HELSINKI UNIVERSITY OF TECHNOLOGY
Telecommunications Software and Multimedia Laboratory
Tik-111.500 Seminar on computer graphics
Spring 2003: jussi.hanhijarvi@tietoenator.com

Where am I?

A literature survey of single camera tracking techniques.

Jussi Hanhijärvi

22476W

Where am I?

A literature survey of single camera tracking techniques.

Jussi Hanhijärvi
Tietoenator Broadcasting Information Technology

Jussi.Hanhijarvi@tietoenator.com

Abstract

This paper describes some recent advances in determining camera position, rotation and internal parameters like focal length. Such determination finds interesting applications in broadcast studios, in augmented reality and in immersive games. Obtained position, orientation and focal length information can then be used for steering an second, computer generated virtual camera that points to virtual scenes or objects. Frames or fields of both cameras can then be mixed to form augmented reality image sequences. In this paper we survey some of the most recent papers and implementation on camera tracking techniques both in known and unknown scenes. These scenes differ from each essentially, in how tracked features are obtained and utilized. Known scenes contain artificial landmarks (fiducials) whereas unknown scenes don't. Moreover, unknown scenes contain the problem of moving objects. That is we are not sure if tracked moving points are results of moving objects or moving camera pointing to static part of the scene.

To demonstrate some features of camera tracking has one software[1] *been developed. Implementation consists of Lucas-Kanade tracking either of manually selected or of through gradient method found feature points. In addition, camera calibration with aid of checkerboard like pattern has been implemented too. For details see separate HTML documentation (in Finnish).*

1 INTRODUCTION AND PROBLEM STATEMENT

Whenever computer generated graphical objects are inserted into image sequences of film or video from a real camera it is of utmost importance that the relative positions, orientation and perspective matches each other. In TV production, chroma-keyed bluescreen techniques has been used extensively. Traditionally, cameras have fixed positions and for each, a stationary fixed virtual background (so called Virtual Sets) is rendered in advance (Doyle). Such approach is cumbersome, as cameras have no freedom to move, pan, tilt or even a possibility to change their focal lengths (zoom) without a noticeable degradation. Film industry lays new requirements for virtual

[1] Refered in this paper as *demonstration software*.

techniques. Computer generated objects can be positioned arbitrarily in the scene. They can be freely moved and rotated in 3D space as can be done with virtual cameras too. There exist a strong need in game industry to exploit so-called web-cams (small cheap, in USB port connected cameras) to build mixed reality games and scenes. One interesting application could be to use these web-cams as user interface device instead of mouse or pen tablets. Clearly there exist a need to determine camera position, orientation and focal length, based on the continuous flow of image sequences (optical flow).

Tracking, also called *position and orientation tracking* or *position tracking and mapping* is used in special mixed or augmented reality applications. Here the position and orientation of the camera or a physical object (or both) are required. In determination of the position of the camera, we need three coordinates in respect to some reference coordinate system. In addition, in many applications, the orientation of the camera or the object is an important parameter and this requires three angles known as pitch, roll and yaw. Finally in camera tracking systems, for determination of correct perspective, we have one parameter more namely, focal length, which in zoom lenses can be varied continuously. Camera internal parameters such as image plane skew or non-square shaped pixels add up to 2 - 4 parameters more. Thus, totally 9 –11 degrees of freedom (DoF) are required to fully describe orientation position and correct perspective of objects or cameras. In this paper we survey some of the recent literature that deals with camera tracking, that is computer aided determination of these parameters in 3-D space in respect to some known or unknown reference points.

During the past 20 years vital research has been performed on different tracking methods. Different sensor techniques have been proposed based on various technical scenarios, for example with the help of optical actuators (typically IR leds) located on the ceilings of studio and special sensor equipment mounted on the camera can be used to calculate position of the camera. Typical example of such equipment is HiBall Tracking System, developed by the researchers at University of North Carolina Chapell Hill and manufactured by 3rdTech[2]. – Although these techniques have shown in practice to be useful we are in this paper interested in optical tracking. The principle of optical tracking system, according Ribo, is based either on the analysis of 2 dimensional projections of image features or sweep bean angles to compute the orientation and position of given target (Ribo, 2001). Our interest is in the former one. To be more specific, we consider such optical camera tracking system that is based on the analysis of 2 D projections of optical flow from frames or fields consisting of single camera viewed scenes. These frames or fields can (partly) be used as a component in final movies. Other part of the movie can be computer generated virtual scenes or backgrounds or other objects.

This survey is organized in several chapters. In chapter 2 we discuss theory of central projection system and derive fundamental equations that are used in determination of internal and external parameters of camera. We discuss more deeply one camera calibration system proposed by Zhang (Zhang 1998). Although this is only one proposal among others we discuss it as it has been implemented in the demonstration software. In chapter 4 we discuss how unknown parameters can be

[2] HiBall tracking equipment is so small that it can be assembled in head mounted display. It is relatively accurate too, for further information see Welch 2002.

measured when position and relative orientation of some landmarks in the scene are known. In chapter 5 will tracking in unconstrained, i.e. landmark free, spaces be discussed. Here we present one method more deeply as this has been implemented in demonstration software (see separate documentation). When internal parameters of the camera have been determined and appropriate kinematic model for the movement and rotation haven selected there remain problem of actual tracking. Kalman filter techniques are very impressive in this respect. Unfortunately, for the reason of lack of the space, Kalman filtering techniques are not discussed here. Interested reader should consult excellent articles written by Welsh and Bishop (Welsh 2001) or Allen , Bishop and Welsh (Allen 2001).

2 CAMERA MODEL AND CAMERA CALIBRATION

Camera motion estimation is vital task in many computer vision implementations. In augmented reality applications, it is essential that both virtual and real cameras be calibrated using the same set of parameters, in order for the real and virtual objects to be properly aligned. These parameters are usually divided to internal (intrinsic) and external parameters. The former describes such things as focal length, picture size, aspect ratio etc. The later describes camera position and orientation in 3-D space. According to Soatto motion estimation from an image sequence is performed in two steps: first the camera is calibrated in order to build relationship between the world and the image plane coordinates. In this step are the intrinsic parameters determined. Once calibration is performed we can estimate camera motion and ambient structure in variety of ways (Soatto 1994).

Transform between two 3D coordinate systems can be described with rigid Euclidean affine transform as $\vec{m} = \vec{T}_0 + \lambda^{-1} R'_{3x3} \vec{M}$ or in extended form (Attkinson 2002):

$$\begin{bmatrix} x_C \\ y_C \\ z_C \end{bmatrix} = \begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \end{bmatrix} + \lambda^{-1} \begin{bmatrix} r_{11} & r_{21} & r_{31} \\ r_{12} & r_{22} & r_{32} \\ r_{13} & r_{23} & r_{33} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{1.1}$$

where $\vec{m} = \begin{bmatrix} x_C & y_C & z_C \end{bmatrix}^T$ is the primary and $\vec{M} = \begin{bmatrix} X\,Y\,Z \end{bmatrix}^T$ secondary coordinate system. Seven parameter define the transform: In Euclidean, orthogonal, space 3 translational movement $\vec{T}_0$ = (X_0, Y_0, Z_0), 3 rotation angles determined in R^t_{3x3} and one scaling parameter. Equation (1.1) has an inverse transform:

$$\vec{M} = \lambda R_{3x3}(\vec{m} - \vec{T}_0) \tag{1.2}$$

where $R_{3x3} = (R^t_{3x3})^{-1}$.

