4

Visual Motion Analysis for 3D Robot Navigation in Dynamic Environments

Chunrong Yuan and Hanspeter A. Mallot
Chair for Cognitive Neuroscience
Eberhard Karls University Tübingen
Auf der Morgenstelle 28, 72076 Tübingen, Germany

1. Introduction

The ability to detect movement is an important aspect of visual perception. According to Gibson (Gibson, 1974), the perception of movement is vital to the whole system of perception. Biological systems take active advantage of this ability and move their eyes and bodies constantly to infer spatial and temporal relationships of the objects viewed, which at the same time leads to the awareness of their own motion and reveals their motion characteristics. As a consequence, position, orientation, distance and speed can be perceived and estimated. Such capabilities of perception and estimation are critical to the existence of biological systems, be it on behalf of navigation or interaction.

During the process of navigation, the relative motion between the observer and the environment gives rise to the perception of optical flow. Optical flow is the distribution of apparent motion of brightness patterns in the visual field. In other words, the spatial relationships of the viewed scene hold despite temporal changes. Through sensing the temporal variation of some spatial persistent elements of the scene, the relative location and movements of both the observer and objects in the scene can be extracted. This is the mechanism through which biological systems are capable of navigating and interacting with objects in the external world.

Though it is well known that optical flow is the key to the recovery of spatial and temporal information, the exact process of the recovery is hardly known, albeit the study of the underlying process never stops. In vision community, there are steady interests in solving the basic problem of structure and motion (Aggarwal & Nandhakumar, 1988; Calway, 2005). In the robotics community, different navigation models have been proposed, which are more or less inspired by insights gained from the study of biological behaviours (Srinivasan et al., 1996; Egelhaaf & Kern, 2002). Particularly, vision based navigation strategies have been adopted in different kinds of autonomous systems ranging from UGV (Unmanned Ground Vehicles) to UUV (Unmanned Underwater Vehicles) and UAV (Unmanned Aerial Vehicles). In fact, optical flow based visual motion analysis has become the key to the successful navigation of mobile robots (Ruffier & Franceschini, 2005).

This chapter focuses on visual motion analysis for the safe navigation of mobile robots in dynamic environments. A general framework has been designed for the visual steering of UAV in unknown environments with both static and dynamic objects. A series of robot vision algorithms are designed, implemented and analyzed, particularly for solving the following problems: (1) Flow measurement. (2) Robust separation of camera egomotion and independent object motions. (3) 3D motion and structure recovery (4) Real-time decision making for obstacle avoidance. Experimental evaluation based on both computer simulation and a real UAV system has shown that it is possible to use the image sequence captured by a single perspective camera for real-time 3D navigation of UAV in dynamic environments with arbitrary configuration of obstacles. The proposed framework with integrated visual perception and active decision making can be used not only as a stand-alone system for autonomous robot navigation but also as a pilot assistance system for remote operation.

2. Related Work

A lot of research on optical flow concentrates on developing models and methodologies for the recovery of a 2D motion field. While most of the approaches apply the general spatial-temporal constraint, they differ in the way how the two components of the 2D motion vector are solved using additional constraints. One classical solution provided by Horn & Schunck (Horn & Schunck, 1981) takes a global approach which uses a smoothness constraint based on second-order derivatives. The flow vectors are then solved using nonlinear optimization methods. The solution proposed by Lucas & Kanade (Lucas & Kanade, 1981) takes a local approach, which assumes equal flow velocity within a small neighbourhood. A closed-form solution to the flow vectors is then achieved which involves only first-order derivatives. Some variations as well as combination of the two approaches can be found in (Bruhn et al., 2005). Generally speaking, the global approach is more sensitive to noise and brightness changes due to the use of second-order derivatives. Due to this consideration, a local approach has been taken. We will present an algorithm for optical flow measurement, which is evolved from the well-known Lucas-Kanade algorithm.

In the past, substantial research has been carried out on motion/structure analysis and recovery from optical flow. Most of the work supposes that the 2D flow field has been determined already and assumes that the environment is static. Since it is the observer that is moving, the problem becomes the recovery of camera egomotion using known optical flow measurement. Some algorithms use image velocity as input and can be classified as instantaneous-time methods. A comparative study of six instantaneous algorithms can be found in (Tian et. al., 1996), where the motion parameters are calculated using known flow velocity derived from simulated camera motion. Some other algorithms use image displacements for egomotion calculation and belong to the category of discrete-time methods (Longuet-Higgins, 1981; Weng et al., 1989). The so-called n-point algorithms, e.g. the 8-point algorithm (Hartley, 1997), the 7-point algorithm (Hartley & Zisserman, 2000), or the 5-point algorithm (Nister, 2004; Li & Hartley, 2006), belong also to this category. However, if there are less than 8 point correspondences, the solution will not be unique.

Like many problems in computer vision, recovering egomotion parameters from 2D image flow fields is an ill-posed problem. To achieve a solution, extra constraints have to be sought

after. In fact, both the instantaneous and the discrete-time method are built upon the principle of epipolar geometry and differ only in the representation of the epipolar constraint. For this reason, we use in the following the term image flow instead of optical flow to refer to both image velocity and image displacement.

While an imaging sensor is moving in the environment, the observed image flows are the results of two different kinds of motion: one is the egomotion of the camera and the other is the independent motion of individually moving objects. In such cases it is essential to know whether there exists any independent motion and eventually to separate the two kinds of motion. In the literature, different approaches have been proposed toward solving the independent motion problem. Some approaches make explicit assumptions about or even restrictions on the motion of the camera or object in the environment. In the work of Clarke and Zisserman (Clarke & Zisserman, 1996), it is assumed that both the camera and the object are just translating. Sawhney and Ayer (Sawhney & Ayer, 1996) proposed a method which can apply to small camera rotation and scenes with small depth changes. In the work proposed in (Patwardhan et al., 2008), only moderate camera motion is allowed.

A major difference among the existing approaches for independent motion detection lies in the parametric modelling of the underlying motion constraint. One possibility is to use 2D homography to establish a constraint between a pair of viewed images (Irani & Anadan, 1998; Lourakis et al., 1998). Points, whose 2D displacements are inconsistent with the homography, are classified as belonging to independent motion. The success of such an approach depends on the existence of a dominant plane (e.g. the ground plane) in the viewed scene. Another possibility is to use geometric constraints between multiple views. The approach proposed by (Torr et al., 1995) uses the trilinear constraint over three views. Scene points are clustered into different groups, where each group agrees with a different trilinear constraint. A multibody trifocal tensor based on three views is applied in (Hartley & Vidal, 2004), where the EM (Expectation and Maximization) algorithm is used to refine the constraints as well as their support iteratively. Correspondences among the three views, however, are selected manually, with equal distribution between the static and dynamic scene points. An inherent problem shared by such approaches is their inability to deal with dynamic objects that are either small or moving at a distance. Under such circumstances it would be difficult to estimate the parametric model of independent motion, since not enough scene points may be detected from dynamic objects. A further possibility is to build a motion constraint directly based on the recovered 3D motion parameters (Lobo & Tsotsos, 1996; Zhang et al., 1993). However such a method is more sensitive to the density of the flow field as well as to noise and outliers.

