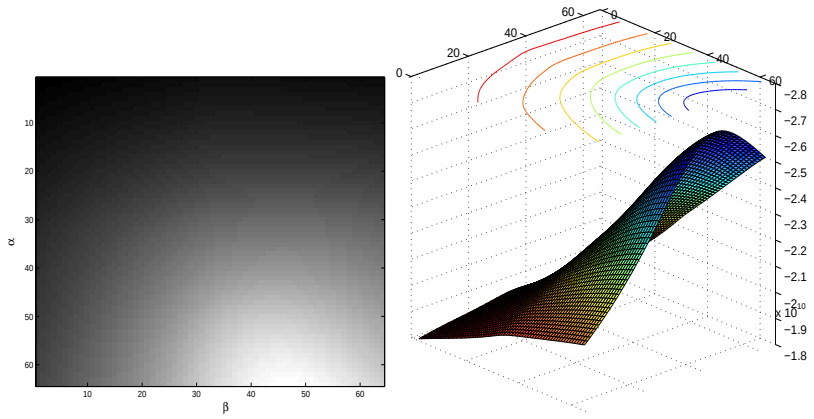


Figure 12: Distribution of the *Energy-map* of Fig. 11 as 2D (Left) and 3D (Right) where the axes are α and β. Higher energy values (colored blue) are located around the object's route.

6.2 Example 2: Traffic movement

We show how to find the global direction of the cars in Fig. 13.

Figure 13: The traffic is analyzed by the algorithm.

6.2.1 Construction of the input

The scene, which we analyze, is similar to the falling snow example. The video was captured by a static camera. We consider the cars to be the spheres in the falling snow example. The viewing angle of the camera does not affect the performance of the algorithm. The frames are of size 128×128. The cars are moving downward. The algorithm was tested in several different periods. The input box is of size $128 \times 128 \times 10$, $128 \times 128 \times 20$ and $128 \times 128 \times 30$. Figure 14 shows a few images from the video input sequence.

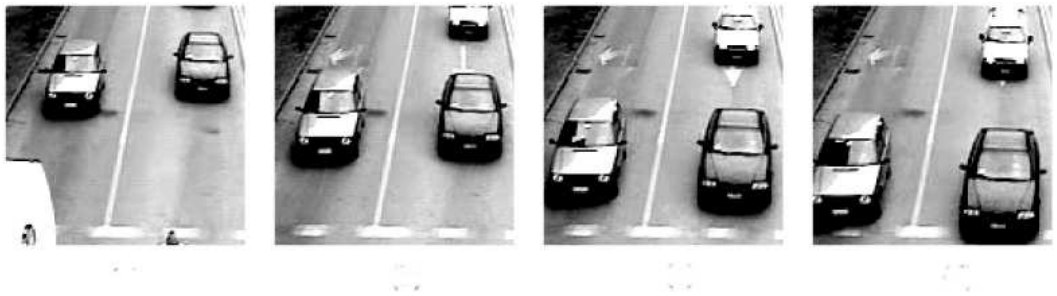


Figure 14: Several instances of video frames from Fig. 13.

Figure 15 shows the *Energy-map* after the application of steps 1-4. The surface in the 3D-Xray box that has maximum energy value, lies in $\alpha = 34$ and $\beta = 33$. By substituting these values into step 4 we get $270°$, which is the right angle.

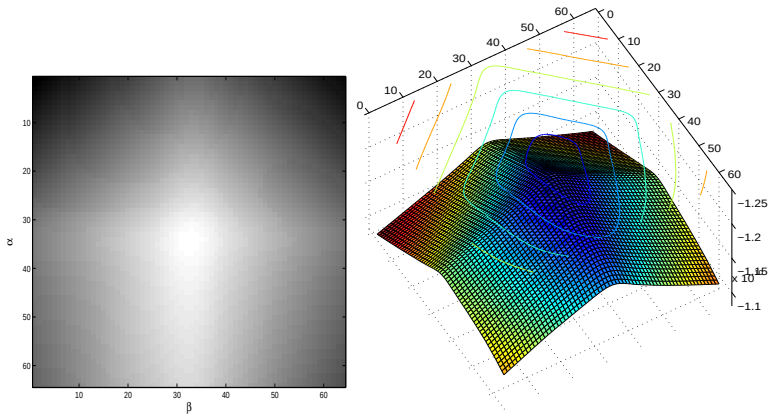


Figure 15: Distribution of the *Energy-map* of Fig. 14 where the axes are α and β. Higher energy values (colored blue) are located on the object's route.

