Updating Camera Location and Heading Using a Sparse Displacement Field

Report LiTH-ISY-R-2318

Per-Erik Forssén

Computer Vision Laboratory, Department of Electrical Engineering

Linköping University, SE-581 83 Linköping, Sweden

November 20, 2000

1 Introduction

This report describes the principles of an algorithm developed within the WITAS project [3]. The goal of the WITAS project is to build an autonomous helicopter that can navigate autonomously, using differential GPS, GIS-data of the underlying terrain (elevation models and digital orthophotographs) and a video camera.

Using differential GPS and other non-visual sensory equipment, the system is able to obtain crude estimates of its position and heading direction. These estimates can be refined by matching of camera-images and the on-board GIS-data. This refinement process, however is rather time consuming, and will thus only be made every once in a while. For real-time refinement of camera position and heading, the system will iteratively update the estimates using frame to frame correspondence only.

In each frame a sparse set of image displacement estimates are calculated, and from these the perspective in the current image can be found. Using the calculated perspective and knowledge of the camera parameters, new values of camera position and heading can be obtained.

The resultant camera position and heading can exhibit a slow drift if the original alignment was not perfect, and thus a corrective alignment with GIS-data should be performed once every minute or so.

2 Principles

The estimation of perspective relies on the existence of image locations that correspond to positions on the ground in the real world. At present the algorithm also requires the ground to be planar. This requirement could however easily be removed if required, although this will require a denser displacement field.

The image produced by the camera is assumed to be the result of a perspective projection of features in *the ground plane*, onto *the image plane*, followed by a *lens distortion*. If

the camera is set to the maximal amount of zoom, the lens distortion is usually negligible. However, for completeness we will explain how to deal with lens distortion in section 3.2.

For the first frame, the perspective projection is known (it has been computed by finding the correspondence between image and GIS-data). We can thus compute the ground plane coordinates of the points in the image plane for this frame. Thus, we first find a number of points that are well suited to tracking in the first frame, then compute their locations in the ground plane. By keeping track of the displacements of the points in the image plane, the perspective projections for the following frames can now be computed (see sections 4 and 5). Once we know the perspective projection relating the image and ground planes, we can finally compute the camera location and viewing direction (see section 6).

3 Geometry

Before we can describe the algorithm in detail, we have to define some terms and concepts relating to the camera parameters, and projective geometry.

3.1 Camera parameters

The camera coordinate system (CCS) is defined as a right-hand system $\{\ \hat{\boldsymbol{n}}\quad \hat{\boldsymbol{u}}\quad \hat{\boldsymbol{n}}\times\hat{\boldsymbol{u}}\ \}$, with the origin at the focal point of the lens. The cross-product vector, $\hat{\boldsymbol{r}} = \hat{\boldsymbol{n}}\times\hat{\boldsymbol{u}}$ points right from the camera, and the view-plane normal, $\hat{\boldsymbol{n}}$, points forward, into the image (see figure 1).

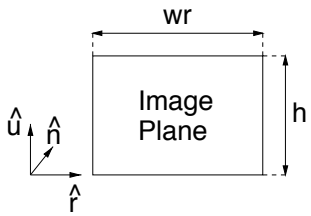

Figure 1: Camera coordinate system (CCS)

The image dimensions in pixels are $w \times h$, but in a physical metric, we have to take non-square pixels into account, and thus we multiply the image width by the *aspect ratio*, r, and get $wr \times h$ as indicated in figure 1.

The camera position and orientation can be described by three vectors:

$\hat{\boldsymbol{n}}$	*view plane normal*
$\hat{\boldsymbol{u}}$	*camera up vector*
$\boldsymbol{p}$	*camera position*

The vector $\boldsymbol{p}$ above is only meaningful in the world coordinate system (WCS). In CCS it is always zero.

In addition to these parameters we need to know, either the view angles α_w and α_h, or the focal length $f = h/2\tan(\alpha_h/2) = wr/2\tan(\alpha_w/2)$ and the aspect ratio r (see figure 2). But since the value of f depends on the physical dimensions of the detector element,

the view angles are preferable. The focal length and the aspect ratio can when needed be derived from the view angles.

