Detection of Object's Distance and Orientation

Esa-Pekka Pyökkimies
Technical Research Centre of Finland, VTT Information Technology
Tekniikantie 4 B, P.O. Box 1203, FIN-02044 VTT, Finland
E-mail: esa-pekka.pyokkimies@vtt.fi

26.6.2002

Contents

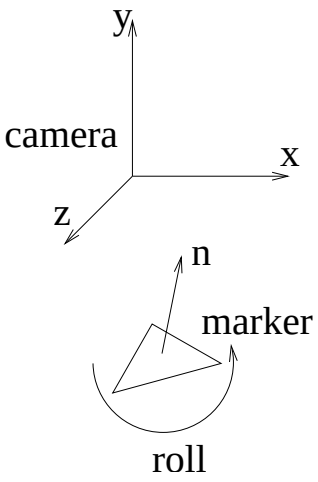


Figure 1: Wanted parameters

1 Introduction

In the pose detection problem, there is a known object, which appears in an image at an arbitrary distance and orientation from the camera. In this paper, we only use one camera. With two cameras the problem is greatly simplified [7]. The problem is calculating these six parameters (figure 1):

- Location of the object in reference to the camera. This gives three parameters, x, y and z.
- The object's normal vector $\vec{n} = (n_x, n_y, n_z)$. Normal vector needs two parameters since $n_z = \sqrt{1 - n_x^2 - n_y^2}$.
- Rotation of the object along the axis of its own normal (roll). One more parameter for a total of six.

Applications of pose detection are numerous. For example, a marker could be attached to one's forehead, allowing a camera to track the location and orientation of the head [8].

In cartography, the known object is the map, and the problem is finding the current location from landmarks on the image.

Another important class of applications are the augmented reality-type applications, where virtual objects are added to a real scene. Pose information is needed for correct registration of real and virtual objects.

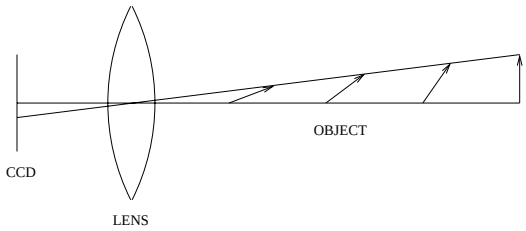


Figure 2: Simple 1D image capturing system

2 Pose Detection in One Dimension

Figure 2 shows a simple example of image capturing in one dimension. The purpose of the image is to demonstrate, that with one-dimensional marker, there are multiple solutions. Given an object of known size and ignoring possible lighting effects, the object could be at an infinite number of locations: These all project to same image on the CCD. To limit the number of possible solutions to a finite set, we must use a two-dimensional object (at least three points).

3 Pose Detection in Two Dimensions

In the following, lower-case roman characters are points in reference to camera. Upper-case roman characters are points in pixel-coordinates (upper-case X and Y are observables).

Possible two dimensional markers are e.g. square and equilateral triangle. We will soon find out that three detected points and their relations from a known object are enough to solve the pose detection problem. Arbitrary triangles are not possible, since it is impossible to tell which edge is which from the captured image. All four points are needed in square case; first, opposing corners are found and then a triangle with known edge lengths $(l, l\sqrt{2}, l)$ is deduced. Figure 3 shows two different types of markers and how they might appear in the CCD.

In figure 3, we know that

$$\begin{cases} |(x_1, y_1, z_1) - (x_2, y_2, z_2)| = l_1 \\ |(x_2, y_2, z_2) - (x_3, y_3, z_3)| = l_2 \\ |(x_3, y_3, z_3) - (x_1, y_1, z_1)| = l_3 \end{cases},$$

where $l_1 = l_2 = l_3 = l$ for equilateral triangle and $l_1 = l_3 = l, l_2 = l\sqrt{2}$ for square marker.

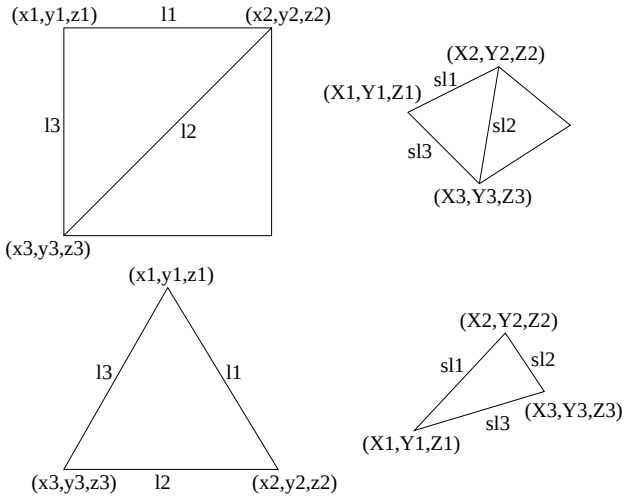


Figure 3: Equilateral triangle and square markers

The three detected points, (X_1, Y_1), (X_2, Y_2), and (X_3, Y_3) originated from directions $\vec{v}_1 = (X_1, Y_1, f)$, $\vec{v}_2 = (X_2, Y_2, f)$, and $\vec{v}_3 = (X_3, Y_3, f)$, where f is the focal length.

Now it is possible to formulate the problem:

$$\left\{ \begin{array}{rcl} |\alpha\vec{v}_1 - \beta\vec{v}_2| & = & l_1 \\ |\beta\vec{v}_2 - \gamma\vec{v}_3| & = & l_2 \\ |\gamma\vec{v}_3 - \alpha\vec{v}_1| & = & l_3 \end{array} \right. . \tag{1}$$

This system of equations has three unknowns, α, β, and γ, and three linearly independent equations, so it should have exactly one solution. Simplifying this we get

$$\left\{ \begin{array}{rcl} (\alpha\vec{v}_1 - \beta\vec{v}_2)^2 & = & l_1^2 \\ (\beta\vec{v}_2 - \gamma\vec{v}_3)^2 & = & l_2^2 \\ (\gamma\vec{v}_3 - \alpha\vec{v}_1)^2 & = & l_3^2 \end{array} \right.$$

$$\left\{ \begin{array}{rcl} \alpha^2\vec{v}_1 \cdot \vec{v}_1 - 2\alpha\beta\vec{v}_1 \cdot \vec{v}_2 + \beta^2\vec{v}_2 \cdot v_2 & = & l_1^2 \\ \beta^2\vec{v}_2 \cdot \vec{v}_2 - 2\beta\gamma\vec{v}_2 \cdot \vec{v}_3 + \gamma^2\vec{v}_3 \cdot \vec{v}_3 & = & l_2^2 \\ \gamma^2\vec{v}_3 \cdot \vec{v}_3 - 2\gamma\alpha\vec{v}_3 \cdot \vec{v}_1 + \alpha^2\vec{v}_1 \cdot \vec{v}_1 & = & l_3^2 \end{array} \right. .$$

Substituting

$$\begin{cases} \vec{v}_1 \cdot \vec{v}_1 = A \\ \vec{v}_2 \cdot \vec{v}_2 = B \\ \vec{v}_3 \cdot \vec{v}_3 = C \\ \vec{v}_1 \cdot \vec{v}_2 = D \\ \vec{v}_2 \cdot \vec{v}_3 = E \\ \vec{v}_3 \cdot \vec{v}_1 = F \end{cases}$$

we arrive to the canonical format:

$$\begin{cases} \alpha^2 A - 2\alpha\beta D + \beta^2 B = l_1^2 \\ \beta^2 B - 2\beta\gamma E + \gamma^2 C = l_2^2 \\ \gamma^2 C - 2\gamma\alpha F + \alpha^2 A = l_3^2 \end{cases} .$$

These three quadratic equations in three unknowns can be compined to yield one equation with one unknown of the fourth degree. Although fourth degree equation is solvable, the derivation is too complex, and the method of approximation was used to simplify this equation further.

4 Weak Perspective

Weak perspective transformation consists of scaling followed by orthogonal projection. Other name for weak perspective is the scaled orthographic transformation. This method takes the perspective between the camera and object in consideration with the scaling term, but perspective effects inside the object are lost. This is a close approximation to perspective transformation, if the object size is much smaller than the distance from the camera.

The weak perspective formula can be seen as an approximation of the two dimensional pose detection.