In this work, we use a simple 2D constraint for the detection of both independent motion and outliers. After the identification of dynamic scene points as well as the removal of outliers, the remaining static scene points are used for the recovery of camera motion. We will present an algorithm for motion and structure analysis using a spherical representation of the epipolar constraint, as suggested by (Kanatani, 1993). In addition to the recovery of the 3D motion parameters undergone by the camera, the relative depth of the perceived scene points can be estimated simultaneously. Once the position of the viewed scene points are localized in 3D, the configuration of obstacles in the environment can be easily retrieved.

Regarding the literature on obstacle avoidance for robot navigation, the frequently used sensors include laser range finder, inertial measurement unit, GPS, and various vision systems. However, for small-size UAVs, it is generally not possible to use many sensors due to weight limits of the vehicles. A generally applied visual steering approach is based on the mechanism of 2D balancing of optical flow (Santos-Victor, 1995). As lateral optical flow indicates the proximity of the left and right objects, robots can be kept to maintain equal distance to both sides of a corridor. The commonly used vision sensors for flow balancing are either stereo or omni-directional cameras (Hrabar & Sukhatme, 2004; Zufferey & Floreano, 2006). However in more complex environments other than corridors, the approach may fail to work properly. It has been found that it may drive the robot straight toward walls and into corners, if no extra strategies have been considered for frontal obstacle detection and avoidance. Also it does not account height control to avoid possible collision with ground or ceiling. Another issue is that the centring behaviour requires symmetric division of the visual field about the heading direction. Hence it is important to recover the heading direction to cancel the distortion of the image flow caused by rotary motion.

For a flying robot to be able to navigate in complex 3D environment, it is necessary that obstacles are sensed in all directions surrounding the robot. Based on this concept we have developed a visual steering algorithm for the determination of the most favourable flying direction. One of our contributions to the state-of-the-art is that we use only a single perspective camera for UAV navigation. In addition, we recover the full set of egomotion parameters including both heading and rotation information. Furthermore, we localize both static and dynamic obstacles and analyse their spatial configuration. Based on our earlier work (Yuan et al., 2009), a novel visual steering approach has been developed for guiding the robot away from possible obstacles.

The remaining part is organized as follows. In Section 3, we present a robust algorithm for detecting an optimal set of 2D flow vectors. In Section 4, we outline the steps taken for motion separation and outlier removal. Motion and structure parameter estimation is discussed in Section 5, followed by the visual steering algorithm in Section 6. Performance analysis using video frames captured in both simulated and real world is elaborated in Section 7. Finally, Section 8 summarizes with a conclusion and some future work.

3. Measuring Image Flow

Suppose the pixel value of an image point $p(x, y)$ is $f^t(x, y)$ and let its 2D velocity be $\mathbf{v} = [u, v]^T$. Assuming that image brightness doesn't change between frames, the image velocity of the point $\mathbf{p}$ can be solved as

$$\mathbf{v} = \begin{bmatrix} u \\ v \end{bmatrix} = \mathbf{G}^{-1}\mathbf{b}, \tag{1}$$

with

$$\mathbf{G} = \sum_{(x,y)\in W} \begin{bmatrix} f_x^2 & f_x f_y \\ f_x f_y & f_y^2 \end{bmatrix} \tag{2}$$

and

$$\mathbf{b} = - \sum_{(x,y)\in W} \begin{bmatrix} f_t f_x \\ f_t f_y \end{bmatrix}. \tag{3}$$

Here f_x and f_y are the spatial image gradients, f_t the temporal image derivative, and W a local neighbourhood around point $\mathbf{p}$.

The above solution, originally proposed in (Lucas & Kanade, 1981), requires that $\mathbf{G}$ is invertible, which means that the image should have gradient information in both x and y direction within the neighbourhood W. For the reason of better performance, a point selection process has been carried out before $\mathbf{v}$ is calculated. By diagonalizing $\mathbf{G}$ using orthonormal transform as

$$\mathbf{G} = \mathbf{U}^{-1} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \mathbf{U}, \tag{4}$$

the following criterion can be used to select point $\mathbf{p}$:

1. λ_1 and λ_2 should be big.
2. The ratio of λ_1 / λ_2 should not be too big.

For the purpose of subpixel estimation of $\mathbf{v}$, we use an iterative algorithm, updating f_t sequentially as follows:

$$f_t = f^{t+1}(x+u, y+v) - f^t(x,y). \tag{5}$$

The initial value of $\mathbf{v}$ is set as $\mathbf{v} = [u,v]^T = [\mathbf{0},\mathbf{0}]^T$. To handle large motion, a further strategy is to carry out the above iterative steps in a pyramidal fashion, beginning with the smallest scale image and refining the estimates in consecutive higher scales.

Once a set of points $\{\mathbf{p_i}\}$ has been selected from image $\mathbf{f^t}$ and a corresponding set of $\{\mathbf{v_i}\}$ is calculated, we obtain automatically a set of points $\{\mathbf{q_i}\}$ in image $\mathbf{f^{t+1}}$, with $\mathbf{q_i} = \mathbf{p_i} + \mathbf{v_i}$. In order to achieve higher accuracy in the estimated 2D displacement $\mathbf{v_i}$, we calculate a new set of backward displacement $\{\hat{\mathbf{v}}_\mathbf{i}\}$ for $\{\mathbf{q_i}\}$ from image $\mathbf{f^{t+1}}$ to $\mathbf{f^t}$. As a result we get a set of backward projected point $\{\hat{\mathbf{p}}_\mathbf{i}\}$ with $\hat{\mathbf{p}}_\mathbf{i} = \mathbf{q_i} + \hat{\mathbf{v}}_\mathbf{i}$.

For an accurately calculated displacement, the following equation should hold:

$$e_i = \left\| \hat{\mathbf{p}}_\mathbf{i} - \mathbf{p}_\mathbf{i} \right\| = 0 \,. \tag{6}$$

For this reason, only those points whose $e_i < 0.1$ pixel will be kept. By this means, we have achieved an optimal data set $\{(\mathbf{p_i}, \mathbf{q_i})\}$ with high accuracy of point correspondences established via $\{\mathbf{v_i}\}$ between the pair of image $\mathbf{f^t}$ and $\mathbf{f^{t+1}}$.

4. Motion Separation

While the observing camera of a robot is moving in the world, the perceived 2D flow vectors can be caused either entirely by the camera motion, or by the joined effect of both camera and object motion. This means, the vector $\mathbf{v_i}$ detected at point $\mathbf{p_i}$ can come either from static or dynamic objects in the environment. While static objects keep their locations and configurations in the environment, dynamic objects change their locations with time.

Without loss of generality, we can assume that camera motion is the dominant motion. This assumption is reasonable since individually moving objects generally come from a distance and can come near to the moving camera only gradually. Compared to the area occupied by the whole static environment, the subpart occupied by the dynamic objects is less significant. Hence it is generally true that camera motion is the dominant motion.

