Chapter 1

Single Camera Geometry

✠ **Over-parameterised 2D line**:

$$a\hat{x} + b\hat{y} + c = 0$$

where it can be converted to the $\hat{y} = m'\hat{x} + c'$ line by $\quad m' = -\frac{a}{b} \quad , \quad c' = -\frac{c}{b}$

✠ **Homogeneous coordinates for 2D points**:

$$\hat{\mathbf{x}} = \begin{bmatrix} \hat{x} \\ \hat{y} \end{bmatrix} \quad \rightarrow \quad \mathbf{x} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

where $\hat{x} = x/z$ and $\hat{y} = y/z$. Hence $(sx, sy, sz)^{\mathrm{T}}$ describe the same point in 2D space. Similarly, *homogeneous coordinates for 3D points*:

$$\hat{\mathbf{x}} = \begin{bmatrix} \hat{x} \\ \hat{y} \\ \hat{z} \end{bmatrix} \quad \rightarrow \quad \mathbf{x} = \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

✠ **Properties of using Homogeneous coordinates**:

1. Lines and points intersect if $\mathbf{Lx} = 0$
2. Points at infinity are represented by setting $z = 0$ like $\mathbf{x} = (1, 0, 0)^{\mathrm{T}}$ where its a point at infinity in the direction $(1, 0)$
3. Intersection of lines is through cross-product: $\mathbf{x} = \mathbf{L}_1 \times \mathbf{L}_2$
4. Line joining two points is also done through cross-product: $\mathbf{L} = \mathbf{x}_1 \times \mathbf{x}_2$

The **cross-product** is computed as:

$$\begin{aligned} a \times b &= \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ a_i & a_j & a_k \\ b_i & b_j & b_k \end{vmatrix} \\ &= \begin{vmatrix} a_j & a_k \\ b_j & b_k \end{vmatrix} \mathbf{i} - \begin{vmatrix} a_i & a_k \\ b_i & b_k \end{vmatrix} \mathbf{j} + \begin{vmatrix} a_i & a_j \\ b_i & b_j \end{vmatrix} \mathbf{k} \end{aligned}$$

✠ **Rotation + Translation (Euclidean / Isometry)**:

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

transformation matrix in this form is *rotation followed by translation.* 3 degrees of freedom. Properties of Rotational Matrix R includes:

1. $R^{\mathrm{T}} = R^{-1}$
2. $\det(R) = 1$
3. $R_{axis,(\theta_1+\theta_2)} = R_{axis,\theta_1} R_{axis,\theta_2}$
4. all singular values equal 1

✠ **Similarity**:

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

transformation matrix in this form is *scaling/rotation followed by rotation/scaling followed by translation.* 4 degrees of freedom (only one scaling parameter s). For rotation followed by translation followed by scaling, the whole transformation matrix would be the same except that the translation would also be influenced by the scaling parameter $\hat{t}_x = st_x$, $\hat{t}_y = st_y$

✠ **Affine**:

$$\begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

is rotation, shearing, scaling, and translation. Has 6 degrees of freedom.

✠ **Projective** also called **Homography** or **collinearity**:

$$\mathrm{T}_{Prj} = \mathrm{H} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$$

can map any four point to any other 4. Has 8 degrees of freedom; the 9^{th} parameter is a redundant scale parameter. The resultant transformed point (homogeneous coordinates) is a linear combination of the

columns of T_{Prj}:

$$\hat{\mathbf{x}}' = \begin{bmatrix} h_{11}\hat{x} + h_{12}\hat{y} + h_{13} \\ h_{21}\hat{x} + h_{22}\hat{y} + h_{23} \\ h_{31}\hat{x} + h_{32}\hat{y} + h_{33} \end{bmatrix} = \begin{bmatrix} h_1^{\mathrm{T}}\hat{\mathbf{x}} \\ h_2^{\mathrm{T}}\hat{\mathbf{x}} \\ h_3^{\mathrm{T}}\hat{\mathbf{x}} \end{bmatrix} \tag{1.1}$$

Iff $\hat{\mathbf{x}} = \begin{bmatrix} \hat{x} \\ \hat{y} \\ 1 \end{bmatrix}$ because this is *not* a homogeneous coordinate. Here h_r^{T} is the transpose of the r^{th} row of H.