We first show that equation (1) is equivalent to equation (2):

$$\begin{cases} |(x_1, y_1, z_1) - (x_2, y_2, z_2)| = l_1 \\ |(x_2, y_2, z_2) - (x_3, y_3, z_3)| = l_2 \\ |(x_3, y_3, z_3) - (x_1, y_1, z_1)| = l_3 \end{cases} . \tag{2}$$

Here x, y, and z are the vertices' coordinates in reference to the camera, measured in arbitrary units (cm, mm), and l_1, l_2, and l_3 are the lengths of the edges in the same units.

Equation (2) has nine unknowns, but luckily there is simple relation between pixel coordinates and camera coordinates. We transform real-world coordinates into pixel coordinates with $X_i = f\frac{x_i}{z_i}$, and $Y_i = f\frac{y_i}{z_i}$. Solving for x_i and y_i and substituting we get

$$\begin{cases} \left|\left(\frac{z_1X_1}{f}, \frac{z_1Y_1}{f}, z_1\right) - \left(\frac{z_2X_2}{f}, \frac{z_2Y_2}{f}, z_2\right)\right| = l_1 \\ \left|\left(\frac{z_2X_2}{f}, \frac{z_2Y_2}{f}, z_2\right) - \left(\frac{z_3X_3}{f}, \frac{z_3Y_3}{f}, z_3\right)\right| = l_2 \\ \left|\left(\frac{z_3X_3}{f}, \frac{z_3Y_3}{f}, z_3\right) - \left(\frac{z_1X_1}{f}, \frac{z_1Y_1}{f}, z_1\right)\right| = l_3 \end{cases}.$$

Now we can take $\frac{z_i}{f}$ as common factor from all and get

$$\begin{cases} \left|\frac{z_1}{f}(X_1, Y_1, f) - \frac{z_2}{f}(X_2, Y_2, f)\right| = l_1 \\ \left|\frac{z_2}{f}(X_2, Y_2, f) - \frac{z_3}{f}(X_3, Y_3, f)\right| = l_2 \\ \left|\frac{z_3}{f}(X_3, Y_3, f) - \frac{z_1}{f}(X_1, Y_1, f)\right| = l_3 \end{cases}.$$

Now, introducing $\alpha = \frac{z_1}{f}$, $\beta = \frac{z_2}{f}$, and $\gamma = \frac{z_3}{f}$, we get exactly equation (1). α, β, and γ were treated as arbitrary variables in equation (1). Through this derivation we gained some insight to the nature of these variables. This derivation was also necessary because we can now approximate the weak perspective formula from equation (2) rather than equation (1).

We start with equation (2) and rewrite it with the equations $X_i = f\frac{x_i}{z_i}$, $Y_i = f\frac{y_i}{z_i}$, and $Z_i = f\frac{z_i}{z_i}$:

$$\begin{cases} \left|\frac{z_1}{f}(X_1, Y_1, Z_1) - \frac{z_2}{f}(X_2, Y_2, Z_2)\right| = l_1 \\ \left|\frac{z_2}{f}(X_2, Y_2, Z_2) - \frac{z_3}{f}(X_3, Y_3, Z_3)\right| = l_2 \\ \left|\frac{z_3}{f}(X_3, Y_3, Z_3) - \frac{z_1}{f}(X_1, Y_1, Z_1)\right| = l_3 \end{cases}.$$

Next comes the clever approximation step. We assume that the object is far away from the camera, that is, z_1, z_2, and z_3 are large. Further, that object dimensions, that is, $|z_1 - z_2|$, $|z_2 - z_3|$, and $|z_3 - z_1|$, are small compared to z_1, z_2, and z_3. Thus we can assume that $z_1 \approx z_2 \approx z_3 = z$. This doesn't get rid of depth information altogether, since we still have Z_1, Z_2, and Z_3, but it compresses α, β, and γ under one multiplier. Substituting $s = \frac{f}{z}$ we get

$$\begin{cases} |(X_1, Y_1, Z_1) - (X_2, Y_2, Z_2)| = sl_1 \\ |(X_2, Y_2, Z_2) - (X_3, Y_3, Z_3)| = sl_2 \\ |(X_3, Y_3, Z_3) - (X_1, Y_1, Z_1)| = sl_3 \end{cases}. \tag{3}$$

Equation (3) is the *weak perspective formula,* and an equation which we can now solve for Z_i and s. s contains large scale depth information, and Z_1, Z_2, and Z_3 local depth information. Although Z_i were equal to f before, we now allow them to be take arbitrary values so we don't lose depth information.

4.1 Solving the weak perspective equation

Since equations (3) have four unknowns (Z_1, Z_2, Z_3, s), and only three equations, we set $Z_1 = 0$. We lose absolute distance from the object, but get Z_2 and Z_3 relative to Z_1, which is enough to get pose information:

$$\begin{cases} |(X_1, Y_1, 0) - (X_2, Y_2, Z_2)| = sl_1 \\ |(X_2, Y_2, Z_2) - (X_3, Y_3, Z_3)| = sl_2 \\ |(X_3, Y_3, Z_3) - (X_1, Y_1, 0)| = sl_3 \end{cases} .$$

Raising both sides to second power we get

$$\begin{cases} (X_1 - X_2)^2 + (Y_1 - Y_2)^2 + Z_2^2 = s^2 l_1^2 \\ (X_2 - X_3)^2 + (Y_2 - Y_3)^2 + (Z_2 - Z_3)^2 = s^2 l_2^2 \\ (X_3 - X_1)^2 + (Y_3 - Y_1)^2 + Z_3^2 = s^2 l_3^2 \end{cases} .$$

Substituting

$$\begin{cases} (X_1 - X_2)^2 + (Y_1 - Y_2)^2 = d_1^2 \\ (X_2 - X_3)^2 + (Y_2 - Y_3)^2 = d_2^2 \\ (X_3 - X_1)^2 + (Y_3 - Y_1)^2 = d_3^2 \end{cases} ,$$

we can simplify to get

$$\begin{cases} d_1^2 + Z_2^2 = s^2 l_1^2 \\ d_2^2 + (Z_2 - Z_3)^2 = s^2 l_2^2 \\ d_3^2 + Z_3^2 = s^2 l_3^2 \end{cases} .$$

Expanding and moving all Zs to left hand side we get

$$\begin{cases} Z_2^2 = (sl_1)^2 - d_1^2 \\ Z_2^2 - 2Z_2 Z_3 + Z_3^2 = (sl_2)^2 - d_2^2 \\ Z_3^2 = (sl_3)^2 - d_3^2 \end{cases} . \tag{4}$$

Multiplying first and third equation with -1 and adding to the second equation yields

$$-2Z_2 Z_3 = -(sl_1)^2 + (sl_2)^2 - (sl_3)^2 + d_1^2 - d_2^2 + d_3^2 = s^2(-l_1^2 + l_2^2 - l_3^2) + d_1^2 - d_2^2 + d_3^2. \tag{5}$$

Squaring equation (5) again and using Z_2 and Z_3 from equations (4), we get

$$4\left[(sl_1)^2 - d_1^2\right]\left[(sl_3)^2 - d_3^2\right] = \left[s^2(-l_1^2 + l_2^2 - l_3^2) + d_1^2 - d_2^2 + d_3^2\right]^2$$

$$4l_1^2l_3^2s^4 - 4l_1^2d_3^2s^2 - 4l_3^2d_1^2s^2 + 4d_1^2d_3^2 = s^4(-l_1^2 + l_2^2 - l_3^2)^2 + 2s^2(-l_1^2 + l_2^2 - l_3^2)(d_1^2 - d_2^2 + d_3^2) + (d_1^2 - d_2^2 + d_3^2)^2,$$

which is a fourth degree equation in one unknown, with third and first degree terms missing, that is, bi-quadratic or quartic equation. Collecting all similars together we get

$$s^4\left[(-l_1^2 + l_2^2 - l_3^2)^2 - 4l_1^2l_3^2\right] + s^2\left[2(-l_1^2 + l_2^2 - l_3^2)(d_1^2 - d_2^2 + d_3^2) + 4l_1^2d_3^2 + 4l_3^2d_1^2\right] + \left[(d_1^2 - d_2^2 + d_3^2)^2 - 4d_1^2d_3^2\right] = 0,$$

which can be written as

$$as^4 + bs^2 + c = 0,$$

with

$$\begin{cases} a = (-l_1^2 + l_2^2 - l_3^2)^2 - 4l_1^2l_3^2 \\ b = 2(-l_1^2 + l_2^2 - l_3^2)(d_1^2 - d_2^2 + d_3^2) + 4l_1^2d_3^2 + 4l_3^2d_1^2 \\ c = (d_1^2 - d_2^2 + d_3^2)^2 - 4d_1^2d_3^2 \end{cases}.$$

This of course has the solutions

$$s^2 = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

$$s = \pm\sqrt{\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}}.$$

Since we raised equations two times to the second power during the derivation, only one of these four solutions is a solution to the original equation. From equations (3) we see that if l_1, l_2, and l_3 are positive, as they are, then s must also be positive. Therefore we can replace the first $\pm$ with a $+$. Since we don't want imaginary solutions, we also pick from the remaining two solutions the one that is real. From experimentation we know that the correct solution always is

$$s = \sqrt{\frac{-b - \sqrt{b^2 - 4ac}}{2a}},$$

but at this point we don't have a proof for this.

