Review of Computer Vision

Siming He

January 26, 2023, 14:42

Contents

1 Camera Model

1.1 3D Point Projection

3D world is projected on to the CCD plane of camera, i.e. from a 3D point (X, Y, Z) to a coordinate on the projection plane $(u_{ccd}, v_{ccd}) = (X\frac{f_m}{Z}, Y\frac{f_m}{Z})$ where f_m is the facal length. Then, the projection on CCD plane is mapped to the pixel space of image, i.e. from (u_{ccd}, v_{ccd}) to $(u_{img}, v_{img}) = (u_{ccd}\frac{W_{img}}{W_{ccd}} + p_x, v_{ccd}\frac{H_{img}}{H_{ccd}} + p_y)$.

1.2 Homogeneous Coordinate

Definition 1.1 (Homogeneous Coordinate)**.** For a given point (x, y) on the Euclidean plane, for any non-zero real number λ, $(x\lambda, y\lambda, \lambda)$ is a set of homogeneous coordinates for (x, y). The set forms the ray passing through origin and (x, y).

Homogeneous representation of 3D point projection is

$$\lambda \begin{bmatrix} u_{img} \\ v_{img} \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = K \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

where K is the camera intrinsic parameter to map metric space to pixel space.

2D inverse projection can be easily defined:

$$\lambda K^{-1} \begin{bmatrix} u_{img} \\ v_{img} \\ 1 \end{bmatrix} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

1.3 Coordinate Transform

Consider coordinate transformation from point u in image I_1 to point v in image I_2.

Uniform Scaling

$$\begin{bmatrix} v_x \\ v_y \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_x \\ u_y \\ 1 \end{bmatrix} \qquad \text{with } s_x = s_y$$

Aspect Ratio Change

$$\begin{bmatrix} v_x \\ v_y \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_x \\ u_y \\ 1 \end{bmatrix} \qquad \text{with } s_x \neq s_y$$

Translation

$$\begin{bmatrix} v_x \\ v_y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_x \\ u_y \\ 1 \end{bmatrix}$$

Rotation

$$\begin{bmatrix} v_x \\ v_y \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_x \\ u_y \\ 1 \end{bmatrix}$$

Euclidean Transform SE(2) The LIE Group SE(2) is

$$SE(2) = \{A | A = \begin{bmatrix} R & r \\ 0_{1\times 2} & 1 \end{bmatrix}, R \in R^{2\times 2}, r \in R^2, R^T R = RR^T = I, |R| = 1\}$$

$$\begin{bmatrix} v_x \\ v_y \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_x \\ u_y \\ 1 \end{bmatrix} = A \begin{bmatrix} u_x \\ u_y \\ 1 \end{bmatrix} \qquad A \in SE(2)$$

Length, angle, and area are invariant under Euclidean Transform SE(2). The degree of freedom is 3 (2 translation and 1 rotation).

Similarity Transform is the combination of Euclidean Transform and Uniform Scaling. Length ratio and angle are invariant under this transformation. The degree of freedom is 4 (2 translation and 1 rotation + 1 scaling).

Affine Transform

$$\begin{bmatrix} v_x \\ v_y \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_x \\ u_y \\ 1 \end{bmatrix}$$

Parallelism, ratio of area, and ratio of length are invariant under this transformation. The degree of freedom is 6 (2 translation and 1 rotation + 1 scaling + 2 shearings).

Perspective Transform (Homography, collineation, projectivity, projective transform minus scale) is the general form of plane to plane linear mapping.

$$\lambda \begin{bmatrix} v_x \\ v_y \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix} \begin{bmatrix} u_x \\ u_y \\ 1 \end{bmatrix}$$

Cross ratio, concurrency, colinearity are invariant under this transformation. The degree of freedom is 8 (2 translation + 1 rotation + 1 scaling + 2 shearings + 2 transformations perpendicular to the plane) (9 variables - 1 scale).

