

Projection, stereo and panoramic images

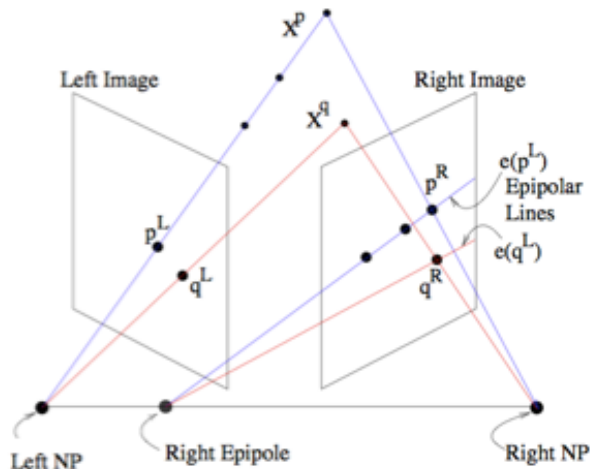
X^p and X^q are the physical location of objects.

NP are the nadir points of the cameras.

The line connecting the left and right NP is the baseline.

The projection of the left nadir, NP is seen as a the left epipole w.r.t. right image. The epipoles may or may not be within the border of the images.

The projection of X^p to left nadir, NP is seen as an epipole line in the right image. If more than one epipole line is present in the right image, they converge at the right epipole (which might not be within the boundaries of the image).



<http://www.cs.toronto.edu/~jepson/csc420/notes/epiPolarGeom.pdf>

(note, the image is posted on an educational site and copied here without following up on permissions. Any further use of the image should follow up on the origins and permissions.)

Once the points in the left image, X_L are matched with points in the right image, X_R , if there are at least 7 points, one can determine the “bifocal tensor”, a.k.a. “fundamental matrix”, relating the points in the 2 images using a 3x3 matrix of rank 2.

$(X_L)^T * F * X_R = 0$ for any pair of points in the images.

Note: the “Essential matrix” is a matrix used if the camera details are known. The “bifocal tensor”, a.k.a. “fundamental matrix” does not need camera details.

2D transformations

from “Computer Vision: Algorithms and Applications” by Szeliski
(Note, please consult author and book for any use of this. it’s presented here for educational purposes only.)

2D Translation. $\tilde{x}' = \tilde{x} + t$

$$\tilde{x}' = \begin{bmatrix} I & t \end{bmatrix} \tilde{x}$$

$$\tilde{x}' = \begin{bmatrix} I & t \\ 0^T & 1 \end{bmatrix} \tilde{x}$$

where I is the (2x2) identity matrix

2D Affine: $\tilde{x}' = A\tilde{x}$

$$\tilde{x}' = \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \end{bmatrix} \tilde{x}.$$

preserves parallel property of lines.

2D Rotation + translation (a.k.a. rigid body motion, a.k.a. 2D euclidian):

$$\tilde{x}' = R\tilde{x} + t$$

$$\tilde{x}' = \begin{bmatrix} R & t \end{bmatrix} \tilde{x} \quad R = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix}$$

2D Scaled rotation (a.k.a. similarity transform): $\tilde{x}' = sR\tilde{x} + t$

$$\tilde{x}' = \begin{bmatrix} sR & t \end{bmatrix} \tilde{x} = \begin{bmatrix} a & -b & t_x \\ b & a & t_y \end{bmatrix} \tilde{x},$$

preserves angles between lines.

2D Projective (a.k.a. perspective transformation, homography): $\tilde{x}' = H\tilde{x}$

Note that H is homogeneous, i.e., it is only defined up to a scale, Two H matrices that differ only by scale are equivalent.

The resulting homogeneous coordinate \tilde{x}' must be normalized in order to obtain an inhomogeneous result:

$$x' = \frac{h_{00}x + h_{01}y + h_{02}}{h_{20}x + h_{21}y + h_{22}} \text{ and } y' = \frac{h_{10}x + h_{11}y + h_{12}}{h_{20}x + h_{21}y + h_{22}}.$$

preserves straightness property of lines.