In central projection system, also sometimes called *pinhole camera model*, the primary coordinate system is positioned arbitrarily. The secondary coordinate system has its origin at the camera center $\vec{o}$, its z-axis coincides with principal axis (of lens system) and is directed away from image, i.e. projection, plane (see figure 1).

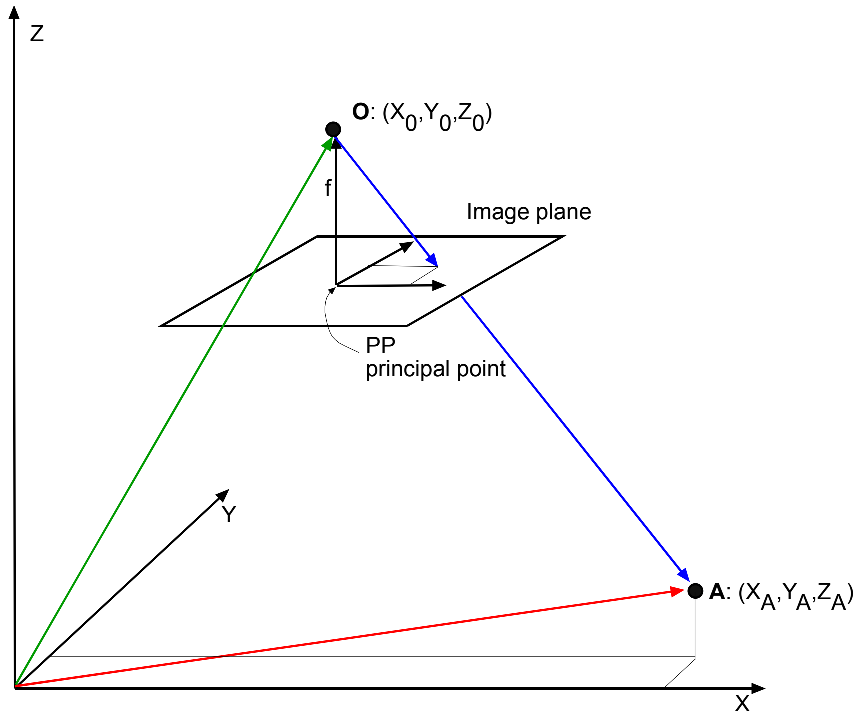


Figure 1

In the primary system we have 3 coordinates of the perspective center (X_0, Y_0, Z_0) and 3 of some point A in space (X_A, Y_A, Z_A). The projection of **A** through **O** in the secondary system give the coordinate image plane point a: (u_a, v_a, -f), where –f is the effective focal length, sometimes called *principal distance*[3]. This distance is defined as euclidean distance between image plane point PP (principal point) and perspective center O (Atkinson 2002). Now we can write the inverse of transform (1.2):

$$\begin{bmatrix} u_a \\ v_a \\ -f \end{bmatrix} = \lambda \begin{bmatrix} \gamma_{11} & \gamma_{21} & \gamma_{31} \\ \gamma_{12} & \gamma_{22} & \gamma_{32} \\ \gamma_{13} & \gamma_{23} & \gamma_{33} \end{bmatrix} \begin{bmatrix} X_A - X_O \\ Y_A - Y_O \\ Z_A - Z_O \end{bmatrix} \tag{1.3}$$

Both u_a and v_a can now easily be calculated if we know transform matrix and positions (X_A, Y_A, Z_A) and (X_O, Y_O, Z_O). These linear systems of equations are known as *Collinearity Equations*.

2.1 Lens distortion

The pinhole camera model (i.e. central perspective camera model found in (1.3)) is commonly used in camera tracking applications and research but unfortunately it is only a simplification of the optical geometry found in real cameras. Especially cheap desktop camera usually exhibits large distortion. One major deviation in the real physical lens is *radial lens distortion.* This effect causes variations in angular magnification with angle of incidence (Atkinson 2002). Second important distortion due to imperfect centering of

[3] Observe that we use notation (u,v) for image plane coordinates instead of (x,y) notation. Reason is that we want to emphasize image plane coordinate system.

the lens components in the manufacturing process, known as *tangential distortion*. Various experiments have verified that lens distortion is dominated by radial components, especially by the first terms.

According to Zheng radial distortion can be modeled in the following way (Zheng 1998). Let (u,v) be the ideal, distortion-free image coordinate and $(\breve{u},\breve{v})$ observed coordinate. Now radial distortion component can be modeled with formula:

$$\begin{aligned}\breve{u}_r &= u + u\left[k_1(u^2+v^2)+k_2(u^2+v^2)^2\right]\\ \breve{v}_r &= v + v\left[k_1(u^2+v^2)+k_2(u^2+v^2)^2\right]\end{aligned} \qquad (1.4)$$

and tangential distortion component with formula:

$$\begin{aligned}\breve{u}_t &= 2p_1uv + p_2[(u^2+v^2)+2u^2]\\ \breve{v}_t &= 2p_2uv + p_1[(u^2+v^2)+2v^2]\end{aligned} \qquad (1.5)$$

In the demonstration software are these parameters (k_1, k_2, p_1, p_2) calculated. Total distortion can be calculated as sum of radial and tangential distortion.

These important distortion topics will not be discussed further in this tutorial paper. Interested reader should consult for example research paper written by Frédéric Deverney and Faugeras (Deverney).

2.2 Camera calibration

It is important to note that perspective model (1.3) maps three-dimensional space $\mathbb{R}^3$ to perspective plane $\mathbb{R}^2$ whose coordinates can be extended in homogenous system, that is they are defined only up to nonzero scale factor of vector $(x,y,z)^T$. Since we can select freely origin of coordinate system can equation (1.3) be written in more convenient form using homogenous coordinate system for each frame k = 1,...,n in video sequence (Romano 2002). Also we use notation $(u(k), v(k))^T$ for image coordinate in frame k and for the world coordinate X_A-X_0 etc. we use notation $(X(k), Y(k), Z(k))^T$. Thus we can rewrite equation (1.3) in form with addition of small scaling factor s (as described in many elementary photogrametry or computer graphics textbooks):

$$s\begin{bmatrix}u(k)\\ v(k)\\ 1\end{bmatrix} = s\vec{m}(k) \simeq P_{3x4}\vec{M}(k) = \begin{bmatrix}p_{11} & p_{21} & p_{31} & p_{41}\\ p_{12} & p_{22} & p_{32} & p_{42}\\ p_{13} & p_{23} & p_{33} & p_{43}\end{bmatrix}\begin{bmatrix}X(k)\\ Y(k)\\ Z(k)\\ 1\end{bmatrix} \qquad (1.6)$$

where $\vec{m}$ denotes projective plane (image plane) coordinate and $\vec{M}$ world coordinate. Camera projection matrix P_{3x4} is an 3 x 4 homogenous matrix and can be easily factored into its internal or intrisic matrix A (i.e. focal distance, pixel size etc.) and into external rotation matrix R and translational vector T:

$$P_{3x4} = A\left[R\,,T\right] \Rightarrow s\vec{m} = A\left[R,T\right]\vec{M} \qquad (1.7)$$

In many papers has the intrisic properties of the camera been modeled by the affine transformation (see for example Zhang 1998, Foroosh 2002, Romano 2002):

$$A = \begin{bmatrix} f_x & d & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{1.8}$$

This model (1.8) depict 5 camera‘s internal (intrisic) parameters:

- f_x, f_y, the focal length measured, commonly, in pixel unit along the coordinate at image plane. In some camera these can be distinct when vertical and horizontal sampling rate differs, i.e. when pixel size is not square shaped.
- $d = \cos(\phi)$ models the angle ϕ between image planes coordinate axis.
- The intersection of the camera’s optical axis with the image plane is defined as coordinate $(c_x, c_y)^T$, these coordinate define position of camera’s principal point PP. Normally this point lies in the center of image.