✠ **DLT (Direct Linear Transform Algorithm)** for computing Homography: given $\mathbf{x}_{1\cdots N}$ in the first image and $\mathbf{x}'_{1\cdots N}$ in the second image, find the homography H which best matches these points. Generally the problem is of minimization:

$$\mathrm{H} = \arg\min_{\mathrm{H}'} \sum_{i=1}^{N} ||\mathbf{x}'_i - \mathrm{H}'\mathbf{x}_i||^2$$

Following from Equation 1.1, since we are dealing with homogeneous coordinates, the point $\mathbf{x}_i$ will represent the same point if it is multiplied by a scale factor: $\mathbf{x}_i = s \begin{bmatrix} \hat{x}_i \\ \hat{y}_i \\ 1 \end{bmatrix} = \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} = s\hat{\mathbf{x}}_i$. Now we can state that $s\hat{\mathbf{x}}'_i = \mathrm{H}\mathbf{x}_i$:

$$\mathrm{H}\mathbf{x}_i = \begin{bmatrix} h_1^{\mathrm{T}}\mathbf{x}_i \\ h_2^{\mathrm{T}}\mathbf{x}_i \\ h_3^{\mathrm{T}}\mathbf{x}_i \end{bmatrix} = s\hat{\mathbf{x}}'_i = \mathbf{x}'_i$$

Since $\mathbf{x}'_i$ needs to be parallel to $\mathrm{H}\mathbf{x}_i$ (remember they are homogeneous coordinates representing the same 2D point), their cross-product would be zero:

$$\begin{aligned}
\mathbf{x}'_i \times \mathrm{H}\mathbf{x}_i &= \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ x'_i & y'_i & z'_i \\ h_1^{\mathrm{T}}\mathbf{x}_i & h_2^{\mathrm{T}}\mathbf{x}_i & h_3^{\mathrm{T}}\mathbf{x}_i \end{vmatrix} && = [0] \\
&= \begin{bmatrix} y'_i h_3^{\mathrm{T}}\mathbf{x}_i - z'_i h_2^{\mathrm{T}}\mathbf{x}_i \\ z'_i h_1^{\mathrm{T}}\mathbf{x}_i - x'_i h_3^{\mathrm{T}}\mathbf{x}_i \\ x'_i h_2^{\mathrm{T}}\mathbf{x}_i - y'_i h_1^{\mathrm{T}}\mathbf{x}_i \end{bmatrix} && = [0] \\
&= \begin{bmatrix} 0^{\mathrm{T}} & -z'_i\mathbf{x}_i^{\mathrm{T}} & y'_i\mathbf{x}_i^{\mathrm{T}} \\ z'_i\mathbf{x}_i^{\mathrm{T}} & 0^{\mathrm{T}} & -x'_i\mathbf{x}_i^{\mathrm{T}} \\ -y'_i\mathbf{x}_i^{\mathrm{T}} & x'_i\mathbf{x}_i^{\mathrm{T}} & 0^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix} && = [0] \\
&= \underbrace{\begin{bmatrix} 0^{\mathrm{T}} & -z'_i\mathbf{x}_i^{\mathrm{T}} & y'_i\mathbf{x}_i^{\mathrm{T}} \\ z'_i\mathbf{x}_i^{\mathrm{T}} & 0^{\mathrm{T}} & -x'_i\mathbf{x}_i^{\mathrm{T}} \end{bmatrix}}_{\mathrm{A}_i} \underbrace{\begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix}}_{\mathrm{h}} && = [0]
\end{aligned}$$

In the last step, we ignored the last row because this is true for upto a scale factor, and the last row would just contribute the (non-zero) scale which can be different for different points. Constructing a

joint matrix, we have:

$$\mathrm{Ah} = \begin{bmatrix} 0^{\mathrm{T}} & -z'_1\mathbf{x}_1^{\mathrm{T}} & y'_1\mathbf{x}_1^{\mathrm{T}} \\ z'_1\mathbf{x}_1^{\mathrm{T}} & 0^{\mathrm{T}} & -x'_1\mathbf{x}_1^{\mathrm{T}} \\ \vdots & \vdots & \vdots \\ 0^{\mathrm{T}} & -z'_N\mathbf{x}_N^{\mathrm{T}} & y'_N\mathbf{x}_N^{\mathrm{T}} \\ z'_N\mathbf{x}_N^{\mathrm{T}} & 0^{\mathrm{T}} & -x'_N\mathbf{x}_N^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix}$$