6.2.2 Rotated sequence

The sequence in Fig. 13 is rotated counterclockwise by 25°. The output are cars that move from north-west to south-east direction. Figure 16 shows a few images from this sequence. We expect that the algorithm will detect the movement in 295°.

Figure 16: Rotated scene of the cars that appear in Fig. 13.

Steps 1-4 produce 296° as its output. Figure 17 shows the *Energy-map* of the rotated traffic.

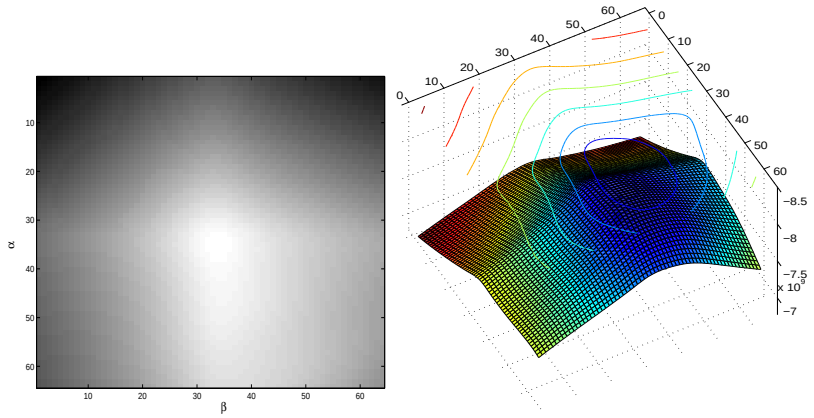


Figure 17: Distribution of the *Energy-map* of Fig. 16 where the axes are α and β. Higher energy values (colored blue) are located on the object's route.

6.3 Example 3: Lateral moving camera that captures a static scene

This example demonstrates how the algorithm operates on a static scene captured by a lateral moving camera. The camera moves horizontally from left to right. We expect that the algorithm will detect the movements in 180°. The camera, which moves from left to right, generates a video sequence. Figure 18 illustrates it.

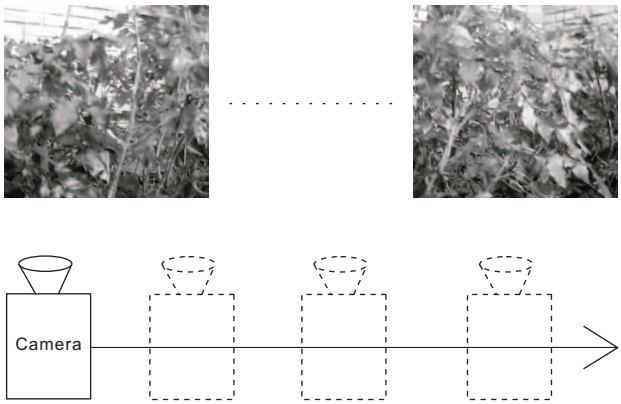


Figure 18: Moving from left to right of a lateral camera that captures a static scene. The video was taken from [2].

This example contains frames of size 128 × 128. The results were checked for several different time periods : 10, 20 and 30 frames with *skip* = 2. Figure 19 shows the *Energy-map* that the algorithm generates from the bush sequence in Fig. 18. The surface that has maximum energy value lies in $\alpha = 32$ and $\beta = 19$. Substituting these values into Step 4 generates 175.91°, which is 4.09° far from the expected result. This is expected due to the accuracy of the algorithm (1.4°) and according to the camera movements on the vertical axis.