Now that s has been solved, Z_2 and Z_3 can be solved from equations (4):

$$\begin{cases} Z_2 = \pm\sqrt{(sl_1)^2 - d_1^2} \\ Z_3 = \pm\sqrt{(sl_3)^2 - d_3^2} \end{cases}.$$

From equation (5) we see that Z_2 and Z_3 hold a certain relation. That is, if $s^2(-l_1^2+l_2^2-l_3^2)+d_1^2-d_2^2+d_3^2$ is positive, then Z_2 and Z_3 must be oppositely signed, and if negative, Z_2 and Z_3 must be both positive or both negative. By introducing helper variable σ :

$$\sigma = \begin{cases} -1 & \text{, if } s^2(-l_1^2+l_2^2-l_3^2)+d_1^2-d_2^2+d_3^2 > 0 \\ 1 & \text{, otherwise} \end{cases},$$

we can write the Z_i as

$$\begin{cases} Z_2 = \sigma\sqrt{(sl_1)^2-d_1^2} \\ Z_3 = \sqrt{(sl_3)^2-d_3^2} \end{cases} \bigvee \begin{cases} Z_2 = -\sigma\sqrt{(sl_1)^2-d_1^2} \\ Z_3 = -\sqrt{(sl_3)^2-d_3^2} \end{cases}.$$

These two solutions are the final Z_i. Selecting the correct one is not possible using only weak-perspective information, but we do not have a proof for this (it just makes sense, if you flip a rectangular marker around the x-axis, it will look the same, ignoring lighting and perspective effects). This duality can be removed in practical applications using perspective information, e.g. checking which edge of a rectangular marker is shorter. Another possibility is to always choose the other one and require the application to be symmetrical in respect to these two solutions.

Now we have enough information to solve two degrees of freedom: the direction of the normal vector of the marker. First we have the two base vectors:

$$\vec{b_1} = \frac{(X_2-X_1, Y_2-Y_1, Z_2)}{\sqrt{(X_2-X_1)^2+(Y_2-Y_1)^2+Z_2^2}} = (b_{11}, b_{21}, b_{31}),$$

and

$$\vec{b_2} = \frac{(X_3-X_1, Y_3-Y_1, Z_3)}{\sqrt{(X_3-X_1)^2+(Y_3-Y_1)^2+Z_3^2}} = (b_{12}, b_{22}, b_{32}).$$

These vectors should be at 90 degree angle to each other, in the weak perspective model. However, since some perspective is always present, the angle might not be exactly 90 degrees and could add some error to calculations.

From the two base vectors we can now calculate via the cross product the third base:

$$\vec{b_3} = \vec{b_1} \times \vec{b_2} = (b_{13}, b_{23}, b_{33}).$$

With a complete base,

$$B = \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix},$$

it is now possible to travel in the marker coordinate system with linear combinations, and do transformations between image and marker coordinate systems.

Algorithm 1 shows reference matlab implementation of the pose detection algorithm.

4.2 Calculating depth information using weak perspective

Although sometimes it is not required to know the distance of the marker from the camera, it can be useful. Now that we know Z_i, i.e. the marker depth in image coordinates, we can solve the approximate distance to the camera in real coordinates quite simply. Of course the distance is an approximation of an approximation, so it is not necessarily very accurate. The only extra information we need is the focal length of the camera. The distance from the camera can be solved from the following system of equations:

$$\begin{cases} |t(X_2, Y_2, f) - q(X_3, Y_3, f)| = L_2 \\ t \cdot f - q \cdot f = Z_2 - Z_3 \end{cases} . \tag{6}$$

Here $L_2 = s \cdot l_2$, since we decide to solve the problem in image space first. Remember that multiplying by s is approximately same as multiplying by $\frac{f}{z}$, that is, it transforms real world coordinates to image coordinates. The first equation requires that the distance between two principal rays is the same as the corresponding edge length, and the second requires that the Z difference is the previously calculated Z difference. Solving $q = t - \frac{Z_2 - Z_3}{f} = t - k$, with $k = \frac{Z_2 - Z_3}{f}$ from the second equation and substituting to the first we get

$$|t(X_2, Y_2, f) - (t - k)(X_3, Y_3, f)| = L_2$$

$$|t(X_2 - X_3) + k \cdot X_3, t(Y_2 - Y_3) + k \cdot Y_3, k \cdot f| = L_2$$

$$|ta + b, tc + d, e| = L_2,$$

with

$$\begin{cases} a = X_1 - X_2 \\ b = k \cdot X_2 \\ c = Y_1 - Y_2 \\ d = k \cdot Y_2 \\ e = k \cdot f \end{cases} .$$

Algorithm 1 Pose detection reference implementation

```
tol=1e-3;

% l1 is distance between vertex (X1,Y1) and (X2,Y2) in arbitraty units (cm,mm). l2 is distance between (X2,Y2) and (X3,Y3), and l3 between (X3,Y3) and (X1,Y1)
l1=1;
l2=sqrt(2);
l3=1;

% Detected marker vertices from the image are X1,Y1,X2,Y2,X3,Y3. These demonstrate a marker facing the camera far away at 640x480 resolution
X1=320;Y1=240;
X2=321;Y2=240;
X3=320;Y3=241;

% calculate pose detection
d1=sqrt((X2-X1)^2+(Y2-Y1)^2);
d2=sqrt((X3-X2)^2+(Y3-Y2)^2);
d3=sqrt((X1-X3)^2+(Y1-Y3)^2);

a=(-l1^2+l2^2-l3^2)^2-4*l1^2*l3^2;
b=2*(-l1^2+l2^2-l3^2)*(d1^2-d2^2+d3^2)+4*l1^2*d3^2+4*l3^2*d1^2;
c=(d1^2-d2^2+d3^2)^2-4*d1^2*d3^2;

s=sqrt((-b-sqrt(b^2-4*a*c))/2/a);

if abs(imag(s))>tol,error('no solution'),end

if s^2*(-l1^2+l2^2-l3^2)+d1^2-d2^2+d3^2>0,si=-1;else,si=1;end

Z2=-si*sqrt((s*l1)^2-d1^2);
Z3=-sqrt((s*l3)^2-d3^2);

if imag(Z2)>tol | imag(Z3)>tol,error('no solution'),end

B1=[X2-X1,Y2-Y1,Z2]/sqrt((X2-X1)^2+(Y2-Y1)^2+Z2^2);
B2=[X3-X1,Y3-Y1,Z3]/sqrt((X3-X1)^2+(Y3-Y1)^2+Z3^2);
B3=cross(B1,B2)

% B3 is the normal vector of the marker in camera coordinate system
```

Solving this we get a quadratic equation in one variable:

$$(ta + b)^2 + (tc + d)^2 + e^2 = L_2^2$$

$$t^2a^2 + 2tab + b^2 + t^2c^2 + 2tcd + d^2 + e^2 = L_2^2$$

$$t^2(a^2 + c^2) + 2t(ab + cd) + (b^2 + d^2 + e^2 - L_2^2) = 0.$$

t will once again have two solutions, but only the other is correct. The correct one can be selected by substituting them to the original equation (6). After correct t is found, the z depth for the second vertex can be found with $z_2 = \frac{t \cdot f}{s}$. Dividing with s transforms image coordinates (approximately) to world coordinates. z_3 can be found with $z_3 = \frac{(t-k) \cdot f}{s}$. z_1 can be found by repeating this procedure e.g. with vertices one and two. The global x and y coordinates can also be calculated in this manner, i.e. $x_2 = \frac{t \cdot X_2}{s}$, and $y_2 = \frac{t \cdot Y_2}{s}$. Now we have solved another three degrees of freedom: global x, y, and z coordinates.