The SVD of a matrix $\mathrm{A} = \mathrm{USV}^{\mathrm{T}}$. The columns of U, $u_1, u_2, \ldots, u_m$, and columns of V, $v_1, v_2, \ldots, v_n$ are orthogonal.

If $rank(\mathrm{A}) = r$, then the column vectors of V, $v_{r+1}, v_{r+2}, \ldots, v_n$ form a basis for the null space of A. If given the SVD, this fact is useful in solving $\mathrm{A}\mathbf{x} = 0$.

Remember that orthogonal matrix are geometrically either rotation or reflection matrices. Essentially, the multiplication $\mathrm{A}\mathbf{x}$ is (1) first a rotation (or reflection) around the origin by V^{T}; (2) non-uniform scaling by the values by S and possible change in dimensions; and, finally, (3) rotation (or reflection) around the origin by U. Note, the determinant of A dictates the scaling of area - if determinant is s, a unit area would become s after transformation.

Given the constraint $\| \mathrm{h} \| = 1$, atleast 4 corresponding points should be given to solve $\mathrm{Ah} = 0$ and no 3 should be collinear. One way to find $\mathrm{Ah} = 0$, i.e. the nullspace of A, is to find the SVD of A, and take the last column of V. To explain this we need to look at the SVD. Since S is in the decreasing order of singular values along the diagonal. Since we are seeking a vector for h which will be multiplied by the smallest singular value (other than 0), the last column of V would be the perfect fit.

✠ **Invariants** are the properties that remain constant under a certain transformation:

$$\#\text{ invariants } = \text{ D.O.F structure } - \text{ D.O.F transformation}$$

1. **Euclidean invariants** include length, angles, areas, etc.
2. **Similarity invariants** include ratio of lengths, angles, ratio of areas, etc.
3. **Affine invariants** include ratio of length of parallel lines, angles, ratio of areas, etc.
4. **Projective invariants** cross-ratio, collinearity.

Notice that the list of invariants for a particular transformation includes all the invariants of the transformations that follow below it. Remember, putting shapes in **Canonical frames** helps detect shapes/features.

✠ **Vanishing point** in perspective projection is the point where different parallel lines meet at different vanishing points. The line joining all vanishings points (from parallel lines on the same plane) is the **horizon**.

Remember, line through *focal point* maps to a point; plane through *focal point* maps to a line.

Types of lens aberrations:

1. **Vignetting**: darkness around edges due to lens construction.
2. **Chromatic aberration**: defocussing of some wavelengths.
3. **Barrel distortion**: decreasing magnification as we move away from the optical axis.

✠ The distance between the optical center and the principal point is the **focal length** f. Smaller the f, larger the field of view:

$$\hat{y}_c = f\frac{\hat{Y}}{\hat{Z}}$$

Where $\hat{y}_c$ is the height on the image plane, $\hat{Y}$ is the original height, and $\hat{Z}$ is the distance of the object from the optical center.

✠ The **Intrinsic matrix K** maps camera coordinates to image coordinates $x_i = \mathbf{K}x_c$; x_c are camera coordinates; x_i are image coordinates:

$$\text{given } \mathbf{K} = \begin{bmatrix} f_x & \alpha & o_x \\ 0 & f_y & o_y \\ 0 & 0 & 1 \end{bmatrix}$$

$$x_i = \mathbf{K}\left[\mathbf{I}|\mathbf{0}\right] X_c$$

$$\begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} = \left[\begin{array}{ccc|c} f_x & \alpha & o_x & 0 \\ 0 & f_y & o_y & 0 \\ 0 & 0 & 1 & 0 \end{array}\right] \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix}$$

Here the f_x, f_y are both focal lengths (since the pixel is not square), α is the skew term (used when the camera plane is not orthogonal to the optical axis); and o_x, o_y are coordinates for the principal point on the image plane (lets say, if the image starts from the top left corner).