Both the rotational and translational term in (1.7) have 3 degrees of freedom and the intrisic matrix A has 5. Therefore, for the camera configuration we have 11 degrees of freedom. In practice the coordinate axis in the image plane are perpendicular (orthogonal) so d = 0, focal lengths are equally scaled so $f_x = f_y$. Thus configuration task of the camera in many practical situations reduces to 9 degrees of freedom.

In order to determine parameters in (1.7) we usually make some measurements on the image plane using multiple frames or multiple cameras. Here world coordinate $\vec{M}$ is either known or unknown. In the later case we continuously refine initial guess. For each frame k=1,...,n equation (1.6) can be solved:

$$u(k) = \frac{p_{11}X(k) + p_{21}Y(k) + p_{31}Z(k) + p_{41}}{p_{13}X(k) + p_{23}Y(k) + p_{33}Z(k)}$$

$$\tag{1.9}$$

$$v(k) = \frac{p_{12}X(k) + p_{22}Y(k) + p_{32}Z(k) + p_{42}}{p_{31}X(k) + p_{32}Y(k) + p_{33}Z(k)}$$

When distinct views of the scene grows beyond 3 then system (1.9) will be overconstrained. It is ironic to observe that when number of measurements increases then (1.9) come in many instances less accurate.

In classical photogrametry there exist many methods to calibrate camera. For example:

- DLT, (Direct Linear Transformation method) originally reported by Abdel-Aziz and Karara. The DLT method uses a set of control points whose object space versus image plane coordinates are known prior to calibration process. Usiually control points are fixed on some rigid frame or body, calibration frame. Flexibility of the method depends solely on how easy it is to handle the calibration frame. When large area is needed the typical rigid frame approach doesn’t work. For example in one DLT derivative (Survey method) is different size control areas with varying number of control points used.
- In least square methods are fixed control points used too. Here are used different weights for different calibration points. Method is iterative too.

In augmented reality application these classical methods are seldom used. According Hassan Foroosh lecture notes possible approach to solve this system of equations (1.9) could be based on linear regression analysis, non linear optimization, structure from motion, using vanishing points or multiple planar pattern (Foroosh 2002). From these methods we discuss more deeply one method that is based on multiple planar pattern (chapter 3.3) as this is implemented in the demonstration software.

In linear regression (least squares) analysis we solve a set of linear equations (1.9) for several frames and use linear regression analysis to approximate the unknown parameters or to minimize error term.

In non–linear optimization measurements are interpreted as a result of random process. That is measurements (u(k), v(k)) in (1.9) are known with some probability. Unfortunately probability density function is not known in many practical situations. It must be estimated from available data. However it is reasonable to assume that the type of density function is Gausssian. We can now write measurement equations (1.9) in form:

$$\begin{aligned} u(k) &= g(P_{3x4}, t_i) + n_i = \hat{u}(k) + n_i,\, n_i \sim N(\mu_u, \sigma_u) \\ v(k) &= g(P_{3x4}, t_i) + m_i = \hat{v}(k) + m_i,\, m_i \sim N(\mu_v, \sigma_v) \end{aligned} \tag{1.10}$$

where N is Gaussian distribution with some (unknown) mean and variance. Clearly probability of u(k) is Bayesian, that is probability of u(k) take some value given that point in reference frame takes value $\hat{u}(k)$, also we calculate conditional probabilities $p(u(k) \mid \hat{u}(k))$. Our goal is now to estimate unknown variables in density function L, given that random samples (measurements) are drawn from pdf L. Assuming statistical independence between samples we can estimate likelihood as[4]:

$$L = \prod_i p(u(k) \mid \hat{u}(k)) p(v(k) \mid \hat{v}(k)) = \prod_i e^{-\frac{(u(k)-\hat{u}(k))^2}{\sigma^2}} e^{-\frac{(v(k)-\hat{v}(k))^2}{\sigma^2}} \tag{1.11}$$

Taking logarithm in (1.11) we obtain (log likelihood):

$$C = -\log L = \sum_i \frac{(u(k)-\hat{u}(k))^2}{\sigma^2} + \sum_i \frac{(v(k)-\hat{v}(k))}{\sigma^2} \tag{1.12}$$

For optimization of C there exist several methods for example non-linear regression analysis or Levenberg-Marquardt optimization (for details see for example Foroosh).

Urban & al. solve the problem of camera intrinsic parameter recovery from multiple views (Urban 1998). Their approach doesn't require any previous knowledge of the scene structure. The only used assumption is that the frames are taken with the same camera, i.e. intrinsic parameter are constant. They use simple pinhole camera model that has similar projective geometry as depicted above.

In the camera model matrix A (1.8) was 5 parameters. Determining these parameters is known as camera calibration task. It is known that having three distinct views of the scene, calibration task leads to an overconstrained system of polynomial equations called Kruppa equations (Sturm). These equations can be solved using fundamental

[4] Interested reader should consult some elementary textbook on probability and stochastic processess.

matrices. In their paper Urban et al give a novel approach to solve these equations. In this approach are used several frames (k=1,…,n) and j=1,…,m point correspondences between 2-D image points and 3-D scene points[5]

$$s_j(k)\begin{bmatrix} u_j(k) \\ v_j(k) \\ 1 \end{bmatrix} = A(k)(R(k)\begin{bmatrix} X_j \\ Y_j \\ Z_j \end{bmatrix} + T(k)) = P(k)\begin{bmatrix} X_i \\ Y_j \\ Z_j \\ 1 \end{bmatrix} \tag{1.13}$$

The calibration task is performed in two steps. First one is a projective reconstruction and the second one is proper evaluation of intrinsic parameters. Projective reconstruction consist of recovery of projective matrices $\hat{P}_i(k)$ and points $\hat{S}_j$ calculated from points $(u_j(k), v_j(k))$. These recovered $\hat{P}_i(k)$ and $\hat{S}_j$ differ from real (actual) $P(k), S_j$ by an unknown 4 x 4 transform matrix H i.e.

$$\begin{aligned} P(k) &\simeq \hat{P}(k)H \\ S_j &\simeq H^{-1}\hat{S}_j \end{aligned} \tag{1.14}$$

In the second step, if the number of frames (images) is equal or greater than 3 the system of equations in (1.14) and the definition of P_i

$$P(k) = A\,[R(k), T(k)] \tag{1.15}$$

gives sufficient constraints for intrinsic matrix A. Their approach have good numerical behavior, as was evidenced with large experimental test. Since they use at least 3 frames their solution ambiguous for any surfaces and coherence of views is assured (Urban 1998).