If only average distance to the marker is needed, we can use the relation $s = \frac{f}{z}$ and solve for $z = \frac{f}{s}$, but this is not distance to any particular vertex. Instead it is some sort of average.

4.3 Other pose calculation methods

There exists a variety of different methods for pose calculation. Some are analytic, others iterative. Exact perspective or different simplifications of perspective can be used. The number of points can be varied. Some algorithms require camera calibration, others don't. The algorithm presented above is an analytic Perspective-3-Point (P3P) method with weak perspective simplication, and doesn't require camera calibration. An analytic solution for exact P4P method is presented in [4]. Different P3P, P4P, and PnP algorithms are derived in [10]. Iterative method using Newton's method to iterate correct solution is described in [5]. Another iterative method solving linear system of equations is shown in [9]. A geometrical proof for uniqueness of pose under weak perspective up to a reflection is shown in [6]. Article [11] discusses that the distance-based PnP-problem (finding a distance to each object), is different than the transformation- based PnP-problem (finding transformation from object to camera coordinates).

5 Accuracy

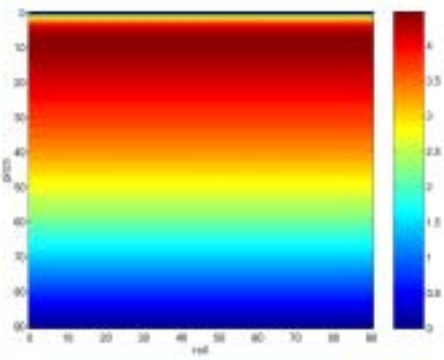
5.1 Pose accuracy

Figure 4 shows results obtained from synthetic accuracy analysis. A camera system was simulated in matlab to first obtain the projected points on the simulated CCD. After this, algorithms discussed in previous sections were used to obtain the direction of the normal vector. After this the angle between the calculated and real normal vector was calculated in degrees, and plotted as a function of pitch and roll. Pitch goes from 0 to 90 degrees and Roll from 0 to 360 degrees, but the quadrants are symmetric (flipping of the image) and therefore roll is plotted from 0 to 90 degrees.

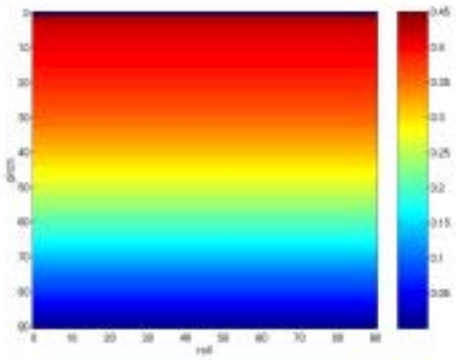
With Pitch=0 and Roll=0, the marker is facing the camera. Then the marker is rotated Pitch degrees around the x-axis (of the marker), and then Roll degrees around the normal vector of the marker.

Figures 4a and 4b show that accuracy is only a function of pitch, assuming infinite resolution. Infinite resolution means that after the virtual marker was projected on the virtual CCD, no rounding to nearest integer was done. The average accuracy of figure 4b is about 10 times as much as that of figure 4a. This comes from the fact that in figure 4b the marker is 10 times as far as in figure 4a. As was discussed before, the algorithm assumes that the marker is sufficiently far from the camera so that there is no perspective inside the object. Thus with increased distance we gain more accuracy, if resolution is not a problem. Same effect can be gained by reducing the size of the marker to 1 cm. We can also see that when Pitch=0, we always get correct results. Then, as the pitch is increased, there is a sharp increase in error. The error maxes around 10 degrees, and then starts to gradually decrease as the marker is rotated away from the camera. Error is zero again when the marker is reduced to a line. The figures 4a and 4b show the minimum possible error. With real world cameras the error must always be larger than this.

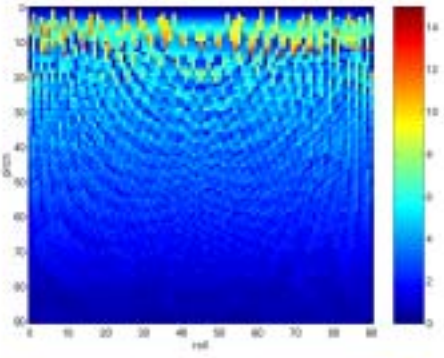
Figure 4c shows how the CCD resolution affects detection error. Now the marker's projections on the virtual CCD were rounded to nearest integer, simulating a real-world camera. The maximum error jumped from figure 4a's 4.5 degrees to 11.1 degrees, but the average error only showed slight increase from 2.7 to 3.0. Again we see that error is largest, when the marker is facing or nearly facing the camera. As the marker is rotated away from the camera, the error drops fast. The first 20 degrees are the most difficult to detect correctly, after that it is easier. What we also see in figure 4c is that



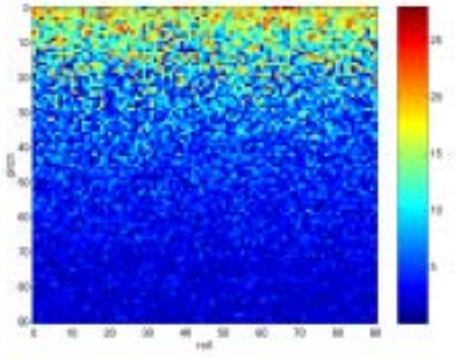
a) 10 cm marker at 100 cm distance. 90 degree field of view. Infinite resolution. Max: 4.5 deg Avg: 2.7 deg



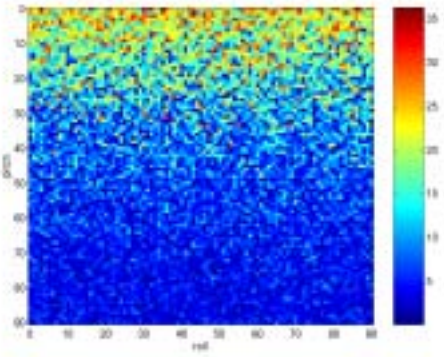
b) 10 cm marker at 1000 cm distance. 90 degree field of view. Infinite resolution. Max: 0.45 deg Avg: 0.26 deg



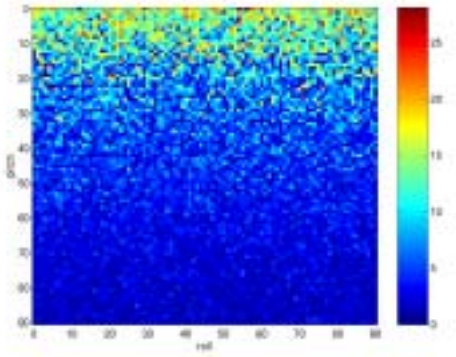
c) 10 cm marker at 100 cm distance. 90 degree field of view. 640x480 resolution. Max: 15.0 deg Avg: 3.0 deg



d) 10 cm marker at 100 cm distance. 90 degree field of view. 640x480 resolution. ±2 pixel error. Max: 28.0 deg Avg: 6.1 deg



e) 10 cm marker at 100 cm distance. 90 degree field of view. 640x480 resolution. ±4 pixel error. Max: 36.3 deg Avg: 9.6 deg



f) 10 cm marker at 100 cm distance. 90 degree field of view. 640x480 resolution. Normally distributed error with variance one. Max: 28.1 deg Avg: 5.5 deg

Figure 4: Pose Accuracy

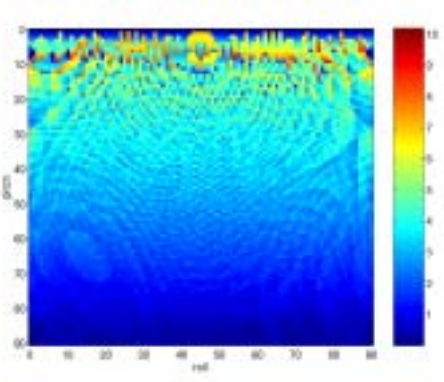
there are spots of zero or near-zero error between the areas of large error. This can show as irritating 'jitter' in the detection. I.e. the marker is only moved a little, but the difference in the pose detection can be much larger.