✠ The **Extrinsic matrix** $\mathbf{T}_{W\to C}$ maps world-coordinates (lets say the obj.s center) to the camera center coordinates. $x_i = \mathbf{K}\left[\mathbf{I}|\mathbf{0}\right]\mathbf{T}_{W\to C}X_c$:

$$\begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} = \underbrace{\left[\begin{array}{ccc|c} f_x & \alpha & o_x & 0 \\ 0 & f_y & o_y & 0 \\ 0 & 0 & 1 & 0 \end{array}\right] \underbrace{\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\mathbf{T}_{W\to C}}}_{\mathbf{P} \text{ the } \textbf{camera matrix}} \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix}$$

The extrinsic matrix $\mathbf{T}_{W\to C}$ has 6 degrees of freedom (euclidean matrix).

✠ Compensation for **Radial Distortion** is usually done in camera coordinates using a polynomial model.

✠ **Calibration** is the process of estimating the 5 intrinsic, 6 extrinsic, and the 2 radial distortion parameters. We start with a least square solution and compute a solution which minimizes the *image distance*.

✠ **Orthographic projection happens** when rays from the scene travel parallel to the optical axis (a situation common when the object in question is far away):

$$\begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & 0 & o_x \\ 0 & f_y & 0 & o_y \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix}$$

✠ When **viewing frontoplanar scene at known distance** i.e. fixed distance D, the camera matrix can be simplified to:

$$\begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} = \left[\begin{array}{ccc|c} f_x & 0 & o_x & 0 \\ 0 & f_y & o_y & 0 \\ 0 & 0 & 1 & 0 \end{array}\right] \begin{bmatrix} r_{11} & r_{12} & 0 & t_x \\ r_{21} & r_{22} & 0 & t_y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ D \\ 1 \end{bmatrix}$$

. . . can also be truncated to a smaller 2×3 matrix.

✠ When **viewing frontoplanar scene at unknown distance** i.e. if the focal distance of the camera is unknown or the distance to the plane is unknown, then the images are also now related to each other by the unknown scale factor m.

✠ When **viewing tilted planar scene at known distance** is approximated with an affine transform and works only when the object is far away as compared to its size:

$$\begin{bmatrix} x_i \\ y_i \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ 1 \end{bmatrix}$$

Note the affine transform preserves parallelism, but a perspective view of a tilted scene doesn't - hence this is not a good approximation.

✠ For a **planar scene with arbitrary camera tilt and pan** can be described by a projective transformation or homography. Since the scene is planar we can say $Z_c = 0$:

$$\begin{aligned} \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} &= \left[\begin{array}{ccc|c} f_x & \alpha & o_x & 0 \\ 0 & f_y & o_y & 0 \\ 0 & 0 & 1 & 0 \end{array}\right] \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ 0 \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} f_x & \alpha & o_x \\ 0 & f_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & t_x \\ r_{21} & r_{22} & t_y \\ r_{31} & r_{32} & t_z \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ 1 \end{bmatrix} \end{aligned}$$

✠ Two images of any planar scene are related by a homography: $\mathrm{H}_2^{-1} H_1$

Chapter 2

Two or more Cameras

✠ **Perspective projection** converts from 3D to 2D.

✠ Stereo vision has the problems of **calibration** i.e. establishing the geometric relationship between the cameras; **correspondence** i.e. finding matching points between the images; and **reconstruction** i.e. estimating the 3D location of a point.

✠ Given two cameras C_1 and C_2, and the transformation relating the first to the second camera, lets say, $\mathrm{T}' = [\mathbf{R}|\mathbf{t}]$, we can concentrate on a certain point $\mathbf{X}_c$ viewable in both the cameras. Suppose it maps to a point $\mathbf{x}$ in C_1 and $\mathbf{x}'$ in C_2. Given that $\mathbf{X}_c$ lies on line L projecting from C_1, we know that $\mathbf{x}'$ in C_2 would lie somewhere along the projection of the line L in the second camera, L'. In short, a point in C_1, projects a line L' in C_2 - the exact point itself in C_2 would lie on the line L'. The line L' is know as the **epipolar line**. It helps us reduce a 2D search for matching points to a 1D search over a line.