As a consequence, it is also true that most vectors $\mathbf{v_i}$ come from static scene points. Under such circumstance, it is possible to find a dominant motion. The motion of static scene points will agree with the dominant motion. Those scene points whose motion doesn't agree with the dominant motion constraint can hence be either dynamic points or outliers. Outliers are caused usually by environmental factors (e.g. changes of illumination conditions or movement of leaves on swinging trees due to wind) that so far haven't been considered during the 2D motion detection process. The goal of motion separation is to find how well each vector $\mathbf{v_i}$ agrees with the dominant motion constraint.

4.1 Motion constraint

Using the method proposed in Section 3, we have gained a corresponding set of points $\{(\mathbf{p_i}, \mathbf{q_i})\}$ with i=1 to N. In order to explain the 2D motion of n static points between $\mathbf{f^t}$ and $\mathbf{f^{t+1}}$, we use a similarity transform $\mathbf{T} = (R, t, s)$ as the motion constraint, where R is a 2D rotation matrix, t a 2D vector and s a scalar. Since $\mathbf{p_i}$ and $\mathbf{q_i}$ are the 2D perspective projections of a set of n static points in two images, the applied constraint is an approximation of the projected camera motion. The transform parameters can be found by minimizing the following distance measure:

$$\varepsilon(R,t,s) = \sum_{i=1}^{n} \| \mathbf{q}_i - (s\mathbf{R}\mathbf{p}_i + \mathbf{t}) \|^2 . \quad (7)$$

We may solve the transform parameters using a linear yet robust minimization method designed by (Umeyama, 1991). Once the parameters of the transform $\mathbf{T}$ are calculated, it can be used to determine whether the scene points agree with the approximated camera motion.

4.2 Motion classification

Since a vector $\mathbf{v_i}$ corresponding to a static scene point is caused by the camera motion, while the $\mathbf{v_i}$ corresponding to a moving scene point is the result of independent motion, the separation of the two kinds of motion is equivalent to classify the set of mixed $\{\mathbf{v_i}\}$ into different classes. Altogether there are three classes: static scene points, dynamic scene points and outliers.

Based on the motion constraint $\mathbf{T} = (R,t,s)$, a residual error can be calculated for each of the points as:

$$d_i = \| (\mathbf{p_i} + \mathbf{v_i}) - (s\mathbf{R}\mathbf{p_i} + \mathbf{t}) \|^2 . \quad (8)$$

We can expect that:

1. d_i =0 $\Rightarrow$ $\mathbf{v_i}$ is correct (inlier), $\mathbf{p_i}$ is a static point
2. d_i is small $\Rightarrow$ $\mathbf{v_i}$ is correct (inlier), $\mathbf{p_i}$ is a dynamic point
3. d_i is very big $\Rightarrow$ $\mathbf{v_i}$ is incorrect, $\mathbf{p_i}$ is an outlier

The remaining problem consists of finding two thresholds k_1 and k_2, so that:

1. if $d_i \le k_1$, $\mathbf{p_i}$ is a static point
2. $k_1 \prec d_i \le k_2$, $\mathbf{p_i}$ is a dynamic point
3. $d_i \succ k_2$, $\mathbf{p_i}$ is an outlier

This belongs to a typical pattern classification problem, which can be solved by analyzing the probabilistic distribution of the set of distance errors $\{d_i\}$. The most direct way is to quantize the distance measures $\{d_i\}$ into L+1 levels, ranging from 0 to L pixels. Following that, a residual distance histogram h(j), j=0 to L, can be calculated. If h(j) is a multimodal histogram, two thresholds k_1 and k_2 can be found automatically for motion separation.

An automatic threshold algorithm has been implemented earlier for the two-class problem (Yuan, 2004). Using this algorithm, we can find a threshold k_1 for separating the points into two classes: one class contains static points; the other is a mixed class of dynamic points and outliers. In case that k_1 doesn't exist, this means there is only a single motion which is the

camera motion. If k_1 does exist, then we will further cluster the remaining mixed set of both dynamic points and outliers by calculating another threshold k_2.

5. Motion & Structure Estimation

Now that we have a set of n points whose image flow vectors are caused solely by the 3D rigid motion of the camera, we can use them to recover the 3D motion parameters. Denoting the motion parameters by a rotation vector $\omega = (r_x, r_y, r_z)^T$ and a translation vector $\mathbf{t} = (t_x, t_y, t_z)^T$, the following two equations hold for a perspective camera with a unit focal length (Horn, 1986):

$$u = \frac{t_x - xt_z}{Z} + [-r_x xy + r_y(x^2 + 1) - r_z y], \tag{9}$$

$$v = \frac{t_y - yt_z}{Z} + [r_y xy - r_x(y^2 + 1) + r_z x], \tag{10}$$

where Z is the depth of the image point $\mathbf{p}$. As can be seen, the translation part of the motion parameter depends on the point depth, while the rotation part doesn't. Without the knowledge of the exact scene depth, it is only possible to recover the direction of $\mathbf{t}$. For this reason, the recovered motion parameters have a total of five degrees of freedom.

As mentioned already in Section 2, we use a spherical representation of the epipolar geometry. Let $\mathbf{u}$ be a unit vector whose ray passes through the image point and $\mathbf{v}$ a unit flow vector whose direction is perpendicular to $\mathbf{u}$, the motion of the camera with parameter $(\omega, \mathbf{t})$ leads to the observation of $\mathbf{v}$ which is equal to

$$\mathbf{v} = -\omega \times \mathbf{u} - (\mathbf{I} - \mathbf{u}\mathbf{u}^T)\mathbf{t} / Z. \tag{11}$$

The goal of motion recovery is to find the motion parameter $(\omega, \mathbf{t})$ that minimizes the following term: $\left\| \mathbf{v} + \omega \times \mathbf{u} + (\mathbf{I} - \mathbf{u}\mathbf{u}^T)\mathbf{t} / Z \right\|$. Using a linear optimization method (Kanatani, 1993), it has been found that the solution for $\mathbf{t}$ is equal to the least eigenvector of a matrix $\mathbf{A} = (\mathbf{A_{ij}})$, i, j=1 to 3 and that

$$\mathbf{A}_{ij} = \mathbf{L}_{ij} - \sum_{k,l,m,n=1}^{3} \mathbf{M}_{ikl} \mathbf{N}^{-1}_{\mathbf{klmn}} \mathbf{M}_{jmn}, \tag{12}$$

with

$$\mathbf{L}_{ij} = \int_{\Omega} \mathbf{v}_i^* \mathbf{v}_j^* d\Omega, \tag{13}$$

$$\mathbf{M}_{ijk} = \int_{\Omega} \mathbf{v}_i^* \mathbf{v}_j \mathbf{v}_k d\Omega, \tag{14}$$

$$\mathbf{N}_{ijkl} = \int_{\Omega} \mathbf{v}_i \mathbf{v}_j \mathbf{v}_k \mathbf{v}_l d\Omega, \tag{15}$$

where

$$\mathbf{v}^* = \mathbf{u} \times \mathbf{v}. \tag{16}$$

Once $\mathbf{t}$ is recovered, the solution for ω can be computed as:

$$\omega = \frac{1}{2}\left[tr(\mathbf{K}) + 3\mathbf{t}^T\mathbf{K}\mathbf{t}\right]\mathbf{t} - 2\mathbf{K}\mathbf{t}, \tag{17}$$

where $tr(\,)$ is the trace of a matrix and

$$\mathbf{K} = (\mathbf{K}_{ij}), \tag{18}$$

$$\mathbf{K}_{ij} = -\sum_{k,l,m,n=1}^{3} \mathbf{N}_{ijkl}^{-1} \mathbf{M}_{mkl} \mathbf{t}_m. \tag{19}$$

Subsequently, the depth Z of a scene point $\mathbf{p}$ can be estimated as

$$Z = \frac{1 - (\mathbf{u}^T\mathbf{t})^2}{\mathbf{u}^T(\omega \times \mathbf{t}) - \mathbf{v}^T\mathbf{t}}. \tag{20}$$

If $(\omega, \mathbf{t})$ is a solution, then $(\omega, -\mathbf{t})$ can also be a solution. The correct one can be chosen based on the cheirality constraint, by assuring positive scene depth (Z>0).