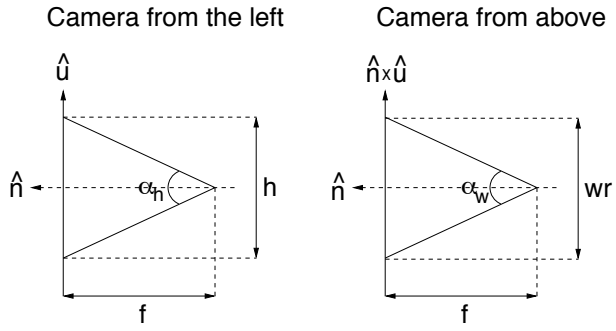

Figure 2: Camera parameters

Optionally we could also include a radial lens distortion parameter $\gamma = \gamma(f)$ to describe the camera state. Adding a lens distortion parameter will allow us to deal with the non uniform spacing of pixels caused by the lens.

3.2 Lens distortion

Figure 3: Lens distortion.
A frame in a sample sequence before (left), and after (right) lens distortion compensation.

Most camera lenses distort the perspective in the scene in such a way that the pixels are moved closer to the centre of the image (see figure 3). The amount of distortion is a function of the zoom (the focal length), and varies radially [4]. Modelling the lens distortion becomes easy if we express it in a polar camera coordinate system, $\{r, \varphi\}$, with the origin situated at the point $\boldsymbol{p_c}$, where the optical axis intersects the image plane. The lens distortion can now be written as a function of the radius only:

$$r_n = r - \gamma r^3 \quad \text{when } r < 1/\sqrt{3\gamma} \tag{1}$$

The displacement of a pixel in the scene $\boldsymbol{x} = (\ x_1 \quad x_2\ )^T$ can thus be written as:

$$\boldsymbol{y} = \boldsymbol{x} - \gamma(\boldsymbol{x} - \boldsymbol{p_c})r^2 \tag{2}$$
$$r^2 = (\boldsymbol{x} - \boldsymbol{p_c})^T(\boldsymbol{x} - \boldsymbol{p_c}) \tag{3}$$

Where $\boldsymbol{y} = (\ y_1 \quad y_2\ )^T$ is the new position [4].

In order to be able to use lens-distortion compensation, we need to know the value of the γ parameter. In a real-time situation, this means that we have to know the current amount of zoom, and the relationship between zoom and γ.

In the example in figure 3, we did not know this relationship. Instead we have estimated γ by making sure that all structures that are straight on the map also are straight in the adjusted video frame.

The relationship between zoom and γ can be found for instance by filming a pattern consisting of straight lines, and altering the zoom in a known manner.

The inverse to the lens-distortion formula (equation 1) can be found using the cube cosine relationship.[1] This method of finding real valued inverses to third-degree polynomial equations is described in detail in [1]. We rewrite equation 1 as:

$$r^3 - \frac{1}{\gamma}r + \frac{1}{\gamma}r_n = 0$$

The inverse now becomes:

$$r = \beta\cos\left(\alpha - \frac{2\pi}{3}\right) \quad \text{for} \tag{4}$$
$$\beta = \sqrt{\frac{4}{3\gamma}} \quad \text{and} \quad \alpha = \frac{1}{3}\arccos\left(\frac{-3r_n}{\beta}\right)$$

3.3 A compact model of the camera view

The camera coordinate system CCS, and the world coordinate system WCS are normally different. We can change world coordinates into camera coordinates, by applying a translation $\boldsymbol{T}$ that moves the camera position $\boldsymbol{p}$ to the WCS origin, followed by a rotation $\boldsymbol{R}$ that aligns the axes of CCS and WCS. In homogeneous coordinates these affine mappings can be described by two matrices:

$$\boldsymbol{T} = \begin{pmatrix} 1 & 0 & 0 & -p_1 \\ 0 & 1 & 0 & -p_2 \\ 0 & 0 & 1 & -p_3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \text{and} \quad \boldsymbol{R} = \begin{pmatrix} u_1 & u_2 & u_3 & 0 \\ r_1 & r_2 & r_3 & 0 \\ n_1 & n_2 & n_3 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