✠ The **epipole** is the image of the optical center of the other camera. It can be made by connecting a line between the two optical centers $\mathbf{O}$ and $\mathbf{O}'$ of C_1 and C_2 respectively. It is also the point where all epipolar lines intersect. It might not be necessarily in the image plane.

✠ Given the following information about the two cameras, we need to construct a relationship between the position of the point $\mathbf{x}$ and $\mathbf{x}'$ in the first and second camera respectively:

	C_1	C_2
Image point	$\mathbf{x}$	$\mathbf{x}'$
Optical center	$\mathbf{O}$	$\mathbf{O}'$
Intrinsic Matrix	$\mathbf{K}$	$\mathbf{K}'$
Extrinsic Matrix	$\mathbf{T} = \mathbf{I}$	$\mathbf{T}' = [\mathbf{R}\|\mathbf{t}]$
Ray vector from Optical center to $\mathbf{X}_c$	$\mathbf{OX}_c = k_1\mathbf{Rx}$	$\mathbf{O}'\mathbf{X}_c = k_2\mathbf{x}'$

Also given is the distance between the two optical centers which is exactly the translation in $\mathbf{T}'$: $\mathbf{OO}' = \mathbf{t}$.

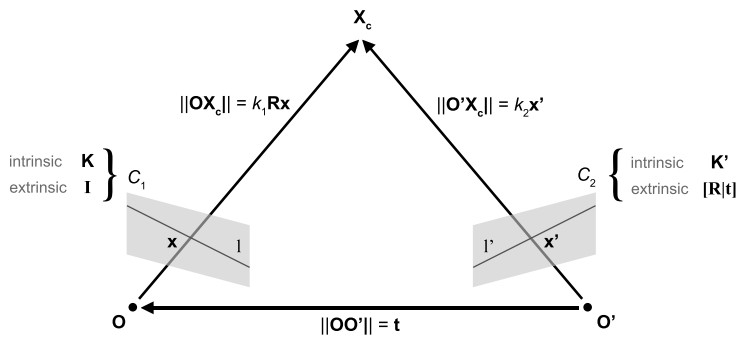

Figure 2.1: The setup for the *Essential Matrix* formulation.

Since $\mathbf{OO'}$, $\mathbf{OX}_c$, and $\mathbf{O'X}_c$ are coplanar, they should satisfy the following condition, if these vectors are transferred to the co-ordinate frame of C_2:

$$\begin{aligned}
\mathbf{O'X}_c.(\mathbf{OO'} \times \mathbf{OX}_c) &= 0 \\
\mathbf{x'}^{\mathrm{T}}.(\mathbf{t} \times \mathbf{Rx}) &= 0 \qquad \text{throwing away the constants } k_1, k_2 \\
\mathbf{x'}^{\mathrm{T}}. \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ t_x & t_y & t_z \\ (\mathbf{Rx})_x & (\mathbf{Rx})_y & (\mathbf{Rx})_z \end{vmatrix} &= 0 \\
\mathbf{x'}^{\mathrm{T}}. \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix} \mathbf{Rx} &= 0 \\
\mathbf{x'}^{\mathrm{T}} \mathbf{t}_{\times} \mathbf{Rx} &= 0 \\
\mathbf{x'}^{\mathrm{T}} \mathbf{Ex} &= 0
\end{aligned} \tag{2.1}$$

The matrix $\mathbf{E}$ gives the relationship between the two points in the image. It is called the **Essential Matrix**. Its a 3×3 matrix; with 6 D.O.F - 3 translational, 3 rotational; and is rank 2.