2.3 Calibration using multiple planar patterns

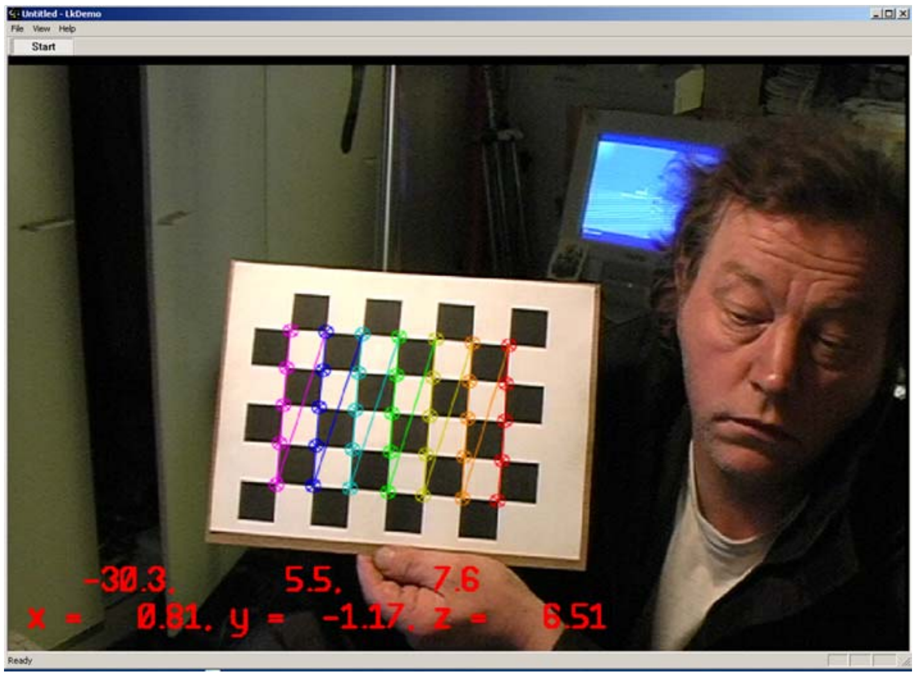
Zhang propose a new approach to calibrate camera using multiple planar patterns (Zhang 1998). His method has been implemented in accompanied demonstration software. Method is widely used in computer vision community and has been implemented in Intel's open Computer Vision library and in several openly available MATLAB routines. The technique only requires that the camera observe one, freely in 3D space moving, planar pattern. The motion in space need not be known. Used pattern is usually checkerboard like picture (see figure 2).

Zhang's method is performed in five steps:

- Take few images of the model plane (in practice 5 –7 different orientations gives sufficient accuracy).
- Detect the feature points of the images (usually corners of the squares).
- Estimate the five intrisic parameters and all the extrisinc parameters using closed

[5] Observe that every frame and measurement point is multiplied with a weighting factor $s_j(k)$.

form solutions (for details see below).
- Estimate the radial and tangential distortion factor by linear least square approximation.
- Refine all calculated parameters with minimizing maximum likelihood estimation of used parameters (intrisic and rotation matrix, translational vector and distortion coefficients).



```
-30.3,      5.5,      7.6
x =   0.81, y =   -1.17, z =   6.51
```

Figure 2: Used checkerboard pattern in Zhang camera calibration implementation. Red numbers in the bottom area of the picture describes rotation and translation of checkerboard.

When camera is calibrated it is relatively straightforward to calculate relative rotation and translation of used planar pattern. That is camera or pattern can be moved and rotated freely and extrinsic parameters can be calculated assumed that camera "see" the pattern.

2.3.1 *Solving the intrisic parameters*[6]

Without loss of generality we can assume that our pattern (model plane) is located at Z = 0 in the world coordinate system. Thus we can rewrite equation (1.7) (for simplicity we have dropped frame indices away):

$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = A[r_1\ r_2\ r_3\ t]\begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix} = A\left[r_1\ r_2\ t\right]\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \qquad (1.16)$$

where r_i is i:th column vector in rotation matrix and t is the translational vector.

[6] Topics in this chapter is based on Zhang's excellent and very readable paper (Zhang 1998).

Observe that we can still use our previous notation $\vec{M}$ for world coordinate (as Z = 0). Image point $\vec{m}$ is now related to model point $\vec{M}$ with the 3 x 3 homography matrix H:

$$s\vec{m} = H\vec{M},\, H = A\left[r_1\; r_2\; t\right] \tag{1.17}$$

where s is some scaling factor.

When an image from model plane is obtained Zhang recommend to estimate homography H as a maximum likelihood estimate This estimate is calculated with Levenberg-Marquardt optimization procedure (see above). - From equation (1.17) we have:

$$H = [h_1\; h_2\; h_3] = \lambda A[r_1\; r_2\; t] \tag{1.18}$$

where λ is arbitrary constant and h_i i:th column vector of H. As r_1 and r_2 are orthonormal we can calculate two basic constraints, given one homography:

$$h_1^T A^{-T} A^{-1} h_1 = h_2^T A^{-T} A^{-1} h_2 \tag{1.20}$$

$$h_1^T A^{-T} A^{-1} h_2 = 0 \tag{1.19}$$

where $A^{-T} = (A^T)^{-1}$or $(A^{-1})^T$.

Let us now define symmetric matrix B:

$$B = A^{-T} A^{-1} = \begin{bmatrix} B_{11} & B_{12} & B_{13} \\ B_{12} & B_{22} & B_{11} \\ B_{13} & B_{11} & B_{33} \end{bmatrix} \tag{1.21}$$

B is defined by 6D vector:

$$b = [B_{11}\; B_{12}\; B_{22}\; B_{13}\; B_{23}\; B_{33}]^T \tag{1.22}$$

Now we have:

$$\begin{aligned} & h_i^T B h_j = v_{ij}^T b \\ & \Rightarrow \\ & v_{ij} = [h_{i1}h_{j1},\, h_{i1}h_{j2} + h_{i2}h_{j1},\, h_{i2}h_{j2}, \\ & \qquad h_{i3}h_{j1} + h_{i1}h_{j3},\, h_{i3}h_{j2} + h_{i2}h_{j3},\, h_{i3}h_{j3}]^T \end{aligned} \tag{1.23}$$

where column vector $h_i = [h_{i1}\; h_{i2}\; h_{i3}]^T$.

Now constraints (1.19) and (1.20) for a given homography can be rewritten in form:

$$\begin{bmatrix} v_{12}^T \\ (v_{11} - v_{22})T \end{bmatrix} b = 0 \tag{1.24}$$

If n images of the model pattern are observed then by combining such equations as (1.24) we get equation:

$$Vb = 0 \tag{1.25}$$

where V is 2n x 6 matrix. If $n \geq 3$ we will have unique solution for b. Without great difficulty we can now calculate the camera intrisic parameters (have a look at equation (1.8) too), translational movements and rotation (Zhang 1998):

Parameter	Equation
Principal Point	$c_x = \frac{dc_x}{f_y} - \frac{B_{13} f_x^2}{\lambda}$
	$c_y = \frac{B_{12}B_{13} - B_{11}B_{23}}{B_{11}B_{22} - B_{12}^2}$
Focal length	$f_x = \sqrt{\frac{\lambda}{B_{11}}}$
	$f_y = \sqrt{\frac{\lambda B_{11}}{B_{11}B_{22} - B_{12}^2}}$
Coordinate axis skew	$d = -\frac{B_{12} f_x^2 f_y}{\lambda}$
Scale (ref. (1.18))	$\lambda = B_{33} - \frac{B_{13}^2 + c_x (B_{12}B_{13} - B_{11}B_{23})}{B_{11}}$
Translational movement	$t = \lambda A^{-1} h_3$
Rotation	$r_1 = \lambda A^{-1} h_1$
	$r_2 = \lambda A^{-1} h_2$
	$r_3 = r_1 \times r_2$

2.4 Furher reading

Further reading can be found for example in papers written by Soatto et al, Quan, Criminisi et al and Simon et al. - Soatto et al describes a recursive method for estimating the motion and structure of the scene from a sequence of images taken with a camera whose intrinsic parameters are unknown. Method is based on recursive motion estimation scheme, called the “essential filter”. Quan describe a new approach to estimate camera motion. His approach is derived with affine camera model, which is more general class of projection including orthographic, weak perspective and para-perspective projection models (Quan 1996). His model needs only 3 intrinsic parameters at most. Criminisi, Reid and Zisserman describe a method how 3D measurements may be computed from single perspective view of a scene given minimal set of geometric information from the image. This minimal information is typically the vanishing line of reference plane and vanishing point to the direction not parallel to the plane (Crimnisi). The main advantage of their method is that affine scene structure is determined from the image without prior knowledge of the camera’s intrisic parameters. Idea for their inventions is originated on the rules for drawing perspective pictures given by Italian Renesaince artist Leon Battista Alberti. G. Simon and M-O. Berger describes an two stage efficient algorithm for handling zoom changes in video sequences. In their approach video is partitioned into camera motions and zoom variations. In the later one camera focal length parameters can be adjusted (Simon).