The variable components of the matrices above are coordinates of the vectors $\hat{\boldsymbol{u}}$, $\hat{\boldsymbol{r}}$, $\hat{\boldsymbol{n}}$, and $\boldsymbol{p}$ in WCS. The compound mapping will look like:

[1] $\cos 3\alpha = 4\cos^3\alpha - 3\cos\alpha$

$$\boldsymbol{RT} = \begin{pmatrix} u_1 & u_2 & u_3 & -\hat{\boldsymbol{u}}^t\boldsymbol{p} \\ r_1 & r_2 & r_3 & -\hat{\boldsymbol{r}}^t\boldsymbol{p} \\ n_1 & n_2 & n_3 & -\hat{\boldsymbol{n}}^t\boldsymbol{p} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

This matrix will map a point $(\; x_w \quad y_w \quad z_w \quad 1 \;)^t$ in homogeneous world coordinates to camera coordinates:

$$\begin{pmatrix} y_c h_c \\ x_c h_c \\ z_c h_c \\ h_c \end{pmatrix} = \boldsymbol{RT} \begin{pmatrix} x_w \\ y_w \\ z_w \\ 1 \end{pmatrix}$$

A camera scene point $(\; y_c \quad x_c \quad z_c \;)^t$ can be transformed into an image point $(\; y_i \quad x_i \;)^t$ as follows:

$$\begin{pmatrix} y_i \\ x_i \end{pmatrix} = \frac{f}{z_c} \begin{pmatrix} y_c \\ x_c \end{pmatrix}$$

Due to this, and to the fact that $\boldsymbol{RT}$ will give us results multiplied by h_c, we can thus move from 3D homogeneous coordinates to 2D homogeneous coordinates through the following mapping:

$$\begin{pmatrix} y_i h_i \\ x_i h_i \\ h_i \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1/f & 0 \end{pmatrix}}_{\boldsymbol{P}} \begin{pmatrix} y_c h_c \\ x_c h_c \\ z_c h_c \\ h_c \end{pmatrix}$$

The compound mapping $\boldsymbol{PRT}$ will look like this:

$$\boldsymbol{PRT} = \begin{pmatrix} u_1 & u_2 & u_3 & -\hat{\boldsymbol{u}}^t\boldsymbol{p} \\ r_1 & r_2 & r_3 & -\hat{\boldsymbol{r}}^t\boldsymbol{p} \\ n_1/f & n_2/f & n_3/f & -\hat{\boldsymbol{n}}^t\boldsymbol{p}/f \end{pmatrix}$$

If we make the approximation $z = 0$ (flat ground), we can remove one column from the compound mapping. We then arrive at a plain 2D perspective transformation in homogeneous coordinates:

$$\boldsymbol{H} = \boldsymbol{PRTF} = \boldsymbol{PRT} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} u_1 & u_2 & -\hat{\boldsymbol{u}}^t\boldsymbol{p} \\ r_1 & r_2 & -\hat{\boldsymbol{r}}^t\boldsymbol{p} \\ n_1/f & n_2/f & -\hat{\boldsymbol{n}}^t\boldsymbol{p}/f \end{pmatrix}$$

Finally this matrix will be normalised by division with the element h_{33}.

If our GIS-data also contains elevation information, the $\boldsymbol{F}$ matrix should be omitted. This will lead to three more parameters to estimate in the $\boldsymbol{H}$ matrix, and somewhat different formulations of the formulae in the following sections.