✠ The epipolar line $\mathbf{l}$ in the first camera, formed by the projection of $\mathbf{x'}$ is given by:

$$\mathbf{l} = \mathbf{x'}^{\mathrm{T}} \mathbf{E}$$

Similarly the epipolar line $\mathbf{l'}$ in the second camera, formed by the projection of $\mathbf{x}$ is given by (transpose the whole Equation 2.1):

$$\mathbf{l'} = \mathbf{x}^{\mathrm{T}} \mathbf{E}^{\mathrm{T}}$$

To find the epipole, we need to find the point which satisfies Equation 2.1 for all epipolar lines. Since it should solve for all points then:

1. Epipole in C_1 $\quad \mathbf{e_1} = \mathrm{NULL}(\mathbf{E})$
2. Epipole in C_2 $\quad \mathbf{e_2} = \mathrm{NULL}(\mathbf{E}^{\mathrm{T}})$

✠ **Fundamental Matrix** solves the system in the *Essential matrix* when the system is *uncalibrated* i.e. we don't know the intrinsic and extrinsic parameters (remember $\mathbf{x}_i = \mathbf{K}\mathbf{x}_c$ and $\mathbf{x'}_i = \mathbf{K'}\mathbf{x'}_c$). Moving Equation 2.1 to image coordinates:

$$\begin{aligned}
\mathbf{x'}_c^{\mathrm{T}}\mathbf{E}\mathbf{x}_c &= 0 \\
(\mathbf{K'}^{-1}\mathbf{x'}_i)^{\mathrm{T}}\mathbf{E}(\mathbf{K}^{-1}\mathbf{x}_i) &= 0 \\
\mathbf{x'}_i^{\mathrm{T}}\mathbf{K'}^{-T}\mathbf{E}\mathbf{K}^{-1}\mathbf{x}_i &= 0 \\
\mathbf{x'}_i^{\mathrm{T}}\mathbf{F}\mathbf{x}_i &= 0
\end{aligned}$$

Where $\mathbf{F}$ is the Fundamental matrix, which maps image coordinates in one camera to epipolar lines in the other camera. $\mathbf{F}$ is 3×3 matrix; rank 2 with 7 independent parametes. It doesn't have 9 degrees of freedom since one is sacrificed for scale ambiguity associated with homogeneous representation; and the other is lost from not being full rank.

✠ The **Eight point algorithm** is used to find the *Fundamental matrix*. It needs atleast 8 points, with no four points lying on a single plane:

$$\begin{aligned}
\begin{bmatrix} x' & y' & 1 \end{bmatrix} \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} &= 0 \\
\begin{bmatrix} f_{11}xx' + f_{12}yx' + f_{13}x' + f_{21}xy' + f_{22}yy' + f_{23}y' + f_{31}x + f_{32}y + f_{33} \end{bmatrix} &= 0 \\
\begin{bmatrix} xx' & yx' & x' & xy' & yy' & y' & x & y & 1 \end{bmatrix} \mathbf{f} &= 0 \qquad (2.2)
\end{aligned}$$

Where $\mathbf{f}$ is the vectorized form of $\mathbf{F}$ in the row major order. If given atleast eight points, we can stack them as:

$$\mathbf{A}\mathbf{f} = \begin{bmatrix} x_1x'_1 & y_1x'_1 & x'_1 & x_1y'_1 & y_1y'_1 & y'_1 & x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_Nx'_N & y_Nx'_N & x'_N & x_Ny'_N & y_Ny'_N & y'_N & x_N & y_N & 1 \end{bmatrix} \mathbf{f} = 0$$

Using the same technique used in DLT, we can take the SVD of $\mathbf{A}$ and pick the last column of V as a solution for $\mathbf{f}$.

✠ Problem with the solution for Fundamental Matrix:

1. Usually the solution for $\mathbf{F}$ is not a rank 2 matrix, because of errors in the data. This is usually done by getting back $\mathbf{F}$ by setting the last singular value to 0.
2. For a more exact solution the error term should depend on the distance of $\mathbf{X}_c$'s from the epipolar lines, rather than minimizing the product terms in Equation 2.2.
3. The matrix $\mathbf{A}$ is badly scaled.
4. ... finding atleast 8 corresponding points without knowing the epipolar geometry is hard - yet we are taking this as a starting point to calculate our epipolar geometry.