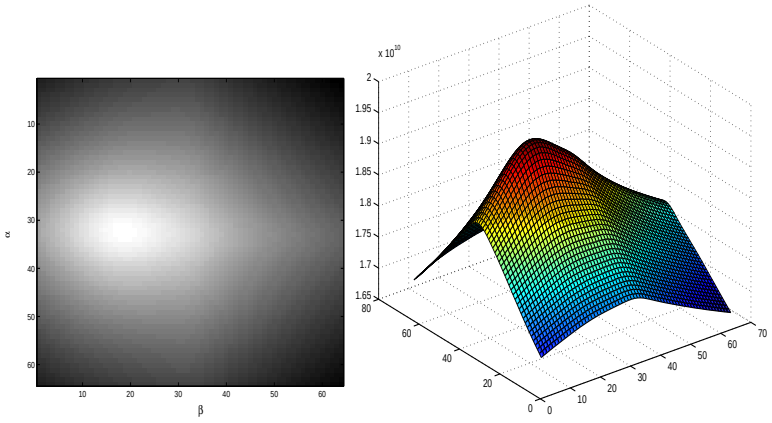


Figure 19: Distribution of the *Energy-map* of Fig. 18 where the axes are α and β. Higher energy values (colored red) are located on the object's route.

6.4 Example 4: Soccer game

We choose few images out of a soccer game (left image in Fig. 20) where most of the players move from left to right. We expect from the algorithm to generate the direction to be 0°. As a preprocessing phase, we segment the video sequence to include only the players without the background. This preprocessing is not required for the performance of the algorithm. The images are of size 128×128 and the algorithms were tested for several different time periods: 10, 20 and 30 frames with $skip = 2$. Figure 20 shows the *Energy-map* of the soccer game sequence. The surface with maximum energy value lies in $\alpha = 33$ and $\beta = 58$. Substituting these values into Step 4 generates the angle 0°, which is the expected result. This example emphasizes that the algorithm can recognize different motions. The result in Fig. 20 is the *Energy-map* that has wide surface of high values. These values demonstrate that motion is recognized all over the field. All the high energy values are located on the right side as expected.

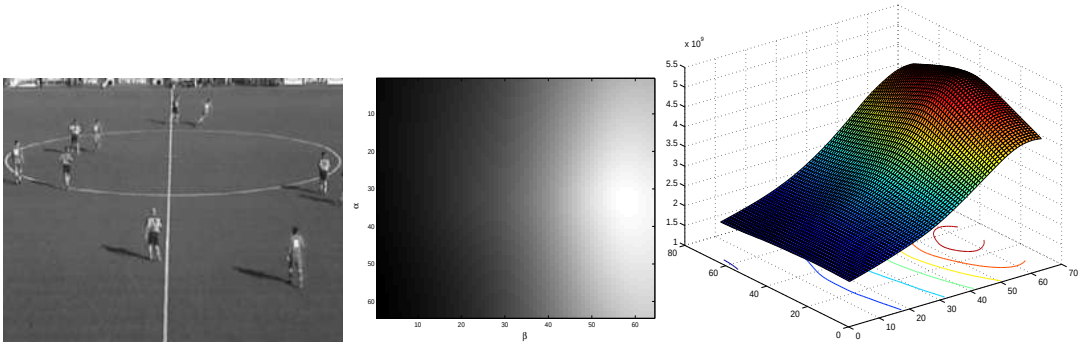


Figure 20: Left: Static camera captures a soccer game. The two rightmost figures are the distribution of the *Energy-map* of the left image where the axes are α and β. Higher energy values (colored red) are located on the objects route.

7 Image segmentation and detection of multiple directions

Sections 4-6 describe an algorithm that computes the angle of the general direction of motion in an *optical snow* scene. We can also use this framework (Steps 1-4) to do motion detection in a specific direction and to segment the moving objects in the video sequence.

7.1 Detection of motion in a specific direction

The output of Step 2 contains the application of the 3D-Xray transform on the video sequence. In order to find the global direction of the movements, Step 4 analyzes the angle (orientation) of the surface that has the maximum energy value in the 3D-Xray planes. Motion in different directions can be detected by analyzing surfaces that have less than maximum energy. For example, assume the input is two crossing objects as illustrated in Fig. 21 (section 7.1.1). One object travels from top-left to bottom-right (denoted by O_1) and the second travels in the opposite direction, from top-right to bottom-left (denoted by O_2). In the *Energy-map* that Step 4 outputs, we see that the moving objects (O_1 and O_2) create two spots with high energy values (see Fig. 22). The spot in the bottom-right corner represents the energy surfaces generated by the motion of O_1. The spot in the bottom-left corner represents the energy surfaces generated by the motion of O_2. This fact enables us to analyze the movements in a specific direction by analyzing only the relevant surfaces in the 3D-Xray output of Step 2. For example, if we want to detect all the people that walk to the left direction in the street then we compute the 3D-Xray transform only for the relevant surfaces and analyze the energy on these surfaces. We can also use this technique to separate between objects in different directions, that have different velocities or different sizes.