3 TRACKING IN KNOWN SPACES

In chapter 2 we discussed mathematical methods on determination of parameters for the central projection system. In this chapter we referee some published papers which describes how these parameters are obtained in practice for scenes that contains known landmarks (other than checkerboards).

Dieter Koller & al. use an automated method to calibrate the internal as well the external parameters of a camera (Koller 1997). They use uniformly colored dark squares on the walls of a room. For the determination of internal parameters (pixel size focal length and focal center (f_x, f_y, c_x, c_y)) they use pinhole camera model with no lens distortion. Internal parameters are fixed for the whole session. They claims that lens distortions (for example fish eye-ball aberrations) can't be taken into account since workstation's graphic pipeline for display does not allow this to be rendered correctly. Their approach consists of two phases, initialization and tracking. In the first one is rectangular patterns sought. In the second phase corner points of these patterns is continuously monitored. They use Kalman filter techniques in tracking phase.

In the initialization phase a watershed algorithm is used for the finding of a dark square shaped pattern (*blob*), used as reference point. Watershed algorithm is a morphological operation that decomposes the whole image in smaller connected regions (*paddles*). Using such transform a dark blob surrounded by relatively large bright area gives a strong filter response related to the (numerical) deepness of the paddle. The deepest, most compact and most promising are then matched against known 3-D squares. To identify the right paddle the squares contains one or more small red squares. Positions of these small squares in the larger dark square represents binary coding of the paddle identification number (Koller 1997).

To model the movements of camera Koller& al. use acceleration free motion model with 6 degrees of freedom. It is well known (from elementary kinematics) that in this case general motion can be decomposed to constant translational velocity v_c at the center of mass and rotation with constant angular velocity ω around the axis through the center of mass. Motion of point p is then given by first derivative:

$$\frac{dp}{dt} = v_c + \omega \times (p - c) \qquad (1.26)$$

Since center of mass c itself is moving then the center of rotation is moving too. Consequently, constant rotation and translation is no longer constant with respect to camera coordinates (Koller 1997). If we substitute $c(t) = c(t_0) + v_c(t - t_0)$ in (1.26) we obtain motion equations where rotation is in respect to world coordinates:

$$\frac{dp}{dt} = v + \omega \times p + at$$
$$v(t_0) = v_c - \omega \times c(t_0) \qquad (1.27)$$
$$a = -\omega \times v_c$$

Calibration and registering of camera is a stationary process. Goal for this is to allow dynamic scene changes i.e. tracking of camera. After the initialization phase, discussed above, can tracking begin. For the tracking Koller&al uses corners of reference squares which already have been used in the initial calibration phase. They keep in this phase internal camera parameters constant, thus their model doesn't allow changing of focal length i.e. zooming. For tracking they use extended Kalman filter techniques for optimal motion estimation.

Wisskirchen & al. and Kansy & al describes an image based tracking system (see figure 2). They use for the calibration of camera precisely measured real world reference points located in a planar surface (Wisskirchen 1996, Kansy 1995). These points are identified interactively in the first frames and used as reference points in subsequent tracking. They use a camera model, which is characterized with four parameter pan, tilt, roll and zoom.

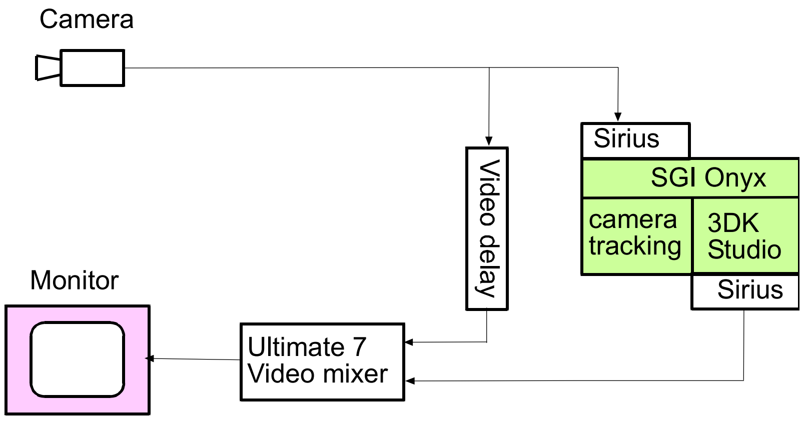


Figure 3

In their scheme, the camera transfers video signal to the Sirius video capture board attached on a SGI Onyx. One processor in Onyx process the images in video sequence, i.e. calculate camera's translational and rotational position relative to reference points. The calculated camera parameters are then used in separate rendering process for generation of the synthetic scene. This processing has a delay up to 160 ms. Therefore before mixing original video and synthetic frames must video signal be delayed in separate delay unit.

In their model camera parameters (focal length, pan, tilt, roll and zoom) are calculated from four or more reference points (Wisskirchen 1996, Kansy 1995). They use popular *view correlation*- algorithm discovered by Rod Boggart (Boggart 1991). Algorithm is characterized with the following steps:

1. Find the error with current camera parameters

2. Calculate partial derivative for image point u, v with respect of camera parameters (position, orientation, field of view, aspect ratio, center of image), populate Jacobian matrix J with these derivatives.
3. Invert this Jacobian matrix in order to find correction values. Since used Jacobian matrix is not square the pseudo inverse is calculated: $J^{-1} = J^T (JJ^T)^{-1}$
4. Apply these values to current camera parameters.

In order to achieve stable behavior for camera tracking Weisskirchen et al and Kansy et al use Kalman filtering techniques.

ORAD has developed a commercial product for camera tracking. Their product is mainly used in a blue-screen based broadcast studio. They use a grid of light blue stripes painted on a normal blue-screen background canvas (see figure 3). Although no detailed description on their system was found, then apparently in some pre-phase in video processing chain will edges sought from the grid (figure 3 left). Based on this "edge enhanced grid" will actual camera tracing be performed. Orad claims that camera must "see" only few lines of grid crossing. They use a special purpose signal processor (DSP) for analyzing grid pattern and for calculating camera parameters.