Figures 4d&e show how uniformly distributed random numbers in the detection affect the calculation. Figure 4d has ± 2 pixel error and figure 4e ± 4 pixel error. The maximum and average errors go up accordingly. The error is still highest when the marker is facing the camera. We conjecture that this comes from the fact that if the marker is pitched a little when it is facing the camera, there will be very little change in the observables, i.e. the vertex positions. On the other hand when the marker is at over 45 degree angle to the camera, pitching the marker yields much larger changes in the vertex positions. Figure 4f has normally distributed error.

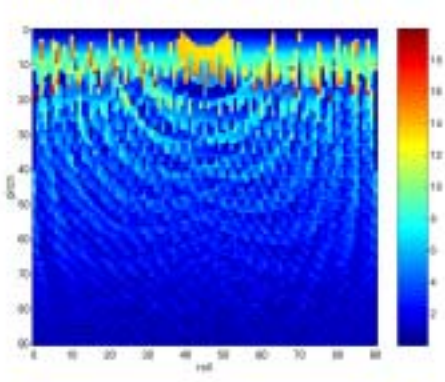
Figure 5 shows how CCD resolution affects pose detection. Generally error goes up as resolution goes down. Figure 5d is exactly the same as 5c. This demonstrates that accuracy can be preserved with lower resolutions, as long as field of view is decreased accordinly. Figure 5d has same focal length as 5c but lower resolution, which means narrower field of view.

Accuracy can always be increased by using more resolution on the CCD, or using better algorithms to detect the vertices from the CCD. If resolution is not a bottleneck, then accuracy can be increased by using smaller marker, larger field of view, or taking the marker further from the camera. On the other hand if resolution is the bottleneck, then the opposite is true. On the algorithm level small is good, but if we can't get reasonable detection on the CCD the algorithm level accuracy is wasted. Increasing/decreasing marker size has both good and bad effects and those must be weighted when using these algorithms in a specific situation. There is probably even an optimum distance from the camera where detection is most accurate.

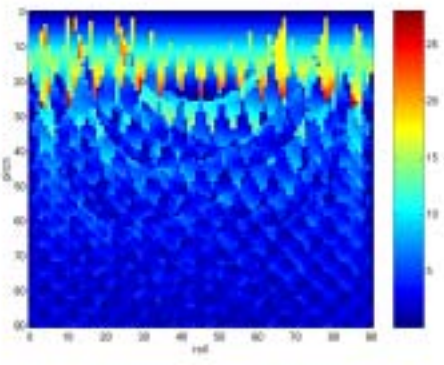
The pose detection is the same regardless of the marker's position on the CCD, since all calculations are done using relative distances. But, if a marker that is at a zero degree angle in front of the camera, i.e. it appears at a line, is moved to edges of the image, it is no longer a line. A marker that is not at the center will thus give more erroneous normal vectors than a marker at the center. However, for image augmenting purposes(section 7.4), this is the desired behavior. If a marker is at zero degree angle in absolute coordinates, but does not appear as a line in the image, then we should augment based on how the marker looks in the image, and not based on absolute coordinates. Figure 6a shows how the normal vector directions become much more erraneous compared to 4a. But, as said before, it doesn't matter for image augmenting purposes. Figure 6b shows that compared to



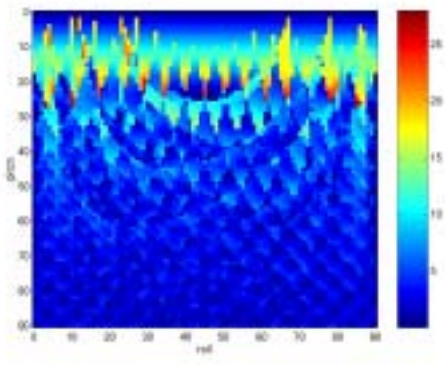
a) 10 cm marker at 100 cm distance. 90 degree field of view. 1280x960 resolution. Max: 10.2 deg Avg: 2.8 deg



b) 10 cm marker at 100 cm distance. 90 degree field of view. 320x240 resolution. Max: 20.0 deg Avg: 3.7 deg

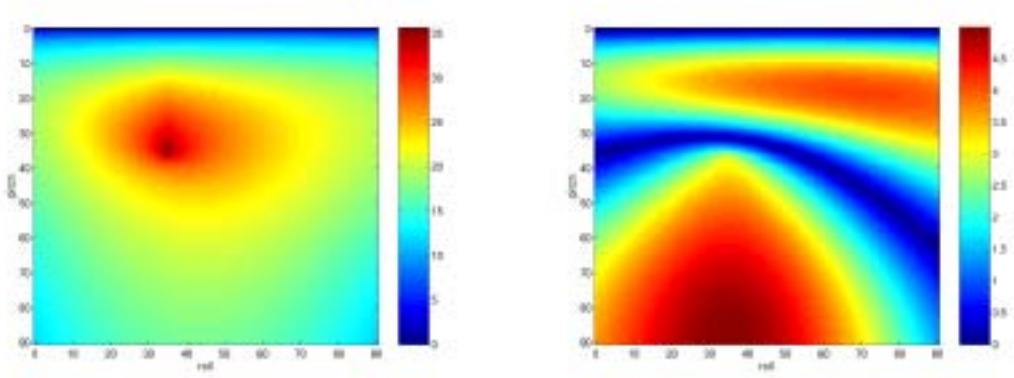


c) 10 cm marker at 100 cm distance. 90 degree field of view. 160x120 resolution. Max: 28.0 deg Avg: 5.3 deg



d) 10 cm marker at 100 cm distance. 53 degree field of view. 80x60 resolution. Max: 28.0 deg Avg: 5.3 deg

Figure 5: Effect of resolution on accuracy



a) Angle accuracy of off-center marker. 10 cm marker at 100 cm distance and 20 cm+20 cm off center. 90 degree field of view. Infinite resolution. Max: 35.7 deg Avg: 19.1 deg.

b) Depth accuracy of off-center marker. 10 cm marker at 100 cm distance and 20 cm+20 cm off center. 90 degree field of view. Infinite resolution. Max: 5.0 cm Avg: 2.7 cm

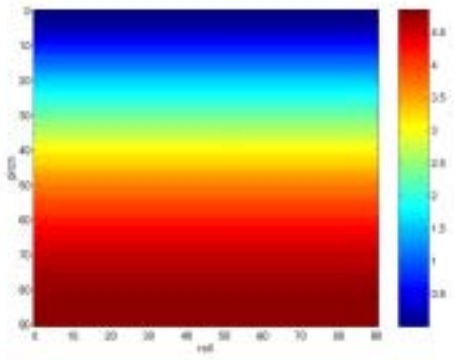
Figure 6: Off center marker accuracy

7a, depth accuracy is about the same, but error distribution is different.

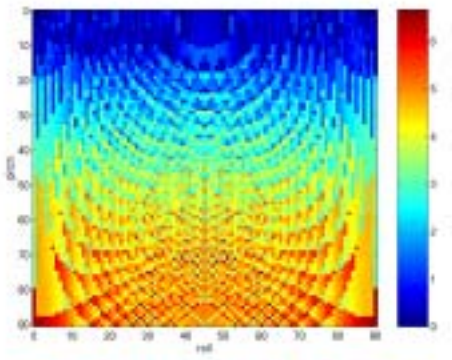
The camera system can have different optical properties on different regions of the CCD, such as quadratic distortions. Distortions should or should not be removed before the pose is calculated, depending on the situation. For image augmenting purposes, the distortions probably don't need to be removed, but if absolute values are needed, then distortions should be corrected before the pose is calculated.

5.2 Depth accuracy

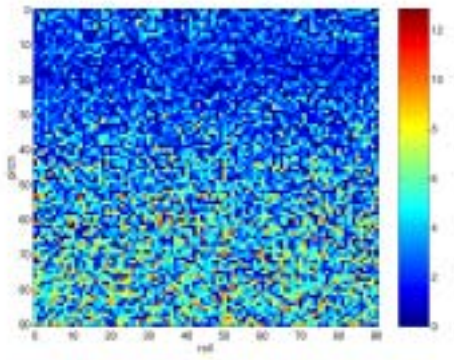
Figure 7 shows depth calculation using the relation $z = \frac{f}{s}$. Error is the distance between calculated and actual depth. Depth accuracy follows an exact opposite trend than pose accuracy. When the marker is facing the camera, we get error-free depth information, and as the marker is turned, the error increases. Biggest error comes when the marker is reduced to a line, which is easy to understand, since the line's width depends on the rotation as well. Figure 7d shows this particularly well. The depth is correct at 0 and 90 degrees, and the error varies in between. The average error is quite constant over various resolutions (3.0 cm at infinite and 3.3 cm 160x120), but the maximum error increases a lot (4.9 cm at infinite and 12.8 cm at 160x120).