Transformation	Matrix	# DoF	Preserves	Icon
translation	$\begin{bmatrix} I & t \end{bmatrix}_{2 \times 3}$	2	orientation	
rigid (Euclidean)	$\begin{bmatrix} R & t \end{bmatrix}_{2 \times 3}$	3	lengths	
similarity	$\begin{bmatrix} sR & t \end{bmatrix}_{2 \times 3}$	4	angles	
affine	$\begin{bmatrix} A \end{bmatrix}_{2 \times 3}$	6	parallelism	
projective	$\begin{bmatrix} H \end{bmatrix}_{3 \times 3}$	8	straight lines	

Table 2.1 Hierarchy of 2D coordinate transformations. Each transformation also preserves the properties listed in the rows below it, i.e., similarity preserves not only angles but also parallelism and straight lines. The 2×3 matrices are extended with a third $[0^T \ 1]$ row to form a full 3×3 matrix for homogeneous coordinate transformations.

Planar surface flow: Bilinear interpolant:

$$x' = a_0 + a_1x + a_2y + a_6x^2 + a_7xy$$

$$y' = a_3 + a_4x + a_5y + a_7x^2 + a_6xy$$

planar surface with a small 3D motion

Bilinear interpolant:

$$x' = a_0 + a_1x + a_2y + a_6xy$$

$$y' = a_3 + a_4x + a_5y + a_7xy$$

useful for interpolation of sparse grids using splines, but does not preserve straight lines

3D transformations

from “Computer Vision: Algorithms and Applications” by Szeliski
(Note, please consult author and book for any use of this. it’s presented here for educational purposes only.)

3D Translation. $\mathbf{x}' = \mathbf{x} + \mathbf{t}$

$$\mathbf{x}' = \begin{bmatrix} \mathbf{I} & \mathbf{t} \end{bmatrix} \bar{\mathbf{x}}$$

where \mathbf{I} is the (3×3) identity matrix

3D Affine: $\mathbf{x}' = \mathbf{A}\mathbf{x}$

$$\mathbf{x}' = \begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \end{bmatrix} \bar{\mathbf{x}}$$

preserves parallel property of lines and planes.

3D Rotation + translation (a.k.a. rigid body motion, a.k.a. 3D euclidian): $\mathbf{x}' = \mathbf{R}\mathbf{x} + \mathbf{t}$

$$\mathbf{x}' = \begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix} \bar{\mathbf{x}}$$

where \mathbf{R} is a 3×3 orthonormal rotation matrix with $\mathbf{R}\mathbf{R}^T = \mathbf{I}$ and $|\mathbf{R}| = 1$.

can describe a rigid motion using

$$\mathbf{x}' = \mathbf{R}(\mathbf{x} - \mathbf{c}) = \mathbf{R}\mathbf{x} - \mathbf{R}\mathbf{c}$$

where \mathbf{c} is the center of rotation (often the camera center).

3D Scaled rotation (a.k.a. similarity transform): $\mathbf{x}' = \mathbf{sR}\mathbf{x} + \mathbf{t}$

$$\mathbf{x}' = \begin{bmatrix} \mathbf{sR} & \mathbf{t} \end{bmatrix} \bar{\mathbf{x}} = \begin{bmatrix} a & -b & t_x \\ b & a & t_y \end{bmatrix} \bar{\mathbf{x}},$$

preserves angles between lines and planes.

3D Projective (a.k.a. perspective transformation, homography): $\tilde{\mathbf{x}}' = \tilde{\mathbf{H}}\tilde{\mathbf{x}}$

Note that $\tilde{\mathbf{H}}$ is homogeneous, i.e., it is only defined up to a scale, Two \mathbf{H} matrices that differ only by scale are equivalent.