4 Estimation of perspective

We can model the projection of a point $\boldsymbol{x} = \begin{pmatrix} x_1 & x_2 & 1 \end{pmatrix}^T$ in the WCS ground plane, into homogeneous camera coordinates as:

$$\boldsymbol{y_h} = \boldsymbol{H}\boldsymbol{x}$$

The components of $\boldsymbol{H}$ can be identified as a translation $\boldsymbol{b}$, a rotation $\boldsymbol{A}$, and skewing with distance, $\boldsymbol{c}$:

$$\begin{pmatrix} y_{h,1} \\ y_{h,2} \\ h \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & b_1 \\ a_{21} & a_{22} & b_2 \\ c_1 & c_2 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ 1 \end{pmatrix} \tag{5}$$

$$\boldsymbol{y} = \frac{1}{h} \begin{pmatrix} y_{h,1} \\ y_{h,2} \end{pmatrix} \tag{6}$$

In order to find the perspective in the image, we thus have to estimate a total of eight parameters. In principle, we could either estimate these from the projection of the image plane onto the ground plane, or the projection of the ground plane onto the image plane. However, since we are going to solve the problem as a least-square fit, and we want to minimise the error in the ground plane, we estimate the parameters of the mapping from the image plane onto the map. This projection can be estimated if we know a set of coordinates $\boldsymbol{y}$ in the image plane, and $\boldsymbol{x}$ in the ground plane. We first reformulate the projection equation in a form suitable for a least-squares solution:

$$\boldsymbol{y} = \boldsymbol{S}\boldsymbol{p} \tag{7}$$

Where the matrices $\boldsymbol{S}$ and $\boldsymbol{p}$ are defined as:

$$\boldsymbol{S} = \begin{pmatrix} x_1 & x_2 & 1 & 0 & 0 & 0 & -x_1y_1 & -x_2y_1 \\ 0 & 0 & 0 & x_1 & x_2 & 1 & -x_1y_2 & -x_2y_2 \end{pmatrix} \tag{8}$$

$$\boldsymbol{p} = \begin{pmatrix} a_{11} & a_{12} & b_1 & a_{21} & a_{22} & b_2 & c_1 & c_2 \end{pmatrix} \tag{9}$$

For each pair of coordinates, we thus get two equations. If we combine all these equations into the matrices $\boldsymbol{y}$, and $\boldsymbol{S}$, we can find the parameters $\boldsymbol{p}$ as:

$$\boldsymbol{p} = (\boldsymbol{S}^T\boldsymbol{S})^{-1}(\boldsymbol{S}^T\boldsymbol{y}) \tag{10}$$

From these coefficients, we can now compose the sought perspective projection matrix as:

$$\boldsymbol{H_i} = \begin{pmatrix} p_1 & p_2 & p_3 \\ p_4 & p_5 & p_6 \\ p_7 & p_8 & 1 \end{pmatrix} \tag{11}$$

The corresponding inverse projection, i.e. the ground-to-image-plane projection, is computed as:

$$\boldsymbol{H} = \frac{1}{(\boldsymbol{H_i}^{-1})_{33}} \boldsymbol{H_i}^{-1} \tag{12}$$

An example of this method is shown in figure 4. The left and centre images show a set of corresponding points. The resultant projection, overlaid on the map, is shown in the image to the right.

Figure 4: Reference points, and the resultant projection.
The correspondence between a set of points in the map (left), and the scene (middle) is needed in order to estimate the parameters of the perspective projection (right).

If we want to deal with lens distortion as well, we should apply the inverse of the lens-distortion formula (see equation 4) on the $\boldsymbol{y}$ coordinates before computing the least-squares fit.

5 Keeping track of the perspective

As discussed in section 1, the perspective estimation is initiated with known camera location and viewing angle. Using these we can create an initial perspective transformation matrix, as described in section 3.3.

Using the Harris corner detector [5], we can find regions in the scene that do not exhibit the *aperture problem* [2]. We then keep track of these regions using the $\cos^2$-*region tracking* algorithm described in [6].

After some time, the regions we have selected will inevitably change or move out of view due to the camera movement. Thus we have to select new points at regular intervals.

We should try to keep the points as long as possible, since each time we change templates we run the risk of introducing an error.

Each time we select a new point to track, we project it onto the ground plane. At each moment, we thus have a set of coordinates in the image plane, with known positions in the ground plane, and can estimate the perspective using the method described in section 4.