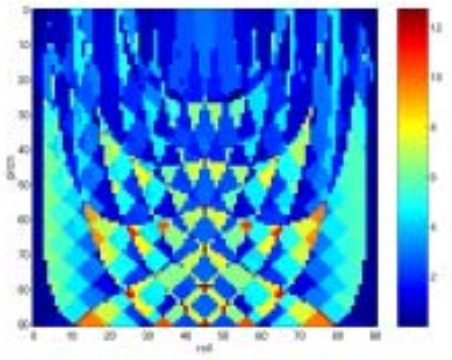
a) 10 cm marker at 100 cm distance. 90 degree field of view. Infinite resolution. Max: 4.9 cm Avg: 3.0 cm



b) 10 cm marker at 100 cm distance. 90 degree field of view. 640x480 resolution. Max: 6.7 cm Avg: 3.0 cm

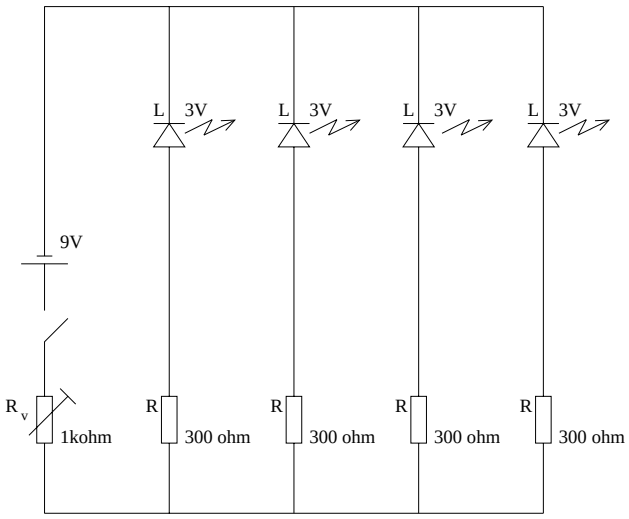


c) 10 cm marker at 100 cm distance. 90 degree field of view. 640x480 resolution. ±2 pixel error. Max: 12.9 cm Avg: 3.4 cm



d) 10 cm marker at 100 cm distance. 90 degree field of view. 160x120 resolution. Max: 12.8 cm Avg: 3.3 cm

Figure 7: Depth accuracy



Part list

R	300 ohm $\frac{1}{4}$ W resistor
Rv	1000 ohm trimmer
L	High intensity 3V white LED

Figure 8: Circuit diagram for LED marker

6 Designing the Marker

We decided to make a square marker, with four bright white LEDs as vertices. This has the advantage of having high signal-to-noise ratio, if CCD's shutter is set almost close. The effect is that only the LEDs are seen in the raw image, even without any processing.

Circuit diagram (figure 8) shows the design that was used. With 3 volt high efficiency white leds, current can be adjusted with a trimmer from a maximum of 20 mA to about 1 mA $\left(I_{led} = \frac{1.5V}{R_v + 75\Omega}\right)$. If lower voltage leds are used, bigger resistors should be used or the leds might burn out. Leds were placed parallel instead of series, because 9 V battery can't power four 3 V leds. Series placement is otherwise advantageous since it saves power and led intensities will be more stable, although we haven't noticed any fluctuations in intensities so far.

The components were soldered into an IC breadboard with copper connects. Figure 9 shows our realization of the circuit. The second picture shows an

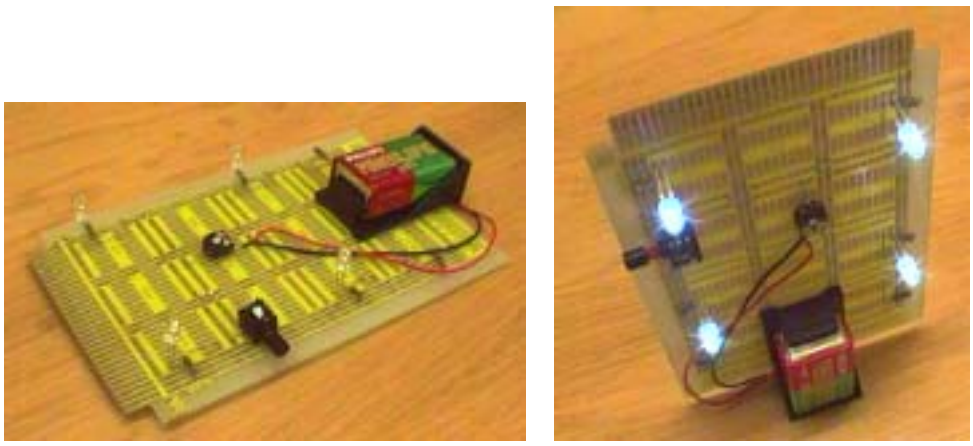


Figure 9: LED marker realization

a) Shutter set to auto b) Shutter manually set to almost closed

Figure 10: LED marker detection

active circuit. Notice how the battery compartment doubles as a stand. The LED separation is 10 cm.

7 Results

Figure 10 shows the effect shutter has on the image. The surroundings vanish completely leaving only the LEDs visible. This means that no processing needs to be to done to extract the LEDs from the image.

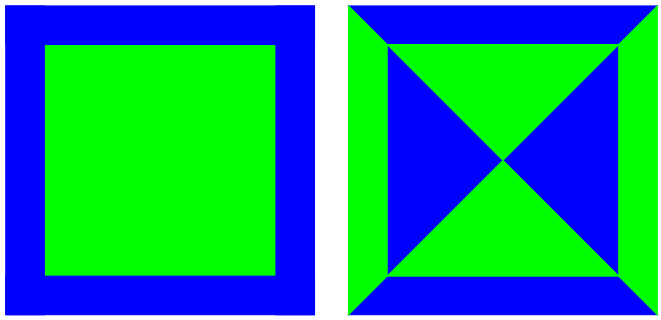


Figure 11: Two color markers

7.1 Other markers

The marker can also be passive – colorful piece of paper for example – but this has several disadvantages. First, e.g. green piece of paper can be detected to be anything from yellow to cyan by the CCD. Second, the scene can easily have similar green in other regions, requiring heuristic to eliminate false positives. We tried with green and blue papers (red is almost impossible because humans are red, and further because we had red floors...), and had mixing success. Sometimes paper works just fine, but sunlight etc. can suddenly make the detection difficult.

More advanced method is to use two-color marker (figure 11). Now the criterion of possible marker edge is adjacent two colors, green and blue. This eliminates almost all false positives. There are still some, but usually only just few pixels, which can be eliminated with continuity checks.

With two-color marker, the RGB thresholding range can be very wide compared to single color detection. Figure 12 shows the RGB values which were used to detect green and blue. As can be seen from the picture, all colors between yellow and cyan are classified as green, and all colors between cyan and magenta are classified as blue. More precisely, green is defined as all hue values from 60 to 180, and blue is defined as all hue values from 181 to 299. Some minimum saturation is also required, which is why in figure 12 the volumes don't reach the grey diagonal. If similarly wide colour ranges were used in single-color markers, the amount of false positives would be extremely high.

Even with very wide colour ranges, the detection is quite robust. Figure 13 demonstrates the phases of marker detection.