The resulting homogeneous coordinate $\tilde{\mathbf{x}}'$ must be normalized in order to obtain an inhomogeneous result:

preserves straightness property of lines.






Transformation	Matrix	# DoF	Preserves	Icon
translation	$\begin{bmatrix} \mathbf{I} & \mathbf{t} \end{bmatrix}_{3 \times 4}$	3	orientation	
rigid (Euclidean)	$\begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix}_{3 \times 4}$	6	lengths	
similarity	$\begin{bmatrix} \mathbf{sR} & \mathbf{t} \end{bmatrix}_{3 \times 4}$	7	angles	
affine	$\begin{bmatrix} \mathbf{A} \end{bmatrix}_{3 \times 4}$	12	parallelism	
projective	$\begin{bmatrix} \tilde{\mathbf{H}} \end{bmatrix}_{4 \times 4}$	15	straight lines	

Table 2.2 Hierarchy of 3D coordinate transformations. Each transformation also preserves the properties listed in the rows below it, i.e., similarity preserves not only angles but also parallelism and straight lines. The 3×4 matrices are extended with a fourth $[0^T \ 1]$ row to form a full 4×4 matrix for homogeneous coordinate transformations. The mnemonic icons are drawn in 2D but are meant to suggest transformations occurring in a full 3D cube.

3D transformations

from “Computer Vision: Algorithms and Applications” by Szeliski

(Note, please consult author and book for any use of this. it's presented here for educational purposes only.)

3D Rotation represented by a rotation axis and angle (a.k.a. Rodriguez's formula).

rotation axis and angle as a vector $\omega = \theta \hat{n}$

let $[\hat{n}]_{\times}$ be the matrix form of the cross product operator with the vector $\hat{n} = (\hat{n}_x, \hat{n}_y, \hat{n}_z)$,

$$[\hat{n}]_{\times} = \begin{bmatrix} 0 & -\hat{n}_z & \hat{n}_y \\ \hat{n}_z & 0 & -\hat{n}_x \\ -\hat{n}_y & \hat{n}_x & 0 \end{bmatrix} \text{ and}$$

$$\mathbf{R}(\hat{n}, \theta) = \mathbf{I} + \sin \theta [\hat{n}]_{\times} + (1 - \cos \theta) [\hat{n}]_{\times}^2$$

good for small rotations

For infinitesimal or instantaneous) rotations and θ expressed in radians, Rodriguez's formula simplifies to:

$$\mathbf{R}(\omega) \approx \mathbf{I} + \sin \theta [\hat{n}]_{\times} \approx \mathbf{I} + [\theta \hat{n}]_{\times} = \begin{bmatrix} 1 & -\omega_z & \omega_y \\ \omega_z & 1 & -\omega_x \\ -\omega_y & \omega_x & 1 \end{bmatrix}$$

3D transformations

from “Computer Vision: Algorithms and Applications” by Szeliski
(Note, please consult author and book for any use of this. it’s presented here for educational purposes only.)

3D Rotation represented Unit quaternions:

(Shoemake 1985)

a unit length 4-vector whose components can be written as

$q = (q_x, q_y, q_z, q_w)$ or $q = (x, y, z, w)$

Unit quaternions live on the unit sphere $\|q\| = 1$ and *antipodal* (opposite sign) quaternions, q and $-q$, represent the same rotation

^ “origin” $q_0 = (0, 0, 0, 1)$.

$$q = (v, w) = \left(\sin \frac{\theta}{2} \hat{n}, \cos \frac{\theta}{2} \right), \quad \begin{aligned} R(\hat{n}, \theta) &= I + \sin \theta [\hat{n}]_{\times} + (1 - \cos \theta) [\hat{n}]_{\times}^2 \\ &= I + 2w[v]_{\times} + 2[v]_{\times}^2. \end{aligned}$$

to express Rodriguez’s formula:

$$R(q) = \begin{bmatrix} 1 - 2(y^2 + z^2) & 2(xy - zw) & 2(xz + yw) \\ 2(xy + zw) & 1 - 2(x^2 + z^2) & 2(yz - xw) \\ 2(xz - yw) & 2(yz + xw) & 1 - 2(x^2 + y^2) \end{bmatrix}. \quad (2.41)$$

The diagonal terms can be made more symmetrical by replacing $1 - 2(y^2 + z^2)$ with $(x^2 + w^2 - y^2 - z^2)$, etc.