1.4 Homography Computation

$$\lambda \begin{bmatrix} v_x \\ v_y \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} u_x \\ u_y \\ 1 \end{bmatrix} = H \begin{bmatrix} u_x \\ u_y \\ 1 \end{bmatrix}$$

Given v_x, v_y, u_x, u_y, we want to find H. Let $\lambda = h_{31}u_x + h_{32}u_y + h_{33}$, then we can get

$$v_x = \frac{h_{11}u_x + h_{12}u_y + h_{13}}{h_{31}u_x + h_{32}u_y + h_{33}}$$
$$v_y = \frac{h_{21}u_x + h_{22}u_y + h_{23}}{h_{31}u_x + h_{32}u_y + h_{33}}$$

i.e.

$$h_{11}u_x + h_{12}u_y + h_{13} - (h_{31}u_x + h_{32}u_y + h_{33})v_x = 0$$
$$h_{21}u_x + h_{22}u_y + h_{23} - (h_{31}u_x + h_{32}u_y + h_{33})v_y = 0$$

we get the new system of equation

$$\begin{bmatrix} u_x & u_y & 1 & 0 & 0 & 0 & -u_xv_x & -u_yv_x & -v_x \\ 0 & 0 & 0 & u_x & u_y & 1 & -u_xv_y & -u_yv_y & -v_y \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

The system of equation is under-determined and we need at least 4 points to solve the system $Ax = 0$ (4 points gives 8 constraints for the 8 degree of freedom i.e. no need to worry about h_{33}). It is a homogeneous linear least squares problem that we solve by doing SVD on $A = U\Sigma V^T$ and finding the right singular vector v_i with smallest singular value as the solution.

Proof. We want to find a non-trivial solution. Hence, we add a constraint such that $\|x\| = 1$. And we do constrained least-square minimization to find x^* that minimizes $\|Ax\| = 1$. $Av_i = U\Sigma V^T v_i = U\Sigma e_i = U\sigma_i e_i = \sigma_i u_i$ where e_i is a standard basis vactor. Since σ_i is the smallest singular value, such Ax^* is the minimized one. □

2 Feature Detection and Matching

We want features that will look different when we move the patch, i.e. corners. **Feature detection** is the process of finding image regions that have maximum variation when moved in all directions around the region. Feature description is used to describe a feature so that it can be find in other images.

2.1 Harris Corner Detection

Since we want to find image regions that have maximum variation when moved in all directions, we find the difference in intensity for a displacement aruond the given region.

Definition 2.1 (Moravec's corner detector function)**.** Define window function $w(x,y) : \mathbb{R}^2 \Rightarrow \mathbb{R}$ which can be a rectangular region A such that $w(x,y) = 1, (x,y) \in A$ and $w(x,y) = 0, (x,y) \notin A$. Or it can be a Gaussian window around the center of the window. Define intensity at (x,y) as $I(x,y)$.
Then, Moravec's corner detector function is

$$\begin{aligned}
E(u,v) &= \sum_{(x,y)} w(x,y)|I(x+u,y+v) - I(x,y)|^2 \\
&\approx \sum_{(x,y)} w(x,y)|I(x,y) + uI_x + vI_y - I(x,y)|^2 \quad \text{first order taylor series approximation} \\
&= \sum_{(x,y)} w(x,y)u^2I_x^2 + v^2I_y^2 + 2uvI_xI_y \\
&= \sum_{(x,y)} w(x,y) \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} \\
&= \sum_{(x,y)} \begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix}
\end{aligned}$$

I_x and I_y are gradients on x and y direction and can be approximate by $I \otimes \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$ and $I \otimes \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}^T$. I_x and I_y cover small shifts in all direction.

The matrix M is the Hessian matrix and its eigenvalues is proportional to the principal curvatures. If both curvatures are high, it indicates a corner. Let's define the corner response function $R = Det(M) - k(Trace(M))^2 = \sigma_1\sigma_2 - k(\sigma_1 + \sigma_2)^2$ where σ_1, σ_2 are the eigenvalues. When $|R|$ is small, the region is flat. When $R < 0$, the region is edge. When R is large, the region is a corner.

2.2 Shi-Tomasi Corner Detection

Shi-Tomasi proposes to use $R = \min(\sigma_1, \sigma_2)$. A region is considered as a corner if σ_1, σ_2 are above some threshold.