This analysis opens new possibilities, for example: 1. The motion analysis is not limited to a single direction like the classical *optical snow*. 2. It enables to segment the viewed scene according to different motion characteristics like direction, velocity and the size of the object.

7.1.1 Example 5: Crossing objects

This example contains two objects that travel in opposite directions and cross each other in the middle of their routes. It illustrates the robustness of the algorithm when objects are occluded in several frames. In addition, it illustrates the capability of the algorithm to detect motion in different directions as explained in section 7.1.

Figure 21 illustrates the video sequence. The sequence contains a white rectangle that travels from top-right to bottom-left and a gray circle that travels from top-left to bottom-right. The objects cross each other in the middle of the image and the circle is partly covered in few frames by the rectangle.

The video sequence is of size 128×128. The algorithms were tested on 50 frames with $skip = 2$. Figure 21 shows the 3D input box of size $128 \times 128 \times 50$.

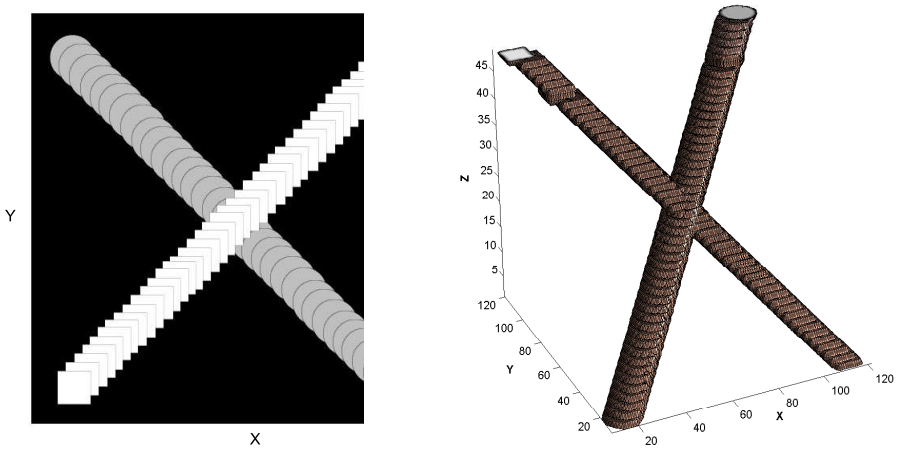


Figure 21: Left: Illustration of synthetic video of two crossing objects. It demonstrates the ability of the algorithm to detect motion in different directions even when occlusions are present. Right: 3D image that was constructed from the video sequence of crossing objects.

Figure 22 is the *Energy-map* of the sequence in Fig. 21. In the *Energy-map* two spots are located at the bottom-left corner and at the bottom-right corner. The spot at the bottom-right corner represents the surfaces that were generated by object O_1 (see section 7.1). The spot at the bottom-left corner represents the surfaces that were generated by the square object O_2 (see section 7.1). Objects O_1 and O_2 generate high energy values, which are well distinguished in spite of the occlusion. We can detect motion in a specific direction by computing the relevant surfaces only. For example, we can distinguish between objects with different velocities. Slow movements create a different trace (tube) than faster movements (see section 4.1). The motion energy of each object is present in a different surface. Therefore, the distinction between slow and fast objects is possible.

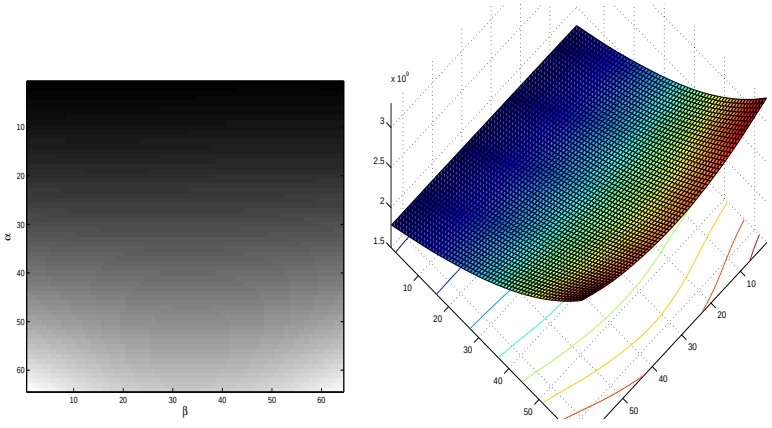


Figure 22: Distribution of the *Energy-map* of Fig. 21. The axes are α and β. High energy values appear in red.