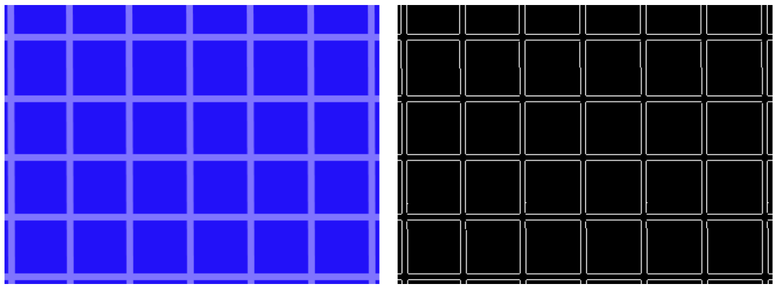


Figure 4: ORAD bluescreen grid, on the left edge view of the grid

Trogeman & al describes a camera tracking system, named as *Trick_track.* For determination of intrisic camera parameter was Boggart's view correlation algorithm used. They use especially colored square shaped reference points (markers). These are interactively selected during initializing phase (Trogeman 1998)). For each reference point a characteristic color interval is defined by measuring the four sides of the markers and their environments. During tracking are these markers sought. They use in prediction for marker's next position the following information (Trogeman 1998):

- The last shift of the marker: prediction for the marker position in the next frame is calculated by subtracting from the screen coordinates of marker's previous position with current position of the markers.
- The current movement trend is calculated based on the positions and the diameters of markers on several frames. They calculate regression line for the last five positions and nine diameters.
- The movements of other markers.

With the help of position information described above actual search range for reference points are largely reduced. Finally inter-frame displacement could be determined.

4 TRACKING IN UNKNOWN SPACES

In previous chapter we discussed camera calibration and tracking when some well known information is available. In this chapter we discus problem how tracking information can be obtained when such knowledge is not at our hands. Cameras must be self-calibrated based on the information obtained by movements in more or less stationary scene. In this chapter we discuss four topics: firstly the question how feature points should be selected, secondly we describe some fundamental topics on movement estimation, thirdly we unravel some practical implementations, finally we describe one implementation based on Lucas Kanade pyramid tracker. In the accompanied software is Lucas-Kanade tracker implemented.

4.1 Feature point selection

No feature based tracking system can work unless good features can be identified and tracked from frame to frame. Tracking in unknown spaces encounters new problems. As some part of the image contains no motion information at all or it can be misleading. A general rule of thump is that features should be selected on area where texture variations are high. Researchers have proposed to track corners, windows with high spatial frequency content or second order derivative is sufficient high. Once feature points has been selected there remain problem if these are adequate during tracking. For example, the selected point is not fixed or it can straddle discontinuity (between frames) or it can be at the boundary of a glossy surface where it can move when lightning conditions changes. Feature points can be occluded resulting drifts in tracker etc.

Jean-Yves Bouget give a practical algorithm for feature selection (Bouguet). Method is based on finding gradients for every pixels in the (initial) image. One central step in tracking is the calculation of optical flow. At that step a matrix G is formed around every pixel in the image. This matrix is required to be invertible, or in other words, the minimum eigenvalue of G must be large enough (Bouguet). Method is originally used for tracker that is based on n level pyramidal image composition but it can be used in a flat (ie. 1-level) tracker too. In the pyramid representation of some image are subsequent images calculated in recursive fashion by sub sampling with lower resolutions (see below). Bouguet selection process goes as follows:

1. For every pixel in the initial frame and in the derived sub images (at level L) calculate derivative for x and y direction:

$$\begin{aligned} I_x(x,y) &= \frac{I^L(x+1,y)-I^L(x-1,y)}{2} \\ I_y(x,y) &= \frac{I^L(x,y+1)-I^L(x,y-1)}{2} \end{aligned} \qquad (1.28)$$

   where $I^L(x,y)$ is the pixel value at location (x,y) and at level L. Then calculate the spatial gradient matrix G(x,y) for every pixel

$$G(x,y)=\sum_{x=p_x-\omega_x}^{p_x-\omega_x}\sum_{y=p_y-\omega_y}^{p_y-\omega_y}\begin{bmatrix} I_x^2(x,y) & I_x(x,y)I_y(x,y) \\ I_x(x,y)I_y(x,y) & I_y^2(x,y)\end{bmatrix} \tag{1.29}$$

Where ω_x and ω_y define some window size[7]. Calculate minimum eigenvalue λ_m for G(x,y)

2. Select the maximum value λ_{maks} from all λ_m.
3. Retain those image pixels that have λ_m value larger than some percentage of λ_{maks} for example 5 % or 10 %
4. Keep the local maximum pixels, that is pixels whose λ_m are greater than any other in its (3 x 3) neighborhood.
5. Finally, keep only those pixels that the minimum distance between any pair of pixels is larger than some threshold distance (usually 5 or 10 pixels).

4.2 Movement estimation

In general, two questions must be answered in order tracking to be successful in previously unknown scenes: first how to select monitored features and how to track them from frame to frame. As the camera moves, patterns in the images changes in complex way. These changes can be described as image motions in small windows (integration windows) that should be far away from occluding boundaries and near surface markings[8]:

$$I(x,y,t+\Delta t)=I(x-\xi(x,y,t,\Delta t),y-\eta(x,y,t,\Delta t)) \tag{1.30}$$

Equation (1.30) is nothing more than observation that image taken at a time $t+\Delta t$ is obtained by moving every point (pixel) of the current image (at time t) by an amount of displacement $\delta=(\xi,\eta)$. Shi & al notices that noticeable degradations of the displacement is sometimes a function of image position and variations in δ (Shi 1994, Tomasi 1991). Even with small window size there can exist different displacements. Thus an *affine motion field model* is a better representation for this movement:

$$\begin{gathered}\delta = D\vec{x}+\vec{d} \\ \textit{where} \\ D=\begin{bmatrix} d_{xx} & d_{xy} \\ d_{yx} & d_{yy}\end{bmatrix}\end{gathered} \tag{1.31}$$

In (1.31) the matrix D (*deformation matrix*) describes motion deformations in directions xx, xy, yx and yy, $\vec{d}$ describes displacement for some window center. Thus we can write motion equation (1.30) in more general form:

[7] This window size is then used in subsequent tracking phase. Window is refereed *as integration window*.

[8] In chapter 4.1 described selection algorithm doesn't guarantee non-occlusion.

$$J(\mathbf{S}\vec{x}+\vec{d})=J((\mathbf{1}+D)\vec{x}+\vec{d})=I(\vec{x}) \tag{1.32}$$

where J is the second image. In pure translational movement D = 0 and displacement $\delta=\vec{d}$.

The problem to find camera motion parameters is then to find parameters S and d that minimizes the *dissimilarity* (rms residual) between two image:

$$\varepsilon=\iint_W [J(\mathbf{S}\vec{x}+\vec{d})-I(\vec{x})]^2 w(\vec{x})d\vec{x} \tag{1.33}$$

where W is the feature, integration, window, $w(\vec{x})$ is a weighting function, I is image at the position $\vec{x}$, J is the second image at position $\mathbf{S}\vec{x}+\vec{d}$. When central area of the window is emphasized then $w(\vec{x})$ could be spatial Gaussian-like function. To obtain the camera movements we minimize equation (1.33). Also we differentiate this equation with respect of unknown entries in the deformation matrix D and the displacement $\vec{d}$ and set the result to zero. Finally we linearize the results with a truncated Taylor expansion (Shi 1994):

$$J(\mathbf{S}\vec{x}+\vec{d})=J(\vec{x})+g^T(u) \tag{1.34}$$

As a result of the calculations we get:

$$\mathbf{T}\vec{z}=\vec{a} \tag{1.35}$$

where the vector $\vec{z}$ contains the values in deformation matrix and displacement i.e. $z^T=\begin{bmatrix} d_{xx} & d_{yx} & d_{xy} & d_{yy} & d_x & d_y \end{bmatrix}$ $\vec{a}$ is an error vector and T is a 6 x 6 matrix that can be computed from the image (for details see Shi 1994 or Tomasi 1991):