2.3 Scale Invariant Feature Transform (SIFT)

1. Scale-space Extrema Detection It uses Difference of Gaussians (DOG) $D(x,y,\sigma_1,\sigma_2) = G_{x,y,\sigma_1} \cdot I - G_{x,y,\sigma_2} \cdot I$ to approximate Laplacian of Gaussian $\nabla^2(G_\sigma \cdot I)$ which shows the local curvatures. For each octave (different blurring out of the image), the initial image is convolved with different Gaussians. The adjacent Gaussian images are subtracted to get DOG. Maxima and minima of the DOG images are detected by comparing a pixel with its eight neighbors and nine pixels of each of the two adjacent DOG images. If a pixel is a local extrema, it is likely to be a keypoint in that scale.

2. Keypoint Localization Firstly, we want to remove points that have low contrast with the surroundings. We firstly do Taylor expansion on $D(x, y, \sigma_1, \sigma_2) \approx D(0, 0, , \sigma_1, \sigma_2) + \frac{\partial D^T}{\partial v} v + \frac{1}{2} v^t \frac{\partial^2 D^T}{\partial v^2} v, v = (x, y, \sigma_1, \sigma_2)^T$ to approximate the function. We taking derivative to find the location of extremum $\hat{v}$. If the extremum is less than a threshold, the point is removed since there is not high contrast.

Secondly, we want to remove points that are edges rather than corners. We compute 2-by-2 Hessian matrix $H = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}$ and use the eigenvalue to get the curvature and to determine if it's an edge or a corner (same as Harris Corner Detection).

3. Orientation Assignment We create an orientation histogram with 36 bins covering 360 degrees. A neighbourhood is taken around the keypoint location. For each point in the neighbourhood, we calculate the gradient magnitude $m(x, y) = \sqrt{(L(x+1, y) - L(x-1, y))^2 + (L(x, y+1) - L(x, y-1))^2}$ and orientation $\theta(x, y) = \tan^{-1}(\frac{L(x,y+1)-L(x,y-1)}{L(x+1,y)-L(x-1,y)})$. The point is put into bins based on orientation and is weighted by its gradient magnitude. In the end the highest peak in the histogram is considered as the orientation (may contain multiple orientation if other peak is within 80% of the highest peak).

4. Keypoint Descriptor Divide the 16-by-16 neighbourhood around the keypoint into 16 4-by-4 sub-blocks. For each sub-block, 8 bin orientation histogram is created. The vector of total of 128 bin values together becomes the keypoint descriptor. We normalize the vectors and threshold large gradient magnitude to reduce effect of illumination changes and rotation.

5. Keypoint Matching We find the nearest neighbours of each descriptor in another image for keypoint matching.

Speeded-Up Robust Features (SURF) Improve on the speed of feature detection.

2.4 Features from Accelerated Segment Test (FAST)

It's designed for real time applications. The algorithm check if a pixel p is brighter or darker than most of (a threshold) its surrounding pixels. Then, it uses decision tree classifier to classify p as corner and non-corner. It also uses non-maximal suppression to remove adjacent keypoints. This algorithm is several times faster than other corner detectors but is not robust to high levels of noise.

2.5 Binary Robust Independent Elementary Features (BRIEF)

BRIEF is a feature descriptor based on binary string and hamming distance, so it's faster and takes less space than the descriptor in SIFT/SURF. It selects a set of n (128, 256, 512) location pairs in an unique way and do some intensity comparison among the pairs to get the binary string of descriptor.

2.6 ORB

ORB is a fusion of FAST and BRIEF with some modifications. It firstly use FAST to find keypoints at different scales and use Harris corner to find the top N keypoints. It uses the direction from the corner point to the intensity-weighted centroid of the patch around the corner point as the orientation. BRIEF is used for descriptor. Since BRIEF performs poorly with rotation, ORB rotates the patches so that they have the same orientation and then compute the descriptor. ORB also runs a greed search to find the set of location pairs that has binary test with high variance and mean close to 0.5, as well as uncorrelated (rBRIEF).