We found that the two-color marker gives reasonably robust detection over a wide variety of lighting conditions, however the LED marker is still superior

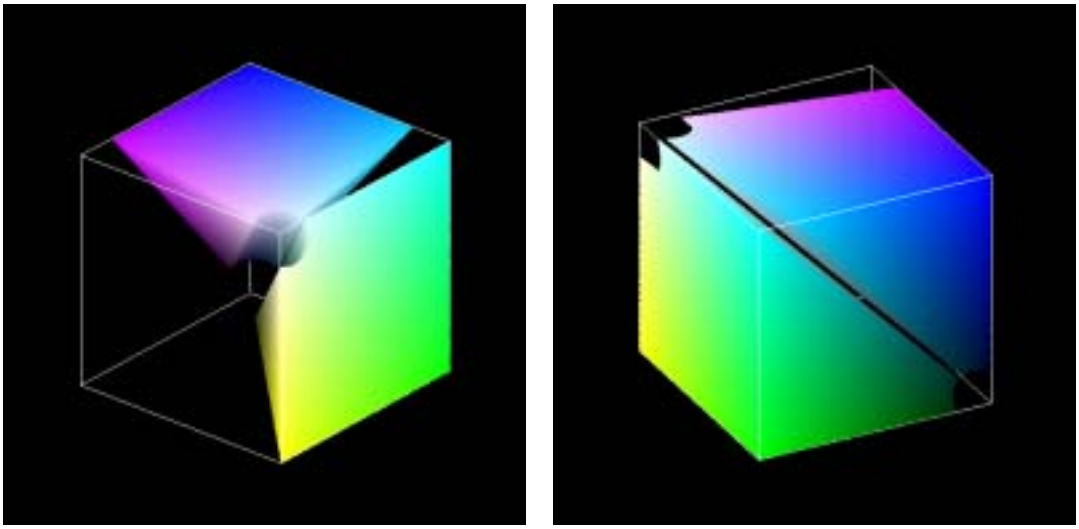
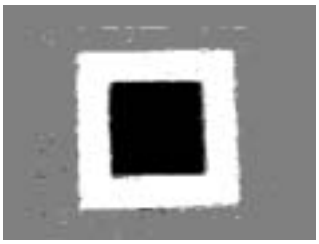
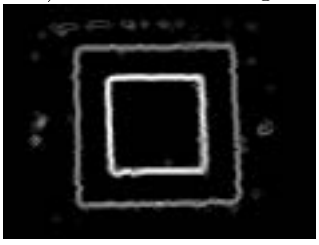


Figure 12: RGB-cube used in two color marker detection

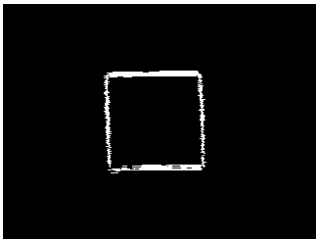
a) Captured image



b) Green=white. Blue=black. Rest gray.

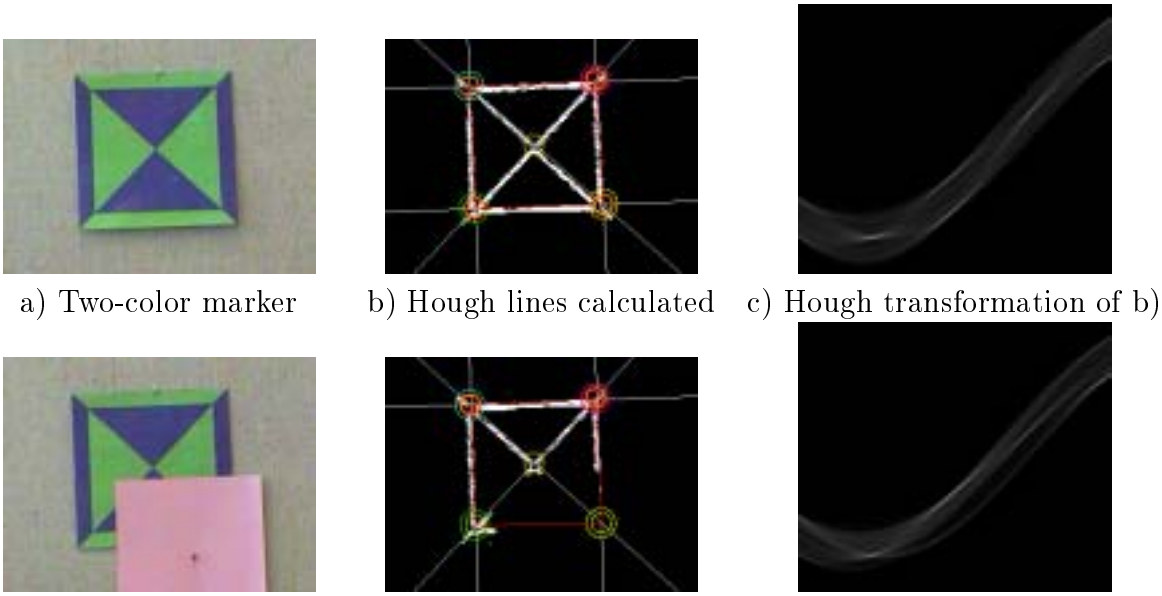


c) Gradient edge detection



d) Thresholding of edges

Figure 13: Processing of two-color marker



a) Two-color marker b) Hough lines calculated c) Hough transformation of b)

d) Part of marker blocked e) Hough lines calculated f) Hough transformation of e)

Figure 14: Partial markers

and requires much less processing. With LEDs, there is an additional possibility of modulation, which would give even more robustness, but we have yet to try that out.

7.2 Partial markers

It is possible to make use of even partially blocked markers using an image processing algorithm called the Hough transformation. Hough transformation is cabable of finding non-continuous lines. This is advantageous if there is a lot of noise in the image and lines are often broken. Figure 14 demonstrates a marker specifically designed to take advantage of the Hough transformation. Even with a very large blockade, there are enough lines to calculate the location of the obstructed corner. The yellow circles are calculated intersections, and double circles are heuristically selected corner locations. The idea is that the location of corners can be encoded in all parts of the image, so that a small portion of the image can be used to find the corners. The marker in figure 14 has 2 "backup" edges in addition to the four normal edges, but there could be more. This way the robustness of two-color markers can be raised even higher.

7.3 Matrix markers

With symmetric markers, the "roll" parameter can't be recovered completely. There will allways be four possible orientations which are possible. By adding a matrix inside the marker, we can label one of the corners and deduce the roll parameter also. Figure 15 shows the matrix marker we used. First, the outer black-white border is used to find the edges, and after corners have been calculated, the matrix inside is checked to see which edge is which. This allows image augmenting, where the virtual object can be rotated 360 degrees, instead of just 90 degrees.

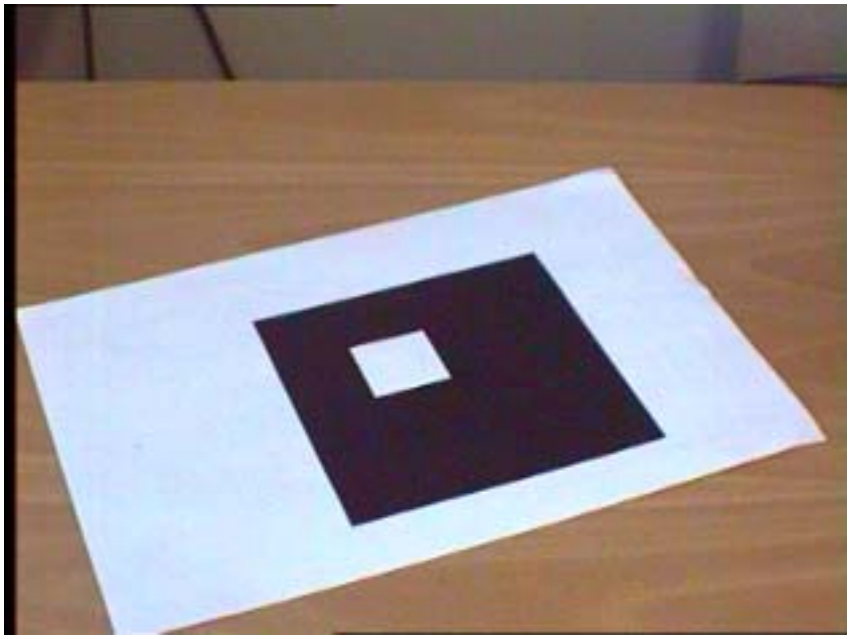
Matrix markers can also act as a checksum to make the detection process more robust. After corners have been found, the matrix can be checked. If the matrix inside is not correct, the corners are probably false. Even 2x2 (white,black,black,black) matrix eliminates most false detections, although not all.

Third use for matrix marker is the possibility to have multiple markers on scene, again with checking of the matrix we can determine which marker is which.