6. Visual Steering

6.1 Obstacle detection

After the motion parameters as well as the relative scene depths of the static points are calculated, we now obtain the viewing direction of the camera together with the location of a set of 3D scene points relative to the camera.

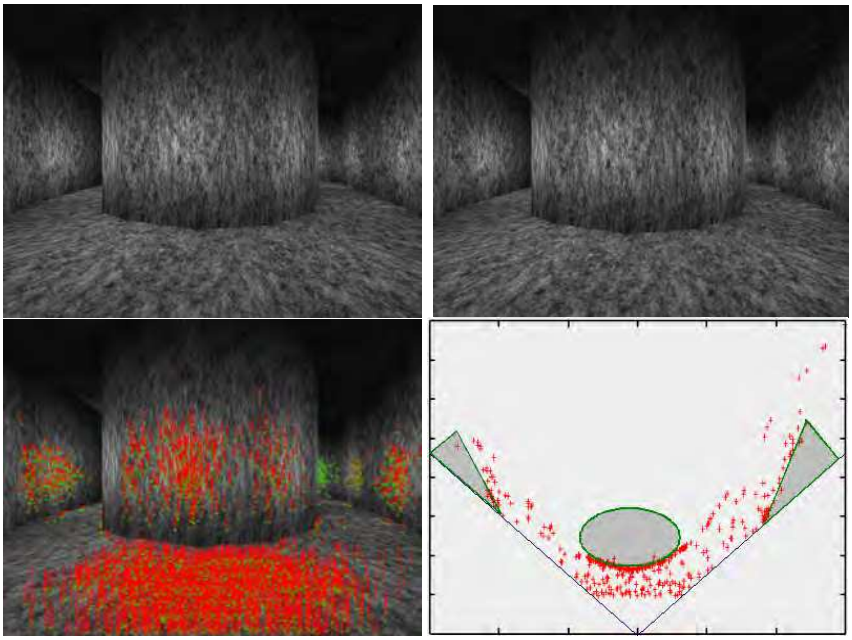


Fig. 1. Configuration of detected 3D scene points relative to the camera.

Shown in Fig. 1 is an illustration of the visual processing results so far achieved. In the top row are two images taken by a camera mounted on a flying platform. The result of image flow detection is shown on the left side of the bottom row. Obviously the camera looking in the direction of the scene is flying upward, since the movement of the scene points are downward, as is shown by the red arrows with green tips. The image on the bottom right of Fig. 1 shows the configuration of detected 3D scene points (coloured in red) relative to the camera. Both the location and orientation of those 3D scene points and the orientation of the camera have been recovered by our motion analysis algorithm. The blue \/ shape in the image represents the field-of-view of the camera. The obstacles are shown in solid shape filled with colour gray. They are the left and right walls as well as the frontal pillar. The goal of visual steering is to determine in the next step a safe flying direction for the platform.

As shown by Fig. 1, each image point $\mathbf{p_i}$ in $\mathbf{f^t}$ corresponds to a 3D point $\mathbf{P_i}$ with a depth Z_i. This distance value indicates the time-to-contact of a possible obstacle in the environment. Our visual steering approach exploits the fact that the set of depth values reveals the distribution of obstacles in different directions. Specifically, we use a concept built upon directional distance sensors to find the most favourable moving direction based on the distance of nearest obstacles in several viewing directions. This is done through a novel idea of cooperative decision making from visual directional sensors.

6.2 Directional distance sensor

A single directional sensor is specified by a direction $\mathbf{d}$ and an opening angle α that defines a viewing cone from the camera center. All the scene points lying within the cone defines one set of depth measurements. Based on the values of these depth measurements, the set can be further divided into a few depth clusters. The clustering criterion is that each cluster K is a subset with at least s scene points whose distances to the cluster center are below δ. The parameter s and δ are chosen depending on the size of the viewing cone and the density of depth measurements.

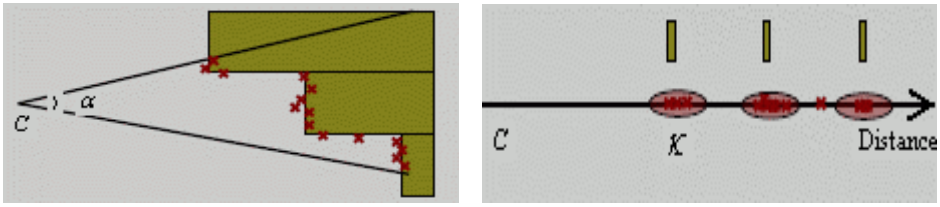


Fig. 2. A directional distance sensor and scene points with corresponding depth clusters.

Shown in Fig. 2 on the left is a directional distance sensor (located at camera center *c*) with the set of detected scene points lying within the viewing cone. The set is divided into three depth clusters, as is shown on the right of Fig. 2. Note that it is possible that some points may not belong to any clusters, as is the case of the points lying left to the rightmost cluster.

Once the clusters are found, it is possible to find the subset κ whose distance to the viewing camera is shortest. In the above example, the nearest cluster is the leftmost one. With the nearest cluster κ identified, its distance to the camera, represented as D_κ, can be determined as the average distance of all scene points belonging to κ. In order to determine whether it is safe for the UAV to fly in this direction, we encode D_κ in a fuzzy way as near, medium and far. Depending on the fuzzy encoding of D_κ, preferences for possible motion behaviours can be defined as follows:

1. Encoded value of D_κ is far, flying in this direction is favourable
2. Encoded value of D_κ is medium, flying in this direction is still acceptable
3. Encoded value of D_κ is near, flying in this direction should be forbidden.

If we scan the viewing zone of the camera using several directional distance sensors, we may obtain a set of nearest depth clusters κ_i together with a corresponding set of fuzzy-encoded distance values. By assigning motion preferences in each direction according to these distance values, the direction with the highest preferences can be determined. Built exactly upon this concept, novel visual steering strategies have been designed.

6.3 Visual steering strategies

Three control strategies are considered for the visual steering of the UAV: horizontal motion control, view control and height control.