2.7 Feature Matching

Nearest neighbor search with ratio test finds the features f_1, f_2 in image 2 that have closest distances $d_1 < d_2$ to the feature f in image 1. Then, we do ratio test; if $\frac{d_1}{d_2} < 0.7$, we consider f_1 and f a good matching. We do feature matching from image 1 to image 2 and from image 2 to image 1, then do bi-directional consistency check. That is if f_1 and f match in both direction, we use it as a matching.

RANSAC The second step is to deal with the outliers. We use RANSAC: Random Sample Concensus to cope with large number of outliers. Specifically, two points are select randomly and from a line. We measure the number of points that lie within a distance threshold t (supports). After N repetitions, we output the line with most supports. The success rate p after N repetition is guarenteed. Assume the probability of choosing an inlier is w. The probability of building a correct model is w^n and the probability of not building a correct model in k iterations is $(1-w^n)^k$. Hence, $(1-w^n)^k = 1-p$ and $k = \frac{\log(1-p)}{\log(1-w^n)}$. t and w are choosing manually.

3 Optical Flow

Definition 3.1 (Motion field)**.** Motion field defines the image velocity of point moving in the scene. The mapping between 3D motion in the world and 2D motion on the sensor can be defined.eld. However, in many cases, they are not. For example, a rotating sphere has a non-zero motion field but zero optical flow.e get the velocity v_i of a point on the sensor is

$$v_i = f\frac{(r_0 \times v_0) \times z}{(r_0 \times z)^2}$$

f is the focal length,

r_0 is the length from pin hole to scene point,

v_0 is the velocity of 3D motion

z is the unit vector in the direction perpendicular to sensor

Definition 3.2 (Optical Flow)**.** Optical flow is the motion of brightness pattern in the image.

Remark. Ideally, optical flow is the same as the motion field. However, in many cases, they are not. For example, a rotating sphere has a non-zero motion field but zero optical flow.

3.1 Optical Flow Constraint Equation

Let $I(x, y, t)$ be the intensity of pixel (x, y) at time t. The pixel moves by $(\delta x, \delta y)$ in time δt. We assume the brightness of a point doesn't change, then we have $I(x, y, t) = I(x+\delta x, y+\delta y, t+\delta t)$. And optical flow is defined as $(u, v) = (\frac{\delta x}{\delta t}, \frac{\delta y}{\delta t})$.

Then we do Taylor series expansion on $I(x + \delta x, y + \delta y, t + \delta t)$:

$$\begin{aligned}
I(x + \delta x, y + \delta y, t + \delta t) &= I(x, y, t) + \frac{\partial I}{\partial x}\delta x + \frac{\partial I}{\partial y}\delta y + \frac{\partial I}{\partial t}\delta t + O(\delta x^2 + \delta y^2 + \delta t^2) \\
&= I(x, y, t) + I_x\delta x + I_y\delta y + I_t\delta t \ \text{ (Assume } \delta x, \delta y, \delta t \text{ are small)} \\
I(x, y, t) &= I(x, y, t) + I_x\delta x + I_y\delta y + I_t\delta t \\
I_x\delta x + I_y\delta y + I_t\delta t &= 0 \\
I_x\frac{\partial x}{\partial t} + I_y\frac{\partial y}{\partial t} + I_t &= 0 \ \text{ (as } \delta t \to 0)
\end{aligned}$$

Hence, we get the optical flow constraint equation $I_x u + I_y v + I_t = 0$. I_x is the difference of mean of $\{(x+1, y, t), (x+1, y, t+1), (x+1, y+1, t), (x+1, y+1, t+1)\}$ and $\{(x, y, t), (x, y, t+1), (x, y+1, t), (x, y+1, t+1)\}$. Geometrically, the vector (u, v) lies on the line formed by the constraint equation. Optical flow $f(u, v)$ can be further decomposed into normal flow f_n and parallel flow f_p. Normal flow f_n is in the direction of (I_x, I_y) (which is parallel to the sensor) with size $\frac{|I_t|}{\sqrt{I_x^2+I_y^2}}$. f_p is unknown since it's perpendicular to the sensor, so the information losses when we map from 3D world to the 2D screen. Hence, this system is underdetermined.