Given two quaternions $q_0 = (v_0, w_0)$ and $q_1 = (v_1, w_1)$,

the *quaternion multiply* operator is defined as:

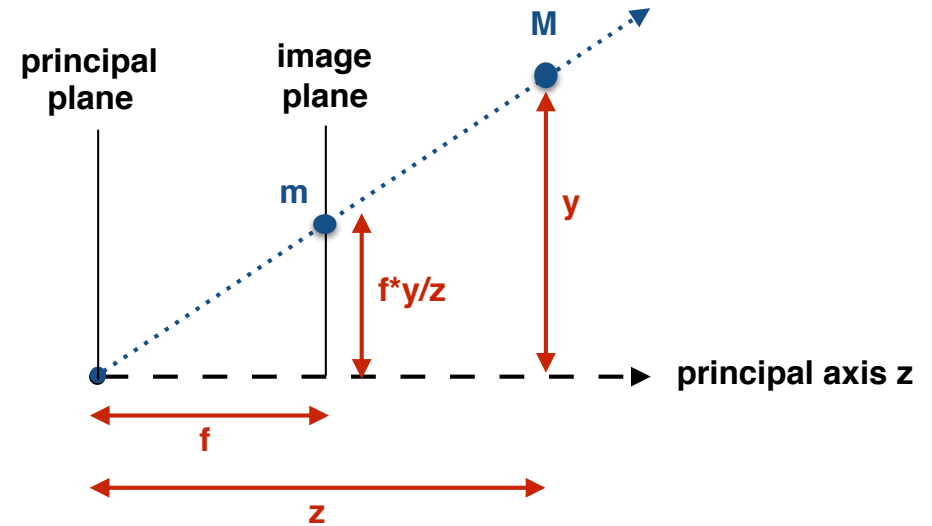
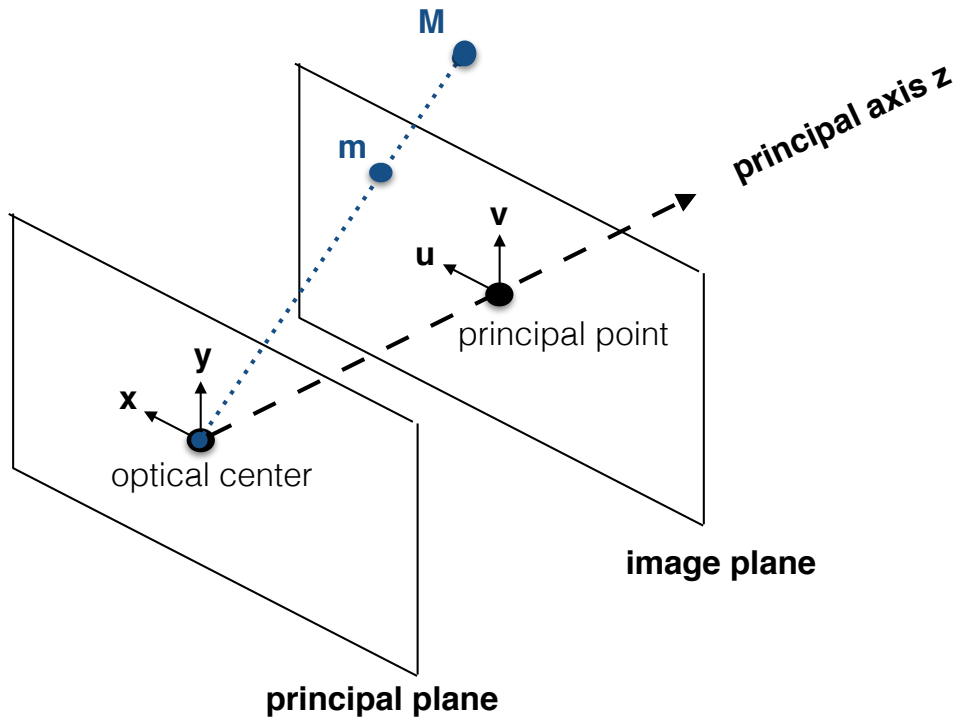
$$q_2 = q_0 q_1 = (v_0 \times v_1 + w_0 v_1 + w_1 v_0, w_0 w_1 - v_0 \cdot v_1),$$