The purpose of view control is to ensure that the camera is always looking in the direction of flight. By doing so, the principle axis of the camera is aligned with the forward flight direction of the UAV so that a substantial part of the scene lying ahead of the UAV can always be observed. Because the camera is firmly mounted on the UAV, changing the viewing direction of the camera is done by changing the orientation of the UAV. Here we would like to point out the relationship between the different coordinate systems. A global coordinate system is defined whose origin is located at the optical center of the camera. The optical axis of the camera is aligned with the z axis pointing forward in the frontal direction of the UAV. The image plane is perpendicular to the z axis, with the x axis pointing horizontally to the right side of the UAV and the y axis pointing vertically down. View control is achieved by rotating the UAV properly, which is done by setting the rotation speed of the UAV around the y axis of the global coordinate system. Obviously this is the yaw speed control for the UAV.

The main part of visual steering is the horizontal motion control. We have defined five motion behaviours: left (←), forward and left (↖), forward (↑), forward and right (↗) and right (→). Once the flying direction is determined, motion of the UAV is achieved by setting the forward motion speed, left or right motion speed and turning speed (yaw speed). The yaw control is necessary because we want to ensure that the camera is aligned with the direction of flight for maximal performance of obstacle avoidance. Hence a left motion will also result in modifying the yaw angle by rotating the UAV to the left via the view control.

In order to realize the horizontal motion control as well as the view control, it is necessary to select one safe flying direction from the five possibilities defined above. We define five virtual directional distance sensors which are located symmetrically around the estimated heading direction. This corresponds to a symmetric division of the visual field into far left, left, front, right and far right, as is shown in Fig. 3.

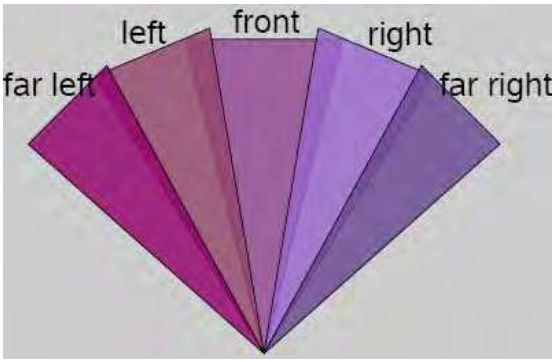


Fig. 3. A symmetric arrangement of five directional distance sensors for visual steering.

As mentioned in Section 6.1, each of these five directional sensors perceives the nearest obstacles in a particular direction. Depending on the nearness of the obstacles detected, every directional sensor will output one preference value for each of the five possible motion behaviours. Three preference values have been defined. They are favourable (FA), acceptable (AC) and not acceptable (NA).

Each directional sensor has its own rules regarding the setting of the preference values. An example of rule setting for the front sensor is given in Table 1. Table 2 shows another example for the far left sensor. As can been seen, once a fuzzy encoded distance measure is determined, a directional sensor outputs a total of five preference values, with one for each possible moving direction.

Distance	Behavioural preferences for each of the five motion directions				
	←	↖	↑	↗	→
far	AC	AC	FA	AC	AC
medium	AC	FA	AC	FA	AC
near	FA	AC	NA	AC	FA

Table 1. Preference setting rules for the front distance sensor.

Distance	Behavioural preferences for each of the five motion directions				
	←	↖	↑	↗	→
far	FA	AC	AC	AC	AC
medium	AC	FA	AC	AC	AC
near	NA	AC	FA	AC	AC

Table 2. Preference setting rules for the far left distance sensor.

From all the five directional sensors shown in Fig. 3, we have got altogether a set of 25 preferences values, five for each moving direction. By adding the preference values together, the motion behaviour with the highest preference can be selected as the next flying direction.

Suppose the fuzzy distance values of the five directional sensors (from left to right) are near, far, far, medium and near, the preference values for each motion behaviour can be determined individually, as shown in Table 3. If we take all the sensors into account by adding all the preference values appearing in each column, the final preference value for each motion direction can be obtained, as is shown in the second last line of Table 3. Apparently, the highest preference value is achieved for the forward direction. Hence the safest flying direction is moving forward.

Sensor	Distance	Behavioural preferences for each of the five motion directions				
		←	↖	↑	↗	→
far left	near	NA	AC	FA	AC	AC
left	far	AC	FA	AC	AC	AC
front	far	AC	AC	FA	AC	AC
right	medium	AC	AC	FA	AC	AC
far right	near	AC	AC	FA	AC	NA
All sensors		1 NA 4 ACs	1 FA 4 ACs	4 FAs 1 AC	5 ACs	1 NA 4 ACs
Decision				○		

Table 3. Decision making based on the fuzzy encoded distance values of all five sensors.

As for the height control of the UAV, a single directional distance sensor looking downwards is used. It estimates the nearness of obstacles on the ground and regulates the height of the UAV accordingly. In addition, we take into account the vertical component of the estimated motion parameter of the camera, which is t_y. The direction of t_y can be up, zero or down. The goal is to let the UAV maintain approximately constant distance to the ground and avoid collision with both ground and ceiling. This is performed by increasing/decreasing/keeping the rising/sinking speed of the UAV so as to change the height of the UAV. Decision rules for the height control of the UAV can be found in Table 4.

ty	Estimated distance to ground		
	near	medium	far
up	no speed change	decrease rising speed	decrease rising speed
zero	increase rising speed	no speed change	increase sinking speed
down	increase rising speed	increase rising speed	no speed change

Table 4. Using a vertical directional sensor and t_y for height control.

7. Experimental Evaluation

The proposed approach has been implemented using C++ running on a Samsung M60 laptop. For each pair of frames ($\mathbf{f}^t, \mathbf{f}^{t+1}$), we first detect 3000 features in frame $\mathbf{f}^t$ and try to find their correspondences in $\mathbf{f}^{t+1}$. Depending on the nature of the input frames, the found number of point correspondences N ranges from a few hundreds to a few thousands. Thanks to the linear methods used, a frame rate of 15 frames/s can be achieved for images with a resolution 640x480, including both motion analysis and visual steering steps.

Both indoor and outdoor experiments have been carried out for evaluation purpose. We have captured videos using both hand-held camera and the camera mounted on a flying robot. Shown in Fig. 4 is the AR-100 UAV we have used, which is a kind of small-size (diameter < 1m) drone whose weight is below 1kg. The onboard camera inclusive the protecting case is ca. 200g. The images captured by the camera are transmitted via radio link

to the laptop on which our motion algorithm is running. The UAV has a remote control panel. In manual flight mode, controlling commands are triggered by human operation. In auto-flight mode, the controlling commands for avoiding detected obstacles are calculated by our visual steering algorithm and sent via radio transmission to the drone. Through a switch on the control panel, the mode can be changed by the human operator.

Fig. 4. The AR-100 UAV with the camera mounted beneath the flight board.

In the following we analyze the results achieved, concentrating on one particular aspect in a single subsection.

7.1 Performance on motion separation

For the evaluation of our motion separation algorithm, we have used several video sequences captured by mobile cameras navigating in the natural environment. The images in the first video sequence are taken on a sunny day in an outdoor environment, using the camera on the drone. The dynamic objects to be detected are people moving around the drone. Some example images with people moving are shown in Fig. 5. As can be seen clearly from the second image, the quality of the transmitted image is not perfect. Altogether there are 3082 frames in the video sequence. Among them, there are 1907 frames which consist only of static scenes. In each of the remaining 1175 frames, there are either one or two objects moving.

Fig. 5. Some images in video sequence 1.