The described method requires that the points we track actually lie in the ground plane. If they don't, clearly the method will fail. Thus we have to remove the *outliers*, the points that do not fit the plane model. Removal of outliers can be made if we look at the *residuals*, r_k, in the image-to-ground transformation:

$$\boldsymbol{x}_{\boldsymbol{p}_k} = \boldsymbol{H}_{\boldsymbol{i}}\boldsymbol{x}_k \tag{13}$$

$$r_k = \sqrt{(\boldsymbol{x}_{\boldsymbol{p}_k} - \boldsymbol{y}_k)^T(\boldsymbol{x}_{\boldsymbol{p}_k} - \boldsymbol{y}_k)} \tag{14}$$

That is, we project our image coordinates onto the ground, and compute the distances to their assumed locations.

If the largest r_k is above a certain threshold, the projection of this point onto the ground plane has moved. This could be either because the point belongs to a moving object, or because it does not lie in the ground plane (parallax movement). Either way, this point should not be used to estimate new perspectives. When detecting outliers this way, we can only find them one at a time. The second largest outlier could in fact lie in the ground plane, and pull in the opposite direction compared to the largest.

Usually outliers are not detected right away, and thus all points selected after the outlier will have slightly erroneous ground-plane coordinates. If, and how this should be dealt with is an unsettled question.

Since the point detection algorithm is blind to what points it finds, we might as well keep tracking the outliers, but exclude them from the perspective estimation, since otherwise we will probably detect, and start tracking them again.

6 Computing the camera parameters

The camera parameters used to build the initial perspective projection matrix (see section 3.3) can be extracted again from the perspective projection matrix $\boldsymbol{H}$.

The first step in this process is to remove the influence of the focal length. If we know the focal length this is easy:

$$\boldsymbol{H} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & f \end{pmatrix} \boldsymbol{H}_f = \alpha \begin{pmatrix} u_1 & u_2 & -\hat{\boldsymbol{u}}^t\boldsymbol{p} \\ r_1 & r_2 & -\hat{\boldsymbol{r}}^t\boldsymbol{p} \\ n_1 & n_2 & -\hat{\boldsymbol{n}}^t\boldsymbol{p} \end{pmatrix}$$

As indicated above, the elements of this matrix can be identified as components of the CCS vectors, except for a scaling factor α.

6.1 Estimating the focal length

In many situations we can obtain a probable value of the focal length from the $\boldsymbol{H}_f$, but this value is not guaranteed to be robust.

In order to do this, we note that $\{\ \hat{\boldsymbol{n}}\ \ \hat{\boldsymbol{u}}\ \ \hat{\boldsymbol{r}}\ \}$ constitutes an ON basis. and thus:

$$\boldsymbol{R}^{-1} = \begin{pmatrix} u_1 & u_2 & u_3 \\ r_1 & r_2 & r_3 \\ n_1 & n_2 & n_3 \end{pmatrix}^{-1} = \begin{pmatrix} u_1 & r_1 & n_1 \\ u_2 & r_2 & n_2 \\ u_3 & r_3 & n_3 \end{pmatrix} = \boldsymbol{R}^t$$

The matrix $\boldsymbol{R}$ maps world coordinates into camera coordinates, and thus $\boldsymbol{R}^t$ does the opposite.

This means that after we have removed the influence of the focal length, the first two columns of $\boldsymbol{H}$ should have the same norm ($= \alpha$) since they are scaled versions of vectors in an ON basis. This gives us the following relations for the coefficients $\{h_{kl}\}$ of $\boldsymbol{H}_f$:

$$\begin{array}{c} h_{11}^2 + h_{21}^2 + f^2 h_{31}^2 = \alpha^2 \\ h_{12}^2 + h_{22}^2 + f^2 h_{32}^2 = \alpha^2 \\ f > 0 \end{array} \quad \Rightarrow \quad f = \sqrt{\frac{h_{11}^2 + h_{21}^2 - h_{12}^2 - h_{22}^2}{h_{32}^2 - h_{31}^2}}$$

In other words, we can estimate f, if we restrict the solution to positive values (these are the only ones that make physical sense anyway). Note that this method will not work when the difference $h_{32}^2 - h_{31}^2$ is close to zero.