quaternion division:

$$q_2 = q_0 / q_1 = q_0 q_1^{-1} = (v_0 \times v_1 + w_0 v_1 - w_1 v_0, -w_0 w_1 - v_0 \cdot v_1).$$

quaternions are good to use if need to keep track of a smoothly moving camera, since there are no discontinuities in the representation. It is also easier to interpolate between rotations and to chain rigid transformations (Murray, Li, and Sastry 1994; Bregler and Malik 1998)

Pinhole camera geometry



camera projection matrix P is defined in $z^* \mathbf{m} = P^* \mathbf{M}$
 where $M = (x, y, z, 1)^T$ and $m = (f^*x/z, f^*y/z, 1)^T$

The right side of the projection eqn can be modified by rotation matrix \mathbf{R} and translation vector \mathbf{t} to put the coordinates in the (external) world coordinate system. The matrix is $\begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix}$
 The six parameters are called *external parameters*.
 rows of \mathbf{R} and the *optical center* describe the *camera reference frame* in world coordinates.

The left side of the projection eqn can be modified to change coordinates in the image frame.

$K = \begin{bmatrix} f/s_x & f/s_x \cot \theta & o_x \\ 0 & f/s_y & o_y \\ 0 & 0 & 1 \end{bmatrix}$ K is the *camera calibration matrix*,
 result is pixel coords in image plane.

Pinhole camera geometry (cont.)

intrinsic parameters:

f is the focal distance in mm

o_x, o_y is the principal point in pixel coordinates

s_x is the width of the pixel in mm

s_y is the height of the pixel in mm

ϕ is the angle between the axes, usually 90 degrees

The aspect ratio s_x/s_y is usually 1

The extrinsic parameters depend upon Rotation and Translation of the camera.

P can be rewritten as $P = \mathbf{K}^* [\mathbf{I} \mid \mathbf{0}]^* \mathbf{G} = \mathbf{K}^* [\mathbf{R} \mid \mathbf{t}]$

A scale factor λ applied is $P = \lambda^* \mathbf{K}^* [\mathbf{R} \mid \mathbf{t}]$

This is a 3x4 full rank matrix and can be factorized by QR factorization.

P can be rewritten as $P = [\mathbf{P}_{3 \times 3} \mid \mathbf{p}_4]$ where $\mathbf{P}_{3 \times 3}$ is the first 3 rows of P and \mathbf{p}_4 is the 4th column.

If $\lambda=1$, the matrix is normalized and the distance of \mathbf{M} from the focal plane of the camera is *depth*.

For the image plane at infinity, the image of points doesn't depend on camera position.

The angle between 2 rays is $\cos \theta = \mathbf{m1}^T * \omega * \mathbf{m2} / (\text{sqrt}(\mathbf{m1}^T * \omega * \mathbf{m1}) * (\text{sqrt}(\mathbf{m2}^T * \omega * \mathbf{m2}))$

extrinsic parameters are the position and orientation of the camera w.r.t. a coord frame.