Shown in Fig. 6 on the left is one example image together with the detected 2D displacement vectors and the result of motion separation. There are a few outliers in the static background as well as on the moving person. This is due largely to illumination variations. Those long vectors (with colour yellow) are the outliers. The vectors shown in colour black come from the static background. The two moving persons have been identified correctly, as can be seen clearly from the vectors shown as red lines with green tips. Another example is shown

in Fig. 6 on the right, where neither independent motion nor outliers occur. In both cases, it is obvious from the visualized flow vectors that the camera motion consists of both rotation and translation component.

Fig. 6. Examples of detection result in sequence 1.

Sequence 2 is captured with a hand-held camera. The moving object to be detected is the AR-100 flying drone. There are also some people moving in the background. In Fig. 7 on the left we can see one input frame, where the moving drone is even difficult to perceive with human eyes. Shown on the right is the result of detection. Three moving objects have been identified. They are the flying drone as well as two persons moving behind the tree. The purpose of performance evaluation with sequence 2 is to find how our algorithm will behave in case the size of the object is very small compared to the visual field of the camera. All together there are 80 frames, with the drone moving all the time in the scene.

Fig. 7. Examples of detection result in sequence 2.

Shown in Fig. 8 are two examples of detection results achieved on the third video sequence (video available at http://robots.stanford.edu/). The moving object is a robot car running on the road. Altogether there are 303 frames. Each frame has a single moving object in it.

Fig. 8. Examples of detection result in sequence 3.

On all three sequences, we have achieved an average separation accuracy of 83.3%. This means, among all those flow vectors detected, 83.3% of them have been correctly classified. Particularly, the flow vectors coming from static scene points have been identified with a precision of 90%, while those form dynamic scene points with an accuracy of slightly over 70%. The false alarm rate of moving object detection is below 5%. Considering that the dynamic objects are either small or moving at far distances and that we use only the last and current frame for motion separation, the results are quite encouraging.

7.2 Performance on motion estimation

Once the flow vectors coming from static scene points have been identified, we use RANSAC together with the approach presented in Section 5 to refine the inliers for correct estimation of motion parameters.

In order to estimate the accuracy of the estimated motion parameters, we need to have the ground truth values of them. This is nearly impossible when a camera is moving freely in space. For this reason we use a simulated environment where the ground truth is available. During the flight of an agent in a virtual world, real images are rendered continuously, with the motion between consecutive frames known. Using these images, we extract flow vectors and compute the motion parameters of the flying agent. They are then compared with the known ground truth to compute the precision of our algorithm. The images rendered from the virtual world have a horizontal field of view of 110 degrees, which is slightly larger than that of the camera on the real robot. To be compatible with real-world situations, we vary the illumination of the virtual world to simulate the lighting changes appeared usually in the real world.

A corridor environment has been constructed and two tests have been performed. The first test consists of a simple rotation around the vertical axis. The agent rotates first 360 degrees clockwise and then 360 degrees anticlockwise. It has been found that the deviation of the estimated final heading direction lies within 1 degree. In Fig. 9 a, the white line shows the heading direction of the agent relative to the corridor.

a b c

Fig. 9. Results of motion estimation.

The second test consists of a flight through the corridor, where the agent computes its translation and rotation parameters from image flow measurements. The average accuracy of the estimated heading direction is 2 degrees. In Fig. 9 b and c, one can observe the difference between the estimated and the ground truth motion of the agent. We project the estimated heading direction into the image and mark it with a red 'x'. The projected point of the ground truth direction of the heading is shown as a red '+'.

Further performance analysis on motion estimation has been carried out through the reconstruction of real-world camera motion. We first calculate the real camera motion from images captured by the flying robot in the outdoor environment. The calculated motion parameters have been used to generate images simulating a virtual camera moving in a textured environment. By observing synchronously the real and virtual videos, the quality of the motion estimation can be visually observed. The fact that no difference has been perceived between the two videos indicates that our algorithm on motion estimation has achieved similar results using real-world images.

7.3 Performance analysis on visual steering

The first visual steering experiment is carried out in the above corridor. A top view of the whole corridor is shown with red colour in Fig. 10 on the left. After the motion and scene depth have been calculated from two initial images, the UAV is set to move in the direction determined by our visual steering algorithm. Then the UAV captures the next frame, determine its next moving direction and moves accordingly. During the experiment, we have recorded the trajectory of the UAV. Shown in Fig 10 on the left in colour green is the recorded trajectory of the UAV. As demonstrated by this green curve, the vehicle is able to navigate through the corridor without any collision.

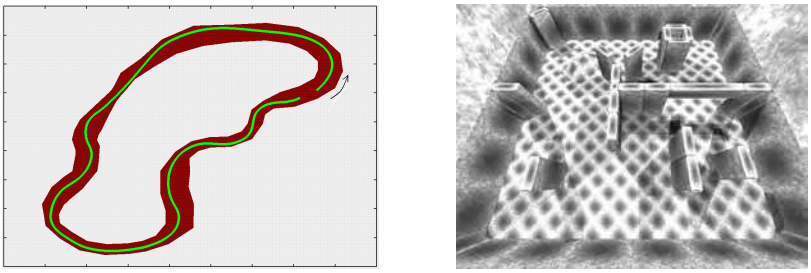


Fig. 10. Left: 3D flight in the corridor environment. Right: the maze-like environment.

Further experiments are carried out in a maze-like environment with irregular configurations of obstacles, as is shown in Fig. 10 on the right. In the beginning, the vehicle is placed randomly within the maze. The flying direction is set to the viewing direction of the camera, i.e., the forward direction. Several experiments with different starting positions show that the vehicle is capable of navigating safely within the maze by finding its way automatically. Particularly, it is able to regulate both flying direction and height to avoid collision with walls, corners, buildings, the ground, the arch etc. Some images showing the path of the drone during its navigation are shown in Figure 11.

As shown in Fig. 11 a to b, one can observe that the vehicle has found free space for navigation and change its flying direction toward the arch to avoid collision with walls and corners. From there, the agent is flying through the arch, as can be read out from Fig 11 image b, c and d. Following that, the vehicle is able to prepare the turning around the corner shown in image e. Having turned around the corner, the vehicle is guided by our visual steering algorithm to fly along the wall and adjust height due to the level of terrain and the obstacle floating in the air, as is shown by image f.

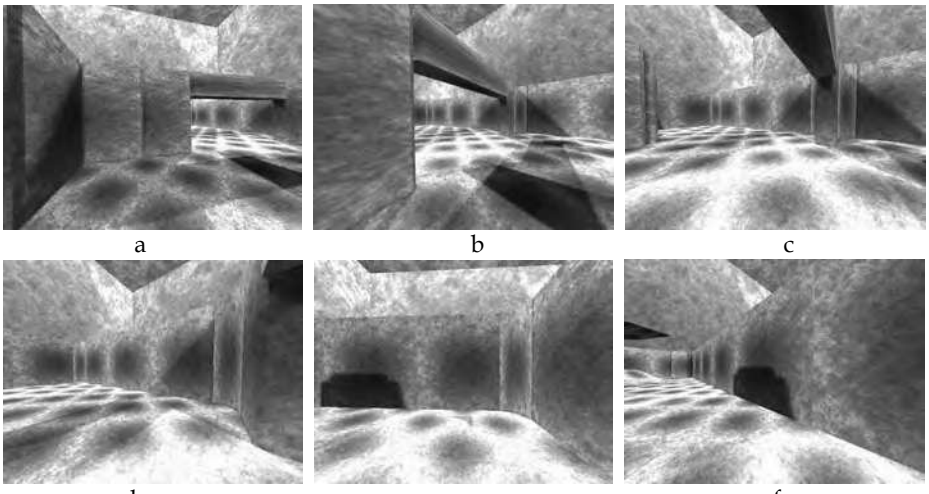


Fig. 11. 3D Flight within the maze.