6.2 The camera coordinate system

Once the influence of the focal length has been removed from $\boldsymbol{H}$, we can estimate the magnitude of the scaling factor α. We are at this point unable to determine the sign of α. This means that the vectors $\hat{\boldsymbol{n}}$, and $\hat{\boldsymbol{u}}$ we will compute could just as well have the opposite signs.

To find the magnitude of α, we again make use of the fact that the first two rows of $\boldsymbol{H}$ are vectors in an ON basis. This gives us the following relations for the coefficients $\{h_{kl}\}$ of $\boldsymbol{H}$:

$$\begin{array}{c} h_{11}^2 + h_{21}^2 + h_{31}^2 = \alpha_1^2 \\ h_{12}^2 + h_{22}^2 + h_{32}^2 = \alpha_2^2 \end{array}$$

These equations will give us two values of α. If our value of f was correct, and if there was no measurement errors, they should be identical. In practise however, they will differ slightly, so we set our final scaling value to their average $\alpha = 0.5(\alpha_1 + \alpha_2)$.

After normalising the rows of $\boldsymbol{H}$ (with α_1 and α_2 respectively), we can find the third vector of our ON basis as their cross product:

$$\frac{1}{\alpha_1}\begin{pmatrix} h_{11} \\ h_{21} \\ h_{31} \end{pmatrix} \times \frac{1}{\alpha_2}\begin{pmatrix} h_{12} \\ h_{22} \\ h_{32} \end{pmatrix} = \hat{\boldsymbol{w}} = \begin{pmatrix} u_3 \\ r_3 \\ n_3 \end{pmatrix}$$

Thus our estimates of $\hat{\boldsymbol{n}}$, $\hat{\boldsymbol{u}}$, and $\hat{\boldsymbol{r}}$ will be:

$$\begin{array}{l} \hat{\boldsymbol{n}} = \begin{pmatrix} h_{11}/\alpha_1 & h_{12}/\alpha_2 & w_1 \end{pmatrix} \\ \hat{\boldsymbol{u}} = \begin{pmatrix} h_{21}/\alpha_1 & h_{22}/\alpha_2 & w_2 \end{pmatrix} \\ \hat{\boldsymbol{r}} = \begin{pmatrix} h_{31}/\alpha_1 & h_{32}/\alpha_2 & w_3 \end{pmatrix} \end{array}$$

These values should be used when computing the camera position $\hat{\boldsymbol{p}}$, but to describe the camera orientation we might have to change the sign of them as well. The sign can be resolved, for instance if we know earlier, or approximate directions $\hat{\boldsymbol{n}}_0$, and $\hat{\boldsymbol{u}}_0$. We can now find correct values of $\hat{\boldsymbol{n}}$ and $\hat{\boldsymbol{u}}$ as:

$$\begin{array}{c} \text{sign}(\hat{\boldsymbol{n}}^t \hat{\boldsymbol{n}}_0)\hat{\boldsymbol{n}} \mapsto \hat{\boldsymbol{n}} \\ \text{sign}(\hat{\boldsymbol{u}}^t \hat{\boldsymbol{u}}_0)\hat{\boldsymbol{u}} \mapsto \hat{\boldsymbol{u}} \end{array}$$

An other way to correct the signs could be to make sure that $\hat{\boldsymbol{n}}$ is pointing downwards, or that $\hat{\boldsymbol{u}}$ is pointing upwards.

6.3 The camera position

The third column of $\boldsymbol{H}$ is defined by the following equation:

$$\begin{pmatrix} u_1 & u_2 & u_3 \\ r_1 & r_2 & r_3 \\ n_1 & n_2 & n_3 \end{pmatrix} \boldsymbol{p} = -\frac{1}{\alpha} \begin{pmatrix} h_{13} \\ h_{23} \\ h_{33} \end{pmatrix} \quad \text{or} \quad \boldsymbol{S}\boldsymbol{p} = -\frac{1}{\alpha}\boldsymbol{h}_3$$

We can thus compute $\boldsymbol{p}$ as:

$$\boldsymbol{p} = -\frac{1}{\alpha}\boldsymbol{S}^{-1}\boldsymbol{h}_3 \approx -\frac{1}{\alpha}\boldsymbol{S}^t\boldsymbol{h}_3$$

As values for the rows of $\boldsymbol{S}$ we should use the values of $\hat{\boldsymbol{n}}$ and $\hat{\boldsymbol{u}}$ obtained before correction of the signs.