(Notes on pinhole camera followed by a few notes on multi-view geometry are from http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/FUSIELLO4/tutorial.html)

projection and the principal and image planes

remembering that the center of projection is in the principal plane one can define the position of m with respect to the 0,0 coordinates in the image plane in which it is in.

u_c and v_c are the coordinates of the principal point.

m is at x,y in that image plane.

$$u = u_c + (x/\text{pixelWidth}) \quad \text{and} \quad v = v_c + (y/\text{pixelHeight})$$

Similarly, the coordinates in the world coordinate system can be transformed to the image plane coordinate system.

$$Z * u = Z * u_c + (X * f) / \text{pixelWidth}$$

$$Z * v = Z * v_c + (Y * f) / \text{pixelHeight}$$

using scaling factor:

$$\begin{bmatrix} \text{scale} * u \\ \text{scale} * v \\ \text{scale} \end{bmatrix} = \begin{bmatrix} (f/\text{pixelWidth}) & 0 & u_c & 0 \\ 0 & (f/\text{pixelHeight}) & v_c & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

camera calibration matrix:

$$C = \begin{bmatrix} r_1 * (f/\text{pixelWidth}) + u_c * r_3 & t_x * (f/\text{pixelWidth}) + u_c * t_z \\ r_2 * (f/\text{pixelHeight}) + v_c * r_3 & t_y * (f/\text{pixelHeight}) + v_c * t_z \\ r_3 & t_z \end{bmatrix}$$

multi-view geometry

With point correspondences $\mathbf{m}_i \longleftrightarrow \mathbf{M}_i$, can solve for camera projection matrix P . Using the Kroenecker delta product and vec operator, coefficients are rewritten to their linearly independent forms of a matrix of size $2n \times 12$ called the coefficient matrix A

From a set of n point correspondences, we obtain a $2n \times 12$ coefficient matrix A , where n must be > 6 .

In general A will have rank 11 (provided that the points are not all coplanar) and the solution is the *1-dimensional right null-space of A* .

The linear system of equations for inexact data is solved using least squares.

The least-squares solution for $\text{vec } P^T$ is the singular vector corresponding to the smallest *singular value* decomposition of A and the direct linear transform.

The equation is usually written in form $A * h = 0$ where h is $\text{vec } P^T$ (its the one dimensional reshaping of the homograph matrix).

multi-view geometry (cont.)

In general:

- dot product of a point and a line is zero if the point lies on the line
- if the *intrinsic parameters* are known, the relationship between corresponding points is given by the *essential matrix*: $\mathbf{m}_r^T * \mathbf{E} * \mathbf{m}_l^T$

- when no camera details are available, one can use the *fundamental matrix*.

It's a 3x3 rank 2 homogeneous matrix. It has 7 degrees of freedom.

For any point \mathbf{m}_l in the left image, the corresponding epipolar line \mathbf{l}_r in the right image can be expressed as $\mathbf{l}_r = \mathbf{F} * \mathbf{m}_l$ and vice versa $\mathbf{l}_l = \mathbf{F}^T * \mathbf{m}_r$

- the *essential and fundamental matrices* are related through $\mathbf{F} = \mathbf{K}_r^{-T} * \mathbf{E} * \mathbf{K}_l^{-T}$ where \mathbf{K}_r and \mathbf{K}_l are the left and right camera matrices

Regarding 3D reconstruction:

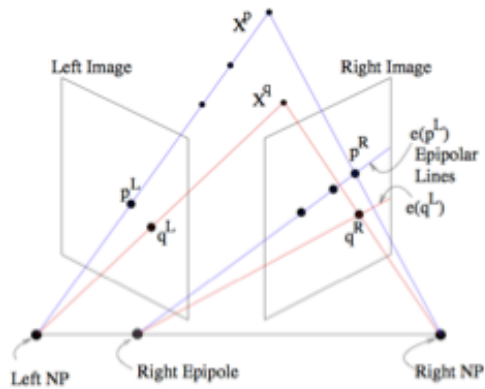
- when both *intrinsic* and *extrinsic* camera parameters are known, the reconstruction is solved unambiguously by triangulation.