✠ **Corrspondence** is the problem of finding matching points in a set of images. The features usually selected for establishing such a correspondence should be tolerant to:

1. noise
2. changes in illumination
3. uniform scaling
4. rotation
5. minor changes in viewing direction

To make this problem tractable we make the following assumptions:

1. Most scene points are visible in both images
2. Corresponding points have similar neighborhoods

They turn out to be true if the distance of the scene is much greater than the **baseline** of the stereo system (distance between the two cameras).

✠ **SIFT - scale invariant feature transform** is a method to find robust image keypoints. This is done by encoding the properties of the neighborhood in a descriptor vector. We can then match points defining an appropriate distance metric.

It leverages the idea that images have different structures at various different scales:

1. Make a **scale space** which is simply a stack of images (the original one at the bottom) produced by iteratively convolving the top-most image in the stack by a fixed Gaussian (blurring) - these represent structures at increasingly large scales.
2. Since image gradients are more stable features than raw luminance, we take the difference of consecutive images in the scale space (**DOG - difference of Gaussians**):

$$\begin{aligned} & G(x,y,\sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2} \\ \text{DOG} \quad \rightarrow \quad & D(x,y,\sigma) = G(x,y,\sigma) * G(x,y,\sigma) * I(x,y) - G(x,y,\sigma) * I(x,y) \\ & \qquad = (G(x,y,k\sigma) - G(x,y,\sigma)) * I(x,y) \end{aligned} \tag{2.3}$$

   We can do step 2.3 because both convolution and difference are linear operators.

3. Now find the extrema in DOG space i.e. points larger (or smaller) than 26 neighbors in the DOG stack.
4. Now we need localize keypoints to subpixel accuracy, which is done by Taylor expansion up to quadratic terms of the scale space function $D(x, y, \sigma)$ around the potential keypoint (taking $\mathbf{x} = (x, y, \sigma)$):

$$D(\mathbf{x}) = D + \frac{\partial D^{\mathrm{T}}}{\partial \mathbf{x}}\mathbf{x} + \frac{1}{2}\mathbf{x}^{\mathrm{T}}\frac{\partial^2 D}{\partial \mathbf{x}^2}\mathbf{x}$$

   where D and its derivatives are evaluated at the keypoint and $\mathbf{x}$ is the *offset* from this keypoint. Taking the derivative of this function with respect to this offset and setting the resulting expression

to zero allows us to estimate the position of the keypoint up to sub-pixel accuracy:

$$0 = \frac{\partial D^{\mathrm{T}}}{\partial \mathbf{x}} + \frac{\partial^2 D}{\partial \mathbf{x}^2}\hat{\mathbf{x}}$$

$$\underbrace{\begin{bmatrix} \frac{\partial^2 D}{\partial x^2} & \frac{\partial^2 D}{\partial yx} & \frac{\partial^2 D}{\partial \sigma x} \\ \frac{\partial^2 D}{\partial yx} & \frac{\partial^2 D}{\partial y^2} & \frac{\partial^2 D}{\partial \sigma y} \\ \frac{\partial^2 D}{\partial \sigma x} & \frac{\partial^2 D}{\partial \sigma y} & \frac{\partial^2 D}{\partial \sigma^2} \end{bmatrix}}_{\text{hessian of } D(\mathbf{x})} \begin{bmatrix} x \\ y \\ \sigma \end{bmatrix} = - \begin{bmatrix} \frac{\partial D}{\partial x} \\ \frac{\partial D}{\partial y} \\ \frac{\partial D}{\partial \sigma} \end{bmatrix}$$

$$\hat{\mathbf{x}} = -\frac{\partial^2 D^{-1}}{\partial \mathbf{x}^2}\frac{\partial D}{\partial \mathbf{x}}$$

5. Rejection of unstable image structures (like edges instead of corners) by singular values of image structure tensor $\mathbf{H}$ (corner: both high eigenvalues; edge: one large one small; homogeneous: both small):

$$\mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}$$

6. Gradients binned by orientation to form a histogram, and orientation of feature is taken as the peak of the histogram (if more than one peak exists, split into two keypoints with different orientations). Since the descriptor is computed relative to this orientation, it achieves rotational invariance.