Further experiments are carried out in an office experiment, which is a T-shaped corridor with isolated obstacles. As can be seen from Fig. 12, the environment is not entirely static, as the doors may be closed and opened. This is indeed an unknown dynamic environment with general configuration of obstacles. In this sense, performance in this environment indicates the effectiveness of our whole visual steering approach, including robust image flow calculation, motion separation, motion and depth estimation, and the determination of flying direction.

In Fig.12 we show one example of the experiments made in this environment. Both the visual path of the flying platform and the decisions made by our algorithm can be viewed.

Based on the output of our visual steering algorithm, the vehicle is able to maintain constant distance to the ground. The decision for horizontal motion control has been shown with red arrows, indicating the next flying direction. The length of the arrows is set in inverse proportion to the distance of obstacles. A longer arrow indicates a shorter distance to the detected obstacle and hence a faster speed needed by the UAV for flying away from it. As can be seen, the platform has been kept safely in the middle of the free space.

Fig. 12. 3D visual steering in the office environment.

8. Conclusion

This chapter concentrates on visual motion analysis for the safe navigation of mobile robots in dynamic environment. The aim is to build one of the important navigation abilities for robot systems: the detection of obstacles for collision avoidance during the 3D autonomous flight of UAVs. In dynamic environment, not only the robot itself but also some other objects are moving. With the proposed approach, we have shown a robot vision system capable of understanding the natural environment, analyzing the different motions and making appropriate decisions.

Most motion estimation algorithms work well with perfect image flow measurement but are very sensitive to noise and outliers. To overcome this problem, we have designed a complete computational procedure for robust 3D motion/structure recovery. A well-known image flow algorithm has been extended and improved for the robust detection of image flow vectors. In order to estimate the camera motion, we proposed a novel approach for the separation of independent motion and removal of outliers. The motion parameters of the camera and the 3D position and orientation of scene points are then recovered using a linear estimation approach. With the output of our visual motion analysis, we are able to facilitate a flying platform with obstacle detection and avoidance ability. As a result, safe and autonomous navigation of UAV systems can be achieved.

As mentioned already, the UAV has both manual and auto flight mode. However, our visual steering algorithm runs independent of the mode chosen. Even in manual mode, the human pilot can observe the visual processing results. Alerted by the decision made by our algorithm, the pilot can hence avoid possible errors and trigger correct operation. In this sense, the robot vision system developed can be used as a pilot assistance system as well. Currently we provide only a primitive visualization of the dynamic/static objects in the environment together with the decision made by our algorithm for the horizontal motion control. Visualization of the decision made for height control is omitted for the purpose of simplicity. We plan to improve the way of information presentation for better human robot interaction.

In the vision system presented, we use always two frames, the current and the last one, for making a decision. While quite efficient, its drawback is that the valuable visual processing results achieved in the past are not considered in the current analysis step. Such results can be used for example for the detection of independent motion. If a moving object is detected in several past frames, it may be more advantageous to switch from detection to tracking method for localizing the object in the current frame. This will probably improve largely the performance regarding the detection of dynamic objects. By tracking dynamic objects continuously, it is also possible to estimate their 3D motion parameters constantly. With the heading direction of the dynamic objects known, a more advance and effective visual steering strategy can be designed.

Another future research direction is to combine the visual steering approach with that of camera-based SLAM (simultaneous localization and mapping). We are updating our UAV system to have two cameras mounted on it. While the available camera looks ahead in the flying direction, another camera will point downwards to the ground. Through the second camera, more features on the ground can be observed. Together with the forward-looking camera, motion as well as location of the UAV can be estimated more accurately. A more elaborate 3D motion constraint can be derived for better motion separation and analysis. At the same time, both safer and efficient navigation of UAV can be achieved by combining the current way of reactive steering with that of goal-directed motion planning.

9. Acknowledgement

This work originates from research activities carried out on the project μDrones (Micro drone autonomous navigation for environment sensing). Funding from the European Union is gratefully acknowledged.

10. References

Aggarwal, J. K. & Nandhakumar, N. (1988). On the computation of motion from sequences of images --- A review, *Proceedings of the IEEE,* Vol. 76, No. 8 (August 1988) pp (917-935), ISSN 0018-9219

Bruhn, A.; Weickert, J. and Schnörr, C. (2005). Lucas/Kanade meets Horn/Schunck: Combining local and global optical flow methods, *Int. Journal of Computer Vision,* Vol. 61, No. 3 (March 2005) pp (211-231), ISSN 0920-5691

Calway, A. (2005). Recursive estimation of 3D motion and structure from local affine flow parameters, *IEEE Trans. Pattern Analysis and Machine Intelligence,* Vol. 27, No. 4 (April 2005) pp (562-574), ISSN 0162-8828

Clarke, J. C. & Zisserman, A. (1996). Detecting and tracking of independent motion, *Image and Vision Computing,* Vol. 14, No. 8 (August 1996) pp (565-572), ISSN 0262-885

Egelhaaf, M. & Kern, R. (2002). Vision in flying insects. *Current Opinion in Neurobiology,* Vol. 12, No. 6 (December 2002) pp (699-706), ISSN 0059-4388

Gibson, J. J. (1974). *The perception of the visual world,* Greenwood Press, ISBN 0-8371-7836-3, Westport, Connecticut, USA.

Hartley, R. I. (1997). In defense of the eight-point algorithm, *IEEE Trans. Pattern Analysis and Machine Intelligence,* Vol. 19, No. 6 (June 1997) pp (580-593), ISSN 0162-8828

Hartley R. I. and Zisserman, A. (2000). *Multiple View Geometry in Computer Vision,* Cambridge University Press, ISBN 0-5216-2304-9, Cambridge, UK.

Hartley, R. I. & Vidal, R. (2004). The multibody trifocal tensor: motion segmentation from 3 perspective views, *Proceedings of Int. Conf. on Computer Vision and Pattern Recognition (CVPR 2004)*, pp. 769-775, ISBN 0-7695-2158-4, Washington DC, USA, July 2004, IEEE Computer Society, Washington, DC.

Horn, B. K. P. & Schunck, B. G. (1981). Determing optical flow. *Artificial Intelligence,* Vol. 17, No. 1-3 (August 1981) pp (185-203), ISSN 0004-3702

Horn, B. K. P. (1986). *Robot Vision,* Mit Press, ISBN 0-2620-8159-8, Cambridge, MA, USA.