7.1.2 Example 6: Crossing people

Live crossing of people is tested. The example illustrates the robustness of the algorithm even when occlusions are evident (see Fig. 23). The size of the image is 128×128 and the input contains 128 frames. Steps 1-4 are applied with $skip = 2$.

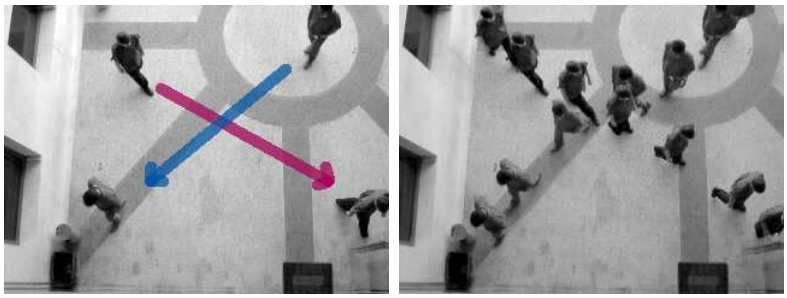


Figure 23: Video sequence of crossing people. Motion in different directions is detected even when occlusions are present. Left: Assembly of the first and last frame where the arrows point to the walking direction. Right: Assembly of few frames.

Figure 24 is the *Energy-map* after the application of Steps 1-4. Two "hot"-spots appear in red at the bottom of the *Energy-map*. Each spot represents the motion of one person as was explained on the objects O_1 and O_2 in Example 5 (section 7.1.1).

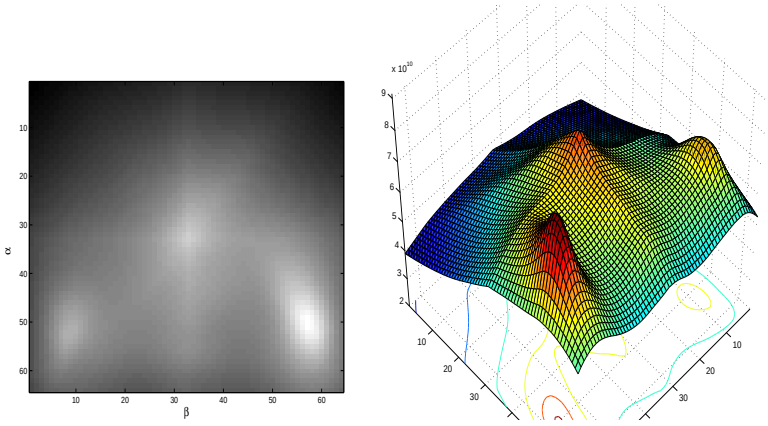


Figure 24: Distribution of the *Energy-map* of Fig. 23. The axes are α and β. High energy values appear in red.

7.2 Segmentation of moving objects

Assume the input is a synthetic image that contains rectangle object that travels from top to bottom (see Fig. 25).

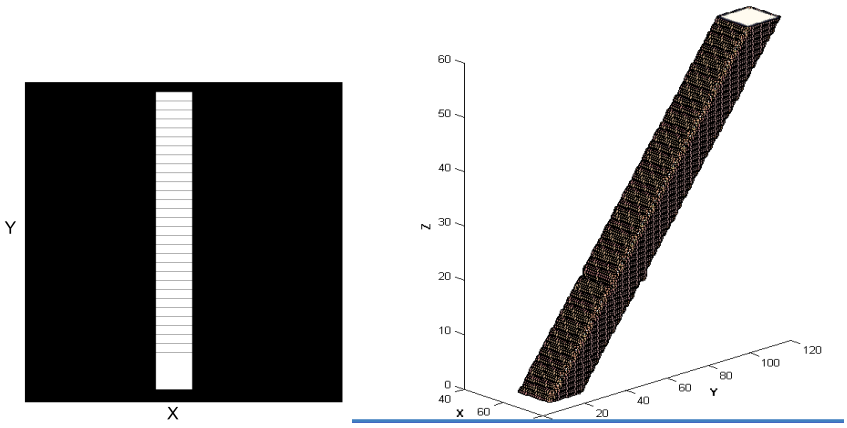


Figure 25: A synthetic video of rectangle object that travels from top to bottom.