3.2 Lucas-Kanade Solution

We assume that for each pixel, motion field and optical flow are constant within a small neighborhood W, i.e. $I_x(k, l)u + I_y(k, l)v + I_t(k, l) = 0, \forall(k, l) \in W$. For W with n entries, we get

$$\begin{bmatrix} I_x(x_1, y_1) & I_y(x_1, y_1) \\ \vdots & \vdots \\ I_x(x_n, y_n) & I_y(x_n, y_n) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} I_t(x_1, y_1) \\ \vdots \\ I_t(x_n, y_n) \end{bmatrix}$$

$$Au = B$$

We can find the closed form least square solution $(A^T A)^{-1} A^T B$. It solution is good if λ_1 and λ_2 of $(A^T A)$ are big and comparable in size (corners or textured regions). Otherwise, the region W may be flat or edge which are not good for optical flow estimation in some directions.

Remark. We assume small optical flow in the above methods. For fast/large motion, we use resolution pyramid to reduce the resolution until it's valid to use the above methods again. Then we use Coarse-to-Fine Estimation Algorithm to calculate the optical flow. We firstly calculate the optical flow $(u, v)^0$ in the lowest-resolution images. Then we refine the optical flow $(\Delta(u, v)^1)$ in the second lowest-resolution images until we refine the optical flow $(\Delta(u, v)^n)$ in the original images. It works since $(u, v)^0$ by definition is small and each residual $(\Delta(u, v)^i)$ is also small.

3.3 Structure from Motion

SFM computes 3D scene structure and camera motion from a sequence of frames. Given sets of image points u_i, v_i in 2D, we want to find corresponding scene points P in 3D when camera position and oriantations are unknow. We have a 3D to 2D mapping under world coordinate $u_i = x^T(P-C)$, $v_i = y^T(P-C)$ where C is the center of the camera and x_i, y_i are the unit vector in the direction of x,y-axis of camera sensor. Fixing world coordinate origin at the centroid of scene points, we get $u_i = x^T P$, $v_i = y^T P$. For N points observed in F frames we have

$$
\begin{aligned}
&W = MS && W \in R^{2F\times N}, M \in R^{2F\times 3}, S \in R^{3\times N}\\
&W = (u_{i,j})_{i,j} \text{ vertically concatenate with } (v_{i,j})_{i,j} && i \in [1,F], j \in [1,N]\\
&M = [x_1, \cdots, x_F, y_1, \cdots, y_F]^T\\
&S = [P_1, \cdots, P_N]
\end{aligned}
$$

W is known and we want to find M and S. We use Tomasi-Kanade Factorization to solve it (there is many solutions). That's doing SVD on W (at most three singular values are non-zero). Then, $W = U_1(\Sigma_1)^{1/2}QQ^{-1}(\Sigma_1)^{1/2}V_1^T = MS, M = U_1(\Sigma_1)^{1/2}Q$ for any 3-by-3 non-singular matrix Q and U_1 and V_1 are singular vectors of top 3 singular values.

3.4 Object Tracking

1. Moving Median (cannot handle significant pixel fluctuations)
2. Guassian mixture model of each pixel
3. Feature detection

4 Projective Geometry and Transformations of 2D

4.1 2D Projective Plane

Definition 4.1 (Homogeneous Representation of Lines)**.** Line is represented by $ax+by+c=0$, hence can be represented by $(a,b,c)^T$. $k(a,b,c)^T, k \neq 0$ is an equivalence class of vectors and is called a **homogeneous** vector. Hence, the set of equivalence classes in $\mathbb{R}^3 - (0,0,0)^T$ forms a projective space $\mathbb{P}^2$.

Definition 4.2 (Homogeneous Representation of Points)**.** Point (x,y) lies on line (a,b,c) iff $ax+by+c=0$, i.e. $k(x,y,1)(a,b,c)^T = 0$. Let $(x_1,x_2,x_3) = k(x,y,1)$ be the homogeneous representation of a point $(\frac{x_1}{x_3}, \frac{x_2}{x_3})$ in $\mathbb{R}^2$.