$$T=\iint_W \begin{bmatrix} U & V \\ V^T & Z \end{bmatrix} wdx \tag{1.36}$$

When movement is pure translational and our goal is to calculate only the displacement then smaller system $\mathbf{Z}\vec{d}=\vec{e}$ should be solved (Tomasi 1991):

$$\begin{aligned} &\mathbf{Z}d=\vec{e} \\ &\mathbf{Z}=\iint_W gg^T w(\vec{x})dW \end{aligned} \tag{1.37}$$

where g is coefficient in truncated Taylor expansion (1.34). For the error term e is calculated :

$$\vec{e}=\iint_W (J(\vec{x})-I(\vec{x}))gwdW \tag{1.38}$$

Shi and Tomasi developed an algorithm, where quality of image features is monitored during tracking. They use measure of feature dissimilarity that quantifies the change of appearance between the first and current frame (Shi 1994). Idea is relatively simple: dissimilarity is the feature's rms residue between the first and current frame (equation (1.33)). *When the dissimilarity grows too large should the feature be abandoned.* They provide experimental evidence that pure translation is not adequate model but affine image changes like warping and translation are satisfactory.

A feature with high texture variation can still be bad feature to track. For example, in an image of some tree, a horizontal bough can intersect with a bough in the background. Intersection occurs only in the image, since the two boughs are in different depths. Dissimilarity (1.33) can often indicate that something is going wrong. While inter-frame changes are small for the pure translational tracker to work the cumulative changes over multiple frames can be rather large and dissimilarity measure (1.33) grows quickly. To overcome this problem Shi and Tomasi developed a method to optimize tracker's accuracy (for details see Shi 1994). They claim that the best features are exactly those that make the tracker to work best.

4.3 Practical implementations

Azarbayejani & al present a method that can be used in recursive estimation of camera motion, description of pointwise structure of the scene and determination of focal length. These features are tracked through image sequence. Above all, what separates their method from the previous (in 1994) is that it is formulated to obtain the focal length variations during tracking. However they assume that the focal length is similar for both direction in the image plane and the image plane coordinates are mutually perpendicular i.e. pixel aspect ratio is nearly square. In practice, these assumptions are reasonable. Their algorithm applies uniformly to both true perspective and orthographic projections. They use extended Kalman filtering techniques as the computational framework. They don't need a calibrated camera and as a camera model they use perspective pinhole camera, without lens distortion, such as described in chapter 2 . They develop a new model for the scene structure, both for the translational and rotational movement. From these models they build a state vector $\vec{s}$ that contains 7 + N parameter, six for the motion one for the camera geometry and N for structure i.e. number of features that are used for tracking:

$$\vec{s} = \begin{bmatrix} t_x & t_y & t_z & \omega_x & \omega_y & \omega_z & f & a_1 & \dots & a_N \end{bmatrix} \tag{1.39}$$

number of parameters grows by 1 if quaternion is required. In their model $\vec{s}$ is the state vector used in standard Kalman filter implementation. In this implementation the measurement vectors contains the image locations of all tracked feature windows in a new frame. In the filter dynamics a model of identity transform + noise was used. Measurement equations are the combination of used central projection model, scene structure model and translation model. For the experiments EKF implementation was straightforward and standard, where only quaternion maintenance was added. Computationally it requires inverting 2(7+N) x 2(7+N) matrix (size of the state vector). They claims that N rarely needs to be larger than 15 or 20 (relatively small amount feature points).

Yao and Calaway describes a system, witch can be used in camera motion estimation in 3 D spaces and depth estimation (Yao 2002). In their system they have no prior knowledge of the camera parameters or any assumption of the camera movements. However, they assume that scene should contain only stationary rigid objects (like buildings, furniture etc.). In addition, they assume that motion in the image sequence was generated by rigid camera motion in 3-D space. Their work is based on algorithms founded by Azarbayejani and Pentland. Yao & al .use similar central projection model for the camera as described in chapter 2. Image points x_i (k) at frame k are obtained by tracking key features in the scene across the sequence (k = 1,…, n) and then an extended Kalman filter is applied to obtain estimates of the motion and scene structure parameters i.e. rotation R(k), translation T(k) and depth parameter for each tracked point in the selected reference frame. Essential component in their algorithm is that it gives estimate for the focal length and for the translational motion and depth. Thus their algorithm allows to be used without pre-calibration of camera (Yao 2002). To obtain “good” 2-D point they need to walk through the image sequence. They use “off the self solution” for the feature tracker, that is KLT tracker invented by Pollefeys & al. They report that a computational requirement is small for Kalman filter and in principle it could be implemented in real time. The main burden in computational complexity is computing the inverse of covariance matrix that is used in the Kalman filter. This covariance matrix has a size of $(2N)^2$, where N is the number of feature points tracked. In their experiments, the filter took 0.05 s per frame with 18 points and 0.65 with 48 points. Experiments was performed on SGI Octane with non optimized code.

4.4 Lucas Kanade implementation

In the demonstration software was a Lucas-Kanade tracker implemented. Tracker is based on pyramidal image decomposition, where movement vectors are sought in every subsequent image at corresponding pyramid level as suggested by Bouguet (Bouguet). In the implementation the tracked features was selected according gradient method described in chapter 4.1.

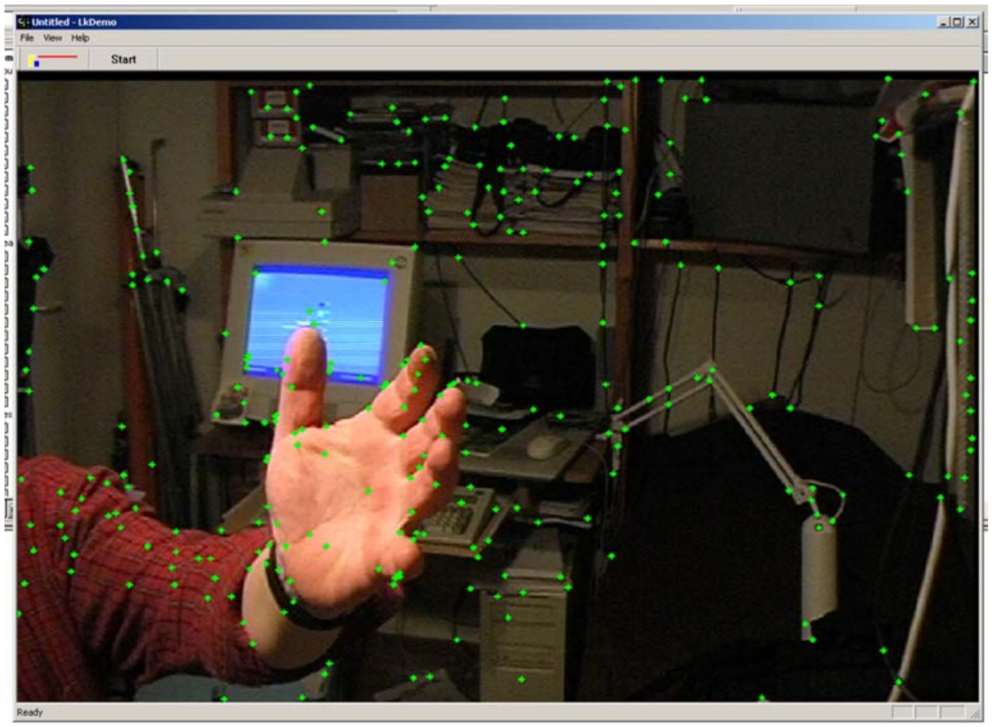


Figure 5: Implementation of Lucas Kanade pyramid tracker. Green dots represents selected tracking points. It can be clearly seen that these are located on places where gradients are relatively large.