The surface with maximum energy is the "most" (the closet to be) perpendicular to the trace (tube) of the object (see section 4.2). Around the object's route we have lower energy values, which look like "shades" (see Fig. 26). These shades are the projection of the object's trace (tube) onto this surface of the 3D-Xray plane. It exists because we use the surface, which is the "most" perpendicular, but not completely perpendicular. Since the 3D-xray transform computes summation along lines, the perpendicular lines to this surface are being visible on the surface. Therefore, the object's trace (tube), which is perpendicular to the surface since it has maximum energy, generates high values that are related to the geometry of the object. A simple threshold creates the blob of object's figure. Figure 27 illustrates the object's trace that it is reflected on the perpendicular surface in the 3D-Xray planes.

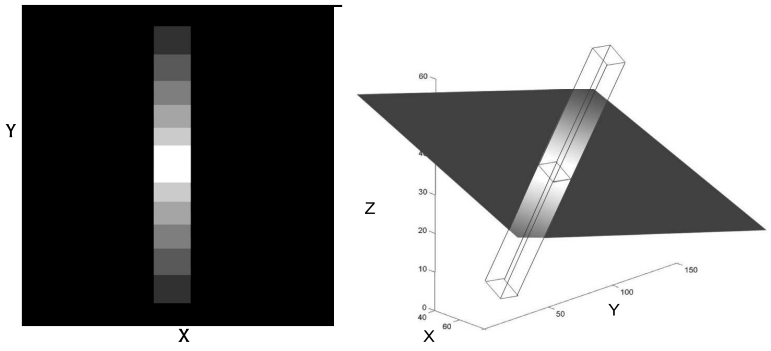


Figure 26: Illustration of the surface with the maximum energy, which is perpendicular to the object's trace. The object's shape is seen where the trace (tube) intersects the surface. The shades around the rectangle were generated by the projection of object's route on this surface, which is not completely perpendicular.

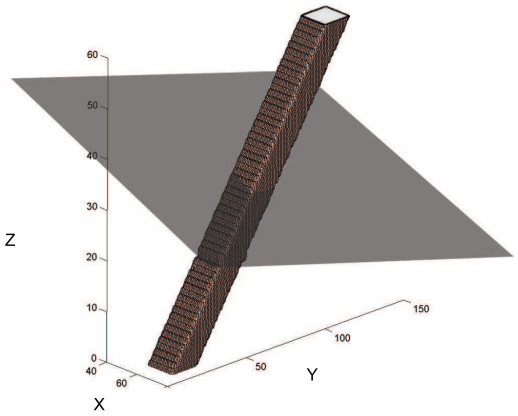


Figure 27: Illustration of the object's trace that intersects the surface that has maximum energy.

8 Conclusions and future research

We present an algorithm that analyzes motion in *optical snow* scenes using the 3D-Xray transform. The algorithm partitions the 3D space into planes. The motion is detected in these specific planes. Through the analysis of plane angles we get different types of information: detection of global direction of movements, detection of motion in a specific direction and segmentation of the input frames.

The algorithms allow efficient computation using 1D frequency domain operations (1D FFTs) with no interpolations.

There are couple of open research topics such as utilization of the information in the 3D-Xray transform to segment moving objects, finding efficient algorithms to do the processing in real-time when new frames are continuously added to the input box and removal of the need to have constant velocity (straight lines) for detection of objects in the input box.

References

[1] A. Averbuch and Y. Shkolnisky. 3D Discrete X-Ray Transform. *Applied Computational Harmonic Analysis*, 17:259–276, 2004.

[2] M. S. Langer and R. Mann. Optical snow. *Int. J. Comput. Vision*, 55(1):55–71, 2003.

[3] Koenderink J.J. Optic flow. (26):161–180, 1986.