The resultant estimated camera location is shown in figure 5 (right). The effect of the perspective projection matrix $\boldsymbol{H}$ on the image is shown left.

Animated sequences of these images can be found at:
`http://www.isy.liu.se/~perfo/witas/`.

Acknowledgements

The work presented in this report was supported by WITAS, the Wallenberg laboratory on Information Technology and Autonomous Systems, which is gratefully acknowledged.

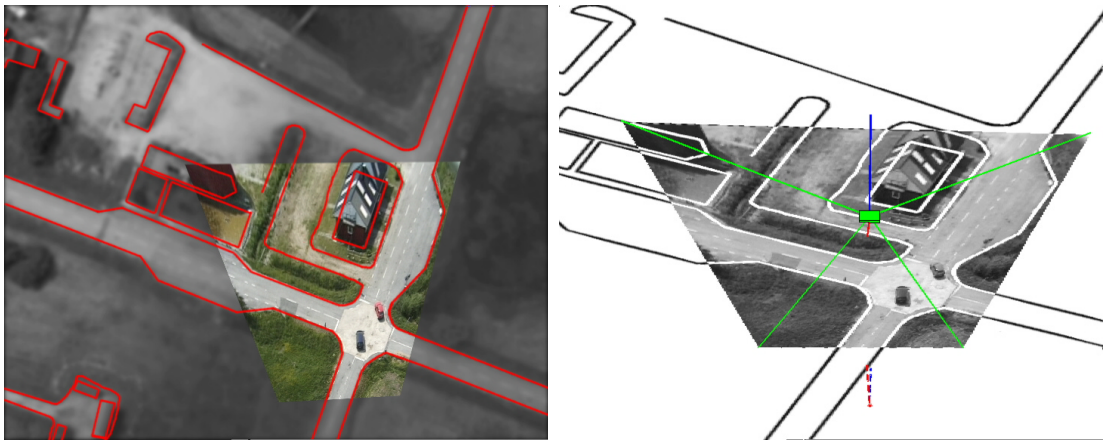

Figure 5: Perspective projection, and calculation of camera position and heading. Alignment of a frame from an image sequence to a map (left), and the corresponding calculated camera position (right).

References

[1] G. Farnebäck. Spatial Domain Methods for Orientation and Velocity Estimation. Lic. Thesis LiU-Tek-Lic-1999:13, Dept. EE, Linköping University, SE-581 83 Linköping, Sweden, March 1999. Thesis No. 755, ISBN 91-7219-441-3.

[2] G. H. Granlund and H. Knutsson. *Signal Processing for Computer Vision*. Kluwer Academic Publishers, 1995. ISBN 0-7923-9530-1.

[3] Gösta Granlund, Klas Nordberg, Johan Wiklund, Patrick Doherty, Erik Skarman, and Erik Sandewall. WITAS: An Intelligent Autonomous Aircraft Using Active Vision. In *Proceedings of the UAV 2000 International Technical Conference and Exhibition*, Paris, France, June 2000. Euro UVS.

[4] Sawhney H. S. and Kumar R. True multi-image alignment and its application to mosaicing and lens distortion correction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(3):235–243, March 1999.

[5] C.G. Harris and M. Stephens. A combined corner and edge detector. In *4th Alvey Vision Conference*, pages 147–151, September 1988.

[6] Anders Moe. Passive Aircraft Altitude Estimation using Computer Vision. Lic. Thesis LiU-Tek-Lic-2000:43, Dept. EE, Linköping University, SE-581 83 Linköping, Sweden, September 2000. Thesis No. 847, ISBN 91-7219-827-3.