- when only *intrinsic* parameters are known, extrinsic parameters can be estimated and the reconstruction can be solved up to an unknown scale factor, that is \mathbf{R} can be estimated, but \mathbf{t} can be only up to a scale factor. the epipolar geometry is the essential matrix and the solvable projection is euclidean (rigid+ uniform scale)

- when neither *intrinsic* nor *extrinsic* parameters are known, i.e., the only information available are pixel correspondences, the reconstruction can be solved up to an unknown, global projective transformation of the world.

In Trifocal geometry, a trifocal tensor is used.

Projection, stereo and panoramic images



<http://www.cs.toronto.edu/~jepson/csc420/notes/epiPolarGeom.pdf>
(note, the image is posted on an educational site and copied here without following up on permissions. Any further use of the image should follow up on the origins and permissions.)

panoramic images here are from
Brown & Lowe 2003



Projection, stereo and panoramic images

single point perspective and two and three point perspectives sometimes can have “foreshortened” object dimensions if the axes of the object are not parallel to the camera plane.

For example, when creating correspondence for the gingerbread man in the image below using partial shape matching and euclidean transformation for evaluation, half of the object is unmatched within a tolerance.

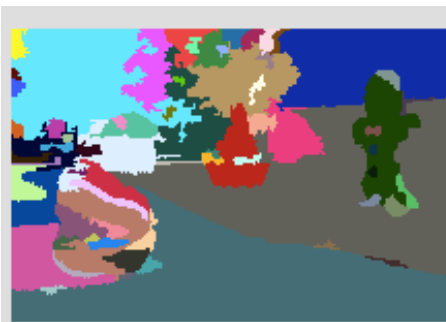
So, a rough calculation of the angle of inclination is needed to find the other matching points consistent with a single-point perspective projection.

The “foreshortening” along the x-axis is corrected using cosine of the angle, that is the true dimension is the measured divided by cosine of angle.

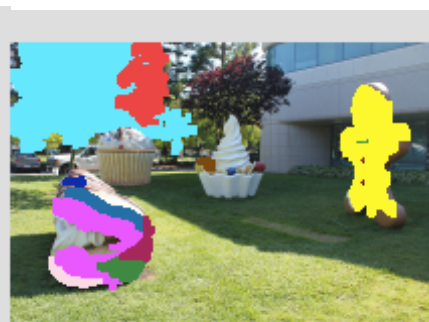
The foreshortening along the y axis can be approximated by the ratio of the far sides of two triangles, one embedded in the other.



**image for the search for
gingerbread man**

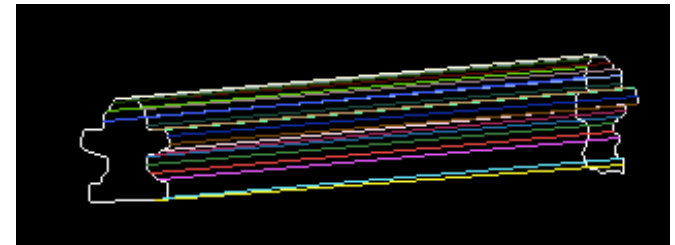


segmentation



**segmentation filtered to
colors similar to the
template object
gingerbread man (not
shown)**

**euclidean transformed matches
...need a perspective projection
search for the other half**



The remaining notes and snapshots on corners, stereo matching and matching under conditions of different lighting, pose and location are in file **doc/colorSegmentation3.pdf**

Note: tried the use of a segmentation filter based upon the atrous wavelet to limit the points to be matched. (later methods of MSER + HOGs proved more successful here).