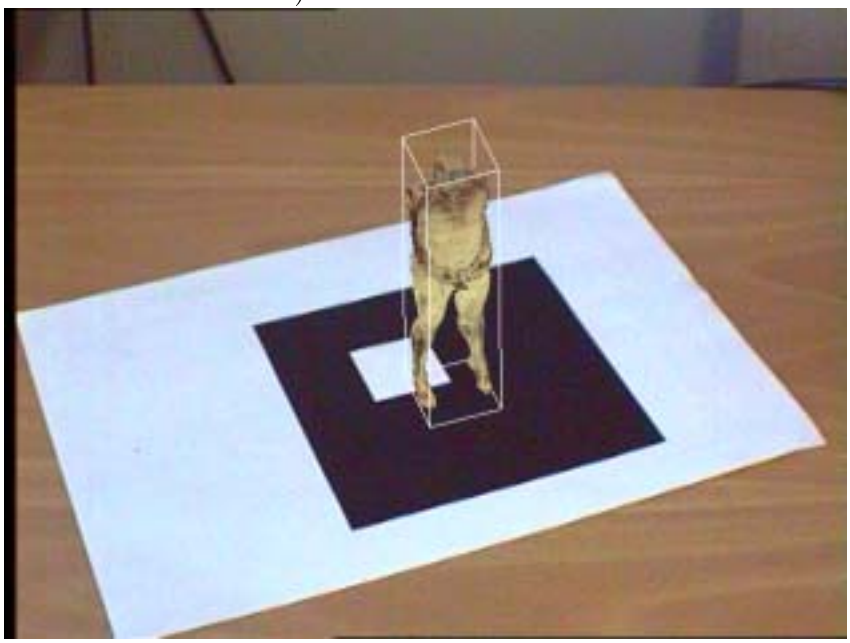
7.4 Image augmenting

Image augmenting means adding virtual objects to a real scene. A survey of different augmentation methods and applications is presented in [1]. Markers utilizing a matrix code printed inside can be used to have multiple markers on scene [2]. With our method of augmenting, no information about camera focal length or other camera calibration parameters are needed. Different method of calibration-free image augmenting is presented in [3].

Using pose information, i.e. the marker-image transformation matrix and scale factor as calculated before, the image can be augmented in reference to the marker. Figure 16 shows the marker captured in different angles, and augmented with a cube. As can be seen, the cube is realistically in correct orientation in respect to the marker. Image augmenting only requires the image normal direction and scale factor, i.e. solving the weak perspective equation. Depth information about the invidual vertices are not necessary. Figure 17 has the visible human data augmented on top of the marker. The data was renderered in correct orientation and size using the normal vector and scale parameter, and then a single translation was done to superimpose the body in the right place in the image. Figure 18 shows augmenting of a MRI head.



a) 2x2 matrix marker



b) Image augmenting using matrix marker

Figure 15: Matrix marker

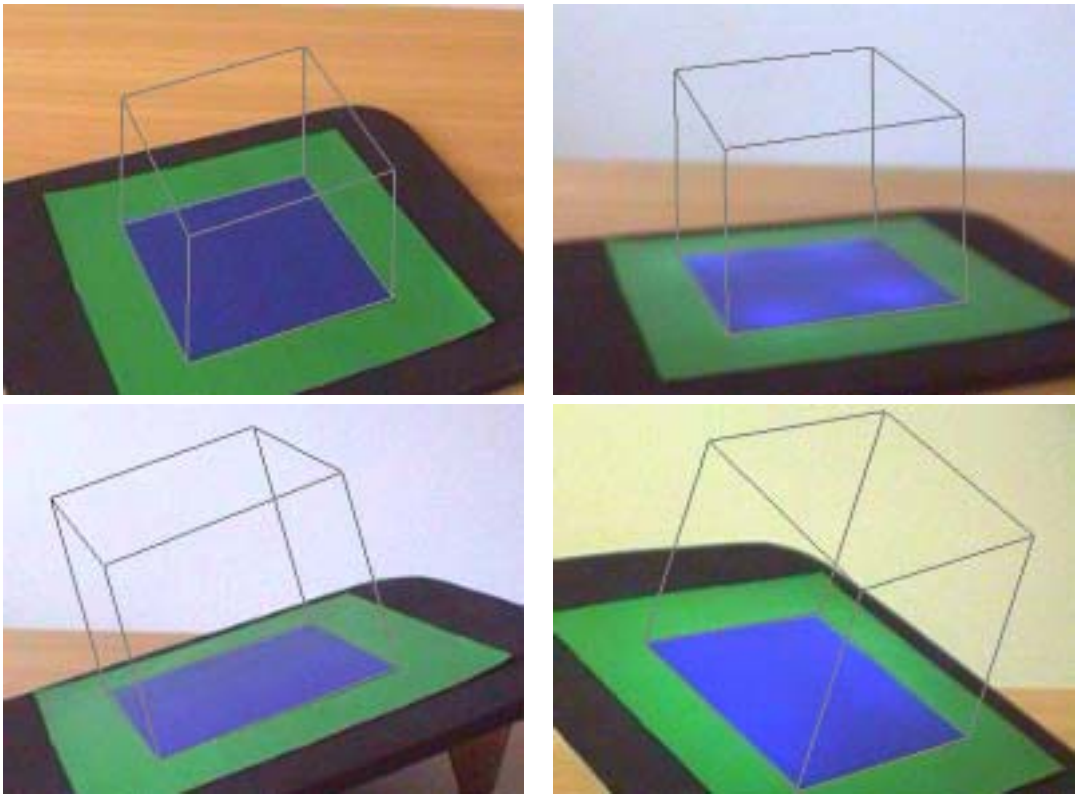


Figure 16: Image augmenting with cubes

a) Visible human held in hand

b) Coke bottle as size reference

Figure 17: Visible human augmentation

Figure 18: MRI data augmenting

8 Conclusions

We developed a complete pose calculation system, with different markers and marker segmentation algorithms, the P3P pose calculation method, and several augmentation methods with wireframe and volumetric data. Future directions for development are better marker detection and detection of multiple markers. Currently marker detection works best with relatively simple scenes with mainly the marker showing. If there are a lot of clutter on the scene, the detection algorithms can be fooled. After marker corners have been correcly detected, the remaining calculations are foolproof. We are confident that the marker detection can be made quite robust with matrix markers, or modulated infrared LEDs.

References

[1] Azuma, R.T.,"A Survey of Agumented Reality", *Presence: Teleoperators and Virtual Environments*, August 1997, pages 355-385.

[2] Rekimoto, J., "Matrix: A Realtime Object dentification and Registration Method for Augmented Reality", *Proc. of Asia Pacific Computer Human Interaction (APCHI '98)*, 1998.

[3] Kutulakos, K.N; Vallino, J.R., "Calibration-Free Augmented Reality", *IEEE Transactions on Visualization and Computer Graphics*, volume 4, number 1, 1998, pages 1-20.

[4] Horaud, R.; Conio, B.; Leboulleux, O., "An Analytic Solution for the Perspective 4-Point Problem", *Computer Vision, Graphics, and Image Processing*, Volume 47, 1989, pages 33-44.

[5] Lowe, D.G., "Three-Dimensional Object Recognition from Single Two-Dimensional Images", *Artificial Intelligence*, Volume 31, 1987, pages 355-395

[6] Huang, T.S.; Bruckstein, A.M.; Holt, R.J.; Netravali, A.N., "Uniqueness of 3D Pose Under Weak Perspective: A Geometrical Proof", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Volume 17, 1995, pages 1220-1221

[7] Pyökkimies, E.-P., "3D camera", Technical Document, Helsinki university of Technology, 2001.

[8] Heuring, J.J.; Murray, D.W., "Visual Head Tracking and Slaving for Visual Telepresence", *Proc. of IEEE International Conference on Robotics and Automation*, Minneapolis, Minnesota, April 1996.

[9] Horaud, R.; Christy, S.; Dornaika, F.; Lamiroy, B., "Object Pose: Links between Paraperspective and Perspective", *Proc. of 5th International Conference on Computer Vision*, Cambridge, MA, USA, June 1995, pages 426-433.

[10] Quan, L.; Lan Z., "Linear N-point camera pose determination", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Volume 21, Issue 8, August 1999, pages 774-780

[11] Hu, Z.Y.; Wu, F.C., "A note on the number of Solutions of the Non-coplanar P4P problem", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Volume 24, Number 4, April 2002, pages 550-555.