Remark. Points and lines all have two degree of freedom. For points, it's the coordinates x, y. For line it's the two ratios $\frac{a}{b}, \frac{a}{c}$

Theorem 4.3. *A homogeneous point x lies on a line l iff $x^T l = 0$.*

Theorem 4.4. *The intersection of lines l and l' is the point $x = l \times l'$. The line through x and x' is $l = x \times x'$.*

Definition 4.5 (Ideal point/Point at infinity)**.** is a point with last coordinate $x_3 = 0$.

Definition 4.6 (Line at infinity)**.** The set of ideal point is $\{(\frac{x_1}{x_2}, 1, 0)^T | \frac{x_1}{x_2} \in R\}$. It forms the line at infinity denoted by $I_\infty = (0, 0, 1)^T$ since $(0, 0, 1)\frac{x_1}{x_2}, 1, 0)^T = 0$.

Theorem 4.7. *Parallel lines $(a, b, c)^T$ and (a, b, c') intersect at $(b, -a, 0)^T$.*

Theorem 4.8 (Duality Principle)**.** *For any theorem in 2d projective geometry, there is a dual theorem by interchange the roles of points and lines in the original theorem.*

Definition 4.9 (Conic)**.** is described by second-degree equation in a plane $ax_1{}^2 + bx_1x_2 + cx_2{}^2 + dx_1x_3 + ex_2x_3 + fx_3{}^2 = 0$ or $(x_1, x_2, x_3)C(x_1, x_2, x_3)^T = 0$ with $C = \begin{bmatrix} a & b/2 & d/2 \\ b/2 & c & e/2 \\ d/2 & e/2 & f \end{bmatrix}$.

Remark. C has five degree of freedom (six degree for the symmetric matrix minus one scale). Hence, it can be defined by the null space of matrix of five points:

$$\begin{bmatrix} x_1{}^2 & x_1y_1 & y_1{}^2 & x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_5{}^2 & x_5y_5 & y_5{}^2 & x_5 & y_5 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix} = 0$$

Theorem 4.10. *The line l tangent to C at x is $l = Cx$.*

Definition 4.11 (Dual conic C^*)**.** is the adjoint matrix of C. A line l tangent to C satisfies $l^T C^* l = 0$.

Definition 4.12 (Degenerate conics)**.** is two lines when rank is 2 and one repeated line when rank is 1.

4.2 Revisit Transformations

Theorem 4.13. *Under a point transformation $x' = Hx$, a line is transformed to $l' = H^{-T}l$ and a conic is transformed to $C' = H^{-T}CH^{-1}$ and $C^{*\prime} = HC^*H^T$.*

Projective linear group $PL(n)$ is a quotient group of general linear group $GL(n)$ where scaling doesn't matter. Affine group is a subgroup of $PL(n)$ where last row is $(0, 0, 1)$. (Oriented) euclidean group is a subgroup of affine group where the left-top 2-by-2 matrix is orthogonal (orthonormal).
In addition to define transformation algebraically, we can define them by their invariant.

Definition 4.14 (Isometries)**.** is transformations that preserves Euclidean distance.

$$\begin{bmatrix} \epsilon\cos\theta & -\sin\theta & t_x \\ \epsilon\sin\theta & \cos\theta & t_x \\ 0 & 0 & 1 \end{bmatrix}$$

For $\epsilon = 1$, it's orientation preserving and Euclidean transformation. For $\epsilon = -1$, it's orientation reversing. Isometries' invariant includes length, angle, and area.

Definition 4.15 (Similarity transformation)**.**

$$\begin{bmatrix} s\cos\theta & -s\sin\theta & t_x \\ s\sin\theta & s\cos\theta & t_x \\ 0 & 0 & 1 \end{bmatrix}$$

5 Bibliography

1. UPenn CIS 581
2. UPenn MEAM 620
3. OpenCV Document (`https://docs.opencv.org/4.x/d6/d00/tutorial_py_root.html`)
4. A COMBINED CORNER AND EDGE DETECTOR
5. Distinctive Image Features from Scale-Invariant Keypoints
6. `https://www.youtube.com/channel/UCf0WB91t8Ky6AuYcQV0CcLw`
7. Multiple View Geometry in Computer Vision