As was noted previously so the goal of tracking is to search in the second image such locations for every feature point connected small window area (integration window) that minimizes dissimilarity between windows and thus dissimilarity between images (equation *(1.33)*). If we now neglect deformation matrix and weighting function, we can rewrite dissimilarity equation *(1.33)* in discreet form:

$$\varepsilon(d) = \varepsilon(d_x, d_y) = \sum_{x=u_x-\omega_x}^{u_x+\omega_x} \sum_{y=v_y-\omega_y}^{v_y+\omega_y} (I(x,y) - J(x+d_x, y+d_y))^2 \qquad (1.40)$$

where I is the first and J is the second image, d is the displacement for x and y directions. Dissimilarity is measured on the feature point neighborhood, i.e. integration window is of size $(2\omega_x+1)\times(2\omega_y+1)$.

In any feature tracker two key components are robustness and accuracy. As accuracy relates to local sub-pixel accuracy then intuitively small integration window would be preferable. Robustness component relates to sensitivity of tracking with respects to size of motion and sensitivity to changes of lightning. Intuitively it is preferable to use a large integration window. Considering equation (1.40) it is preferable that motions are smaller than window size $d_x < \omega_x$ *and* $d_y < \omega_y$ thus when we try to track large displacement should the window size be large. It is clear that these two requirements are conflicting. To solve this problem we could use several images derived from base image for example by using sub sampling and arange them in a pyramid pattern..

Let us now denote the original, zero level, image as I^0 = I. Now the pyramid representation is build in recursive fashion: compute image 1 from image 0, image 2 from image 1 and so on. Let L = 1,2, 3… be pyramid level and image I^{L-1} is the image at level L-1. In the implementation image composition is calculated as follows with anti aliased low passfilter :

$$
\begin{aligned}
I^{L}(x,y) = {} & \frac{1}{4} I^{L-1}(2x,2y) + \\
& \frac{1}{8}[I^{L-1}(2x-1,2y) + I^{L-1}(2x+1,2y) + I^{L-1}(2x,2y-1) + I^{L-1}(2x,2y+1)] + \\
& \frac{1}{16}[I^{L-1}(2x-1,2y-1) + I^{L-1}(2x+1,2y+1) + I^{L-1}(2x-1,2y+1) + I^{L-1}(2x+1,2y+1)]
\end{aligned}
\tag{1.41}
$$

In a typical image it make no sense to go above level 4. For example if original image is 640x480 then at level 1, 2, 3 image sizes are 320x240, 160x120 and 80x60. In the implementation, we used 3 levels.

The overall pyramidal tracking algorithm proceeds as follows (Bouguet): First, the displacement (optical flow) is calculated on the deepest pyramid level, say L_m. Then the result of computation is propagated to level $L_m - 1$ in a form of an initial guess for the displacement. This guess is then refined at this level. The result is then propagated to next upper level, say $L_m - 2$ and so on until the original image level 0 is reached. In this level should displacement be extended to sub pixel level. Thus this tracker is both accurate and robust.

REFERENCES:

Allen B., Bishop G., Welch G.: Tracking: Beyond 15c Minutes of Thought, Course Notes 11, Siggraph 2001

Atkinson K. (ed): Close Range Photogrametry and Machine Vision, Whittles Publishing 2001, Scotland

Azarbayejani A., Pentland P.: Recursive Estimation of Motion, Structure and Focal Length, IEEE Transaction on Pattern Analysis and Machine Intelligence, Vol 17. No 6, June 1995

Bouguet J.Y. : Pyramid Implementation of the Lucas Kanade Feature Tracker Description of the Algorithm.

Boggart R. 1991: View Correlation, Graphics Gems II, Academic Press 1991

Crimnisi A., Reid I., Zisserman: Single View Metrology, Department of Engineering Science, University of Oxford

Doyle, A. 1999: Virtual Sets: Designing for Broadcast, millimeter 1999, *http://millimeter.com*

Deverney F., Faugeras O.: Automatic calibration and removal of distortion from scenes of structured environments, Inria Sophia Antinopolis

Foroosh H.: Visual Simulation CAP 6938, University of Central Florida

Kansy, Klaus, Thomas Berlage, Günther Schmitgen, Peter Wißkirchen (1995): Real-time integration of synthetic computer graphics into live video scenes. Proc. Interface of Real and Virtual Worlds (Montpellier, France, 26-30 June 1995)

Orad: The reality of the virtual set: Conception, Misconceptions and new Perceptions, Orad Hi-Tec Systems, in Broadcast Papers, *http://www.broadcastpapers.com*

Quan, L. 1996: Self-calibration of an Affine Camera from Multiple Views, International Journal of Computer Vision 1996, Kluwer Academic Publisher

Ribo, M. 2001: State of the Art Report on Optical Tracking

Romano, R. 2002: Projective Minimal Analysis of Camera Geometry, PhD Thesis, MIT Computer Science

Shi J., Tomasi C.: Good Feature to Track, IEEE Conference on Computer Vision and Pattern Recognition 1994

Simon, G.; Breger, M-O.: Registration with a Moving Zoom Lens Camera for Augmented Reality Applications Loria-Inria

Simon, G.; Fizzgibbon, A.; Zisserman A.: Markerless Tracking using Planar Structures in the Scene, Robotics Research Group, Department of Engineering Science

Soatto S., Perona P. 1994: Recursive Estimation of Camera Motion from Uncalibrated Image Sequences, CDS Technical Report CIT-CDS 94-005, California Institute of Technology, January 1994

Sturm P.1996: Self-Calibration of Moving Camera by Pre-calibration, 7 th Brittish Machine Vision Conference, Edinburg, Scottaland 1996

Tomasi C., Kanade T.: Detection and Tracking of Point Feature, Technical Report CMU-CS-91132, Carnegie Mellon University

Trogeman G., Graffmann B, Piesk J.: 3D-Tracking: Image Based Reconstruction of Camera Parameters for Mixed Reality Postproduction, Proceedings of the 98 European SMPTE Conference on Imaging Media, Cologne, pp 145 - 156, September 17-19, 1998

Urban M., Pajdla T., Hlavač V. 1998 Camera Self-Calibration from Multiple Views, Research Report No: K335-CMP-1998-169, Czech Technical University

Welch, G.; Bishop G. 2001: An Introduction to the Kalman Filter, Course Notes Siggraph 2001 (Course 8)

Welch, G. 2002: *The UNC Tracker Project 6D Pose Estimation for Humans and Devices,* http://www.cs.unc.edu/~tracker/

Wisskirchen, P., Kansy K. Schmitgen G. 1996: Integrating Graphics Into Video-Image-Based Camera Tracking and Filtering, In: Göbel M., David J., Slavik P., vanWijk J. (eds) Virtual Environments and Scientific Visualization ’96 Wien, Springer, ISBN 3-211-82886-9

Yao, A.; Calway A. 2002: Robust Estimation of 3-D Camera Motion for Uncalibrated Augmented Reality, 1st Symbosium on 3D Dta Processing, Visualization and Transmission (3DPVT), Padova Italy 2002

Zhang Z 1998: A Flexible New Techniques for Camera Calibration, Technical Report MSR-TR-98-71, Microsoft Research Corporation