Hrabar, S. & Sukhatme, G. S. (2004). A comparison of two camera configurations for optic-flow based navigation of a UAV through urban canyon, *Proceedings of 2004 IEEE/RSJ Int. Conf. on Intelligent Robots and systems (IROS 2004),* 2673-2680, ISBN 0-7803-8463-6, Sendai, Japan, September 2004, IEEE press.

Irani, M. & Anadan, P. (1998). A unified approach to moving object detection in 2D and 3D scenes, *IEEE Trans. Pattern Analysis and Machine Intelligence,* Vol. 20, No. 6 (June 1998) pp (577-589), ISSN 0162-8828

Kanatani, K. (1993). 3-D interpretation of optical flow by renormalization, *Int. Journal of Computer Vision,* Vol. 11, No. 3 (December 1993) pp (267-282), ISSN 0920-5691

Li, H. & Hartley, R. (2006). Five point motion estimation made easy, *Proceedings of Int. Conf. on Pattern Recognition (ICPR 2006)*, pp. 630-633, ISBN 0-7695-2521-0, Hongkong, China, August 2006, IEEE Computer Society, Washington, DC.

Lobo, N. & Tsotsos, J. (1996). Computing egomotion and detecting independent motion from image motion using collinear points. *Computer Vision and Image Understanding,* Vol. 64, No. 1 (July 1996) pp (21-52), ISSN 1077-3142

Longuet-Higgins, H. C. (1981). A computer algorithm for reconstructing a scene from two projections, *Nature,* Vol. 293, No. 5828 (September 1981) pp (133-135), ISSN 0028-0836

Lourakis, M.; Argyros, A. & Orphanoudakis, S. (1998). Independent 3D motion detection using residual parallax normal flow field, *Proceedings of the sixth Int. Conf. on Computer Vision (ICCV 1998)*, pp. 1012-1017, ISBN 8-1731-9221-9, Bombay, India, January 1998, Narosa Publishing House, New Delhi, India.

Lucas, B.D. & Kanade, T. (1981). An iterative image registration technique with an application to stereo vision, *Proceedings of the 7th Int. Joint Conf. on Artificial Intelligence (IJCAI'81)*, pp. 674-679, ISBN 1-5586-0044-2, Vancouver, British Columbia, Canada, August 1981, William Kaufmann, Los Altos, CA.

Nister, D. (2004). An efficient solution to the five-point relative pose problem, *IEEE Trans. Pattern Analysis and Machine Intelligence,* Vol. 26, No. 6 (June 2004) pp (756-770), ISSN 0162-8828

Patwardhan, K.; Sapiro, G. & Morellas, V. (2008). Robust foreground detection in video using pixel layers, *IEEE Trans. Pattern Analysis and Machine Intelligence,* Vol. 30, No. 4 (April 2008) pp (746-751), ISSN 0162-8828

Ruffier, F. & Franceschini, N. (2005). Optic flow regulation: the key to aircraft automatic guidance. *Robotics and Autonomous Systems,* Vol. 50, No. 4 (March 2005) pp (177-194), ISSN 0921-8890

Santos-Victor, J. ; Sandini, G.; Curotto, F. & Garibaldi, S. (1995). Divergent stereo for robot navigation: A step forward to robotic bee, *Int. Journal of Computer Vision,* Vol. 14, No. 2 (March 1995) pp (159-177), ISSN 0920-5691

Sawhney, H.S. & Ayer, S. (1996). Compact representations of videos through dominant and multiple motion estimation, *IEEE Trans. Pattern Analysis and Machine Intelligence,* Vol. 18, No. 8 (August 1996) pp (814-830), ISSN 0162-8828

Srinivasan, M. V.; Zhang, S.W.; Lehrer, M. & Collett, T.S. (1996). Honeybee navigation en route to the goal: visual flight control and odometry. *The Journal of Experimental Biology,* Vol. 199, No. 1 (January 1996) pp (237-244), ISSN 0022-0949

Tian, T.Y.; Tomasi, C. & Heeger, D.J. (1996). Comparison of approaches to egomotion computation, *Proceedings of Int. Conf. on Computer Vision and Pattern Recognition (CVPR 1996)*, pp. 315-320, ISBN 0-8186-7258-7, Washington, DC, USA, June 1996, IEEE Computer Society, Washington, DC.

Torr, P.; Zisserman, A. & Murray, D. (1995). Motion clustering using the trilinear constraint over three views, *Proceedings of the Europe China Workshop on Geometric Modelling and Invariants for Computer Vision (GMICV'95),* 118-125, ISBN 7-5656-0383-1, Xian, China, April 1995, Xidian University Press, Xian.

Umeyama, S. (1991). Least-squares estimation of transformation parameters between two point patterns, *IEEE Trans. Pattern Analysis and Machine Intelligence,* Vol. 13, No. 4, (April 1991) pp (376-380), ISSN 0162-8828

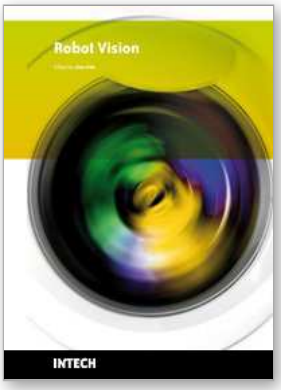
Weng, J.; Huang, T. S. & Ahuja, N. (1989). Motion and structure from two perspective views: algorithms, error analysis and error estimation, I*EEE Trans. Pattern Analysis and Machine Intelligence,* Vol. 11, No. 5 (May 1980) pp (451-476), ISSN 0162-8828

Yuan, C (2004). Simultaneous tracking of multiple objects for augmented reality applications, *Proceedings of the seventh Eurographics Workshop on Multimedia (EGMM 2004),* 41-47, ISBN 3-9056-7317-7, Nanjing, China, October 2004, Eurographics Association.

Yuan, C.; Recktenwald, F. & Mallot, H. A. (2009). Visual steering of UAV in unknown environment, *Proceedings of 2009 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS 2009),* pp. 3906-3911, ISBN 978-1-4244-3803-7, St. Louis, Missouri, USA, October 2009, IEEE press.

Zhang, Z.; Faugeras, O. and Ayache, N. (1993). Analysis of a sequence of stereo scenes containing multiple moving objects using rigidity constraints, *Proceedings of Int. Conf. on Computer Vision (ICCV 1993)*, pp. 177-186, ISBN 0-8186-3870-2, Berlin, Germany, May 1993, IEEE Computer Society, Washington DC.

Zufferey, J. C. & Floreano, D. (2006). Fly-inspired visual steering of an ultralight indoor aircraft, *IEEE Trans. Robotics,* Vol. 22, No. 1, (January 2006) pp (137-146), ISSN 1552-3098



Robot Vision

Edited by Ales Ude

ISBN 978-953-307-077-3
Hard cover, 614 pages
Publisher InTech
Published online 01, March, 2010
Published in print edition March, 2010

The purpose of robot vision is to enable robots to perceive the external world in order to perform a large range of tasks such as navigation, visual servoing for object tracking and manipulation, object recognition and categorization, surveillance, and higher-level decision-making. Among different perceptual modalities, vision is arguably the most important one. It is therefore an essential building block of a cognitive robot. This book presents a snapshot of the wide variety of work in robot vision that is currently going on in different parts of the world.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821