[4] S. S. Beauchemin J. L. Barron and D. J. Fleet. On optical flow. In I. Plander, editor, *6th Int. Conf. on Artificial Intelligence and Information-Control Systems of Robots (AIICSR)*, pages 3–14, Bratislava, Slovakia, 1994. World Scientific. Sept. 12-16, 1994, Smolenice Castle, Slovakia.

[5] Andrew B. Watson and Albert J. Ahumada. Model of human visual-motion sensing. volume 2, pages 322–342, 1985.

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

[7] B.K.P. Horn and B.G. Schunck. Determining optical flow: A retrospective. *AI*, 59(1-2):81–87, January 1993.

[8] B.D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *IJCAI81*, pages 121–130, 1981.

[9] E P Simoncelli, E H Adelson, and D J Heeger. Probability distributions of optical flow. In *Proc Conf on Computer Vision and Pattern Recognition*, pages 310–315, Mauii, Hawaii, 1991. IEEE Computer Society.

[10] H. Nagel and W. Enkelmann. an investigation of smoothness constraints for the estimation of displacement vector fields from image sequences. volume 8, pages 565–593. IEEE Trans. Pattern Analysis and Machine Intelligence, 1986.

[11] P. Anandan. a computational framework and an algorithm for the measurement of visual motion. volume 2, pages 283–310. Intl J. Visual Computing, 1989.

[12] Peter J. Burt and Edward H. Adelson. The laplacian pyramid as a compact image code. *IEEE Transactions on Communications*, COM-31,4:532–540, 1983.

[13] A. Singh. Optical flow computation: A unified perspective. *IEEE CS Press*, 1992.

[14] D. Heeger. Optical flow using spatiotemporal filters. *Int'l J. Visual Computing*, 1(2):279–302, 1988.

[15] F. Heitz and P. Bouthemy. Multimodal estimation of discontinuous optical flow using markov random fields. Technical Report 12, Dec. 1993.

[16] F. Heitz and P. Bouthemy. Multimodal estimation of discontinuous optical flow using markov random fields. Technical report, 1997.

[17] Marc Gelgon and Patrick Bouthemy. A region-level graph labeling approach to motion-based segmentation. Technical report, 1997.

[18] C. Kervrann and F. Heitz. A markov random field model-based approach to unsupervised texture segmentation using local and global spatial statistics. Technical Report 6, 1995.

[19] Y. Boykov, O. Veksler, and R. Zabih. Markov random fields with efficient aproximations. Technical report, 1998.

[20] D. Koller, K. Daniilidis, T. Thorhallson, and H.-H. Nagel. Model-based object tracking in traffic scenes. Technical report, 1992.

[21] C. Bregler and J. Malik. Tracking people with twists and exponential maps. Technical report, 1998.

[22] T. Cham and J.M. Rehg. A multiple hypothesis approach to figure tracking. Technical report, 1998.

[23] J. Deutscher, A. Blake, , and I. Reid. Articulated body motion capture by annealed particle filtering. Technical report, 2000.

[24] M.J. Black H. Sidenbladh and D.J. Fleet. Stochastic tracking of 3d human figures using 2d image motion. Technical report, 2000.

[25] C. Fennema and W. Thompson. Velocity determination in scenes containing several moving objects. Technical report, 1979.

[26] A. Waxman and K. Wohn. Contour evolution, neighbourhood deformation and global image flow: Planar surfaces in motion. Technical report, 1985.

[27] J.Y.A. Wang and E.H. Adelson. Layered representation for motion analysis. In *CVPR93*, pages 361–366, 1993.

[28] Trevor Darrell and Alex Pentland. Cooperative robust estimation using layers of support. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(5):474–487, 1995.

[29] Harpreet S. Sawhney and Serge Ayer. Compact representations of videos through dominant and multiple motion estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(8):814–830, 1996.

[30] M.J. Black and P. Anandan. The robust estimation of multiple motions: Parametric and piecewise-smooth flow-fields. volume 63, pages 75–104, 1996.

[31] A. Averbuch and Yoel Shkolnisky. 3D Fourier based discrete Radon transform. *Applied Computational Harmonic Analysis*, 15(1):33–69, July 2003.

[32] Y. Shkolnisky, A. Averbuch, D. L. Donoho, and M. Israeli. The 2-D discrete Radon transform. *Preprint*.