✠ Now the task comes down to finding the keypoint *correspondence*. Once *SIFT* gives us stable feature points, they need to be characterized by their appearance. This is done by a histogram of image gradients in a region around a keypoint:

1. Calculate the image gradients around a keypoint, weighted by a gaussian.
2. Pool over a space to form 8 bin orientation histograms in a 4×4 grid.
3. The values of histograms are concatenated to form a 128 elements feature vector f $(8 \times 4 \times 4)$. Normalize to unit length to make them invariant to changes in contrast.

The feature vector f is matched to feature vectors $\mathbf{F}'$ in the other image around a certain region. The similarity measure is the Euclidean distance between the SIFT descriptors.

✠ **RANSAC - Random sample consensus** is a method to fit a model to the data corrupted by outliers. The algorithm is to do the following steps multiple times:

1. Select a random minimal subset of points (i.e. 2) that defines a line.
2. Fit a line through this subset of points.
3. For all remaining points calculate the squared distance from the fitted line.
4. Divide the points into inliers (close to the line) and outliers (far away).
5. Count the proportion of inliers to outliers.

The step that results in the highest proportion of inliers is the best set of points which would fit the model. Now the actual model/line can be estimated by a least squares method using only the inliers. This can be also be used to estimate the homographies between pairs of points between two images.

✠ To estimate the *Fundamental matrix* using *RANSAC*, for each iteration we will need to choose 8 pairs of points in the two images. Now a pair would be counted as an inlier if the epipolar line in the second image, produced by the point in the first image, falls close to the point in the second image (and vice versa). The random selection of pairs is done from the initial correspondence. Once the best set of inliers is found, the *Fundamental matrix* is re-estimated.

✠ **Sparse stereo reconstruction** is problem of finding depth of certain feature points, and using them to reconstruct a scene. As can be seen from Figure 2.1, the point of intersection of $\mathbf{OX}_c$ and $\mathbf{O'X}_c$ would be the feature's location is space. Due to noise in the system, these lines might not intersect and we would need to take the mid-point of the line segment which gives the smallest distance between these two lines. This is given by:

$$t + k_2\mathbf{x}' = k_1\mathbf{Rx} + k_3(\mathbf{Rx} \times \mathbf{x}')$$

This is the known case where the intrinsic and extrinsic parameters are known.

In the case we only know the intrinsic parameters, we will first need to estimate the *essential matrix* given some correct point matches. Using the estimated *essential matrix* we will be able estimate the extrinsic parameters (up to a scaling factor of $\mathbf{t_x}$) by breaking $\mathbf{E}$ into its constituent parts, $\mathbf{E} = \mathbf{t_x R}$. This will establish the relationship between the two cameras:

1. $\mathbf{E} = ULV^{\mathrm{T}}$
2. $\mathbf{t_x} = ULWU^{\mathrm{T}}$ where $W = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix}$
3. $\mathbf{R} = UW^{-1}V^{\mathrm{T}}$
4. in the absence of scaling information, set $|\mathbf{t_x}| = 1$

✠ **Image rectification** warps the two camera images so that all points on the first line of the first image have matches on the first scan line of the second image and vice versa (horizontal *epipolar lines*). Here, the horizontal difference in projection of the same point in the rectified image is called the **disparity**.

Polar rectification is done by polar parameterization around the *epipole*: x-axis in the *polar rectified* image gives the distance from the *epipole* and y-axis gives the angle from the *epipole*.

Chapter 3

Probability, Learning and Inference

✠ **Random Variable** X denotes an uncertainty in an event. **Probability distribution** $\Pr(X)$ captures that uncertainty. Both discrete (e.g. roll of a dice), and continuous (e.g. time to finish a 2 hours exam) random variables exist. Both exist in finite (e.g. roll of a dice) and infinite (e.g. velocity of a car if we ignore nothing travels faster than speed of light) flavors. In case of a discrete random variable, sum of all probabilities should be 1: $\sum \Pr(X) = 1$. In case of a continuous random variable, it is visualized using a probability density function/PDF (which shows the relative propensity of X to take a ceratin value) whose integral is 1: $\int \Pr(X)dX = 1$.

✠ **Hinton diagrams** are used to represent distributions of discrete random variables using squares of different sizes indicating their relative probability.

✠ When considering multiple random variables, $X_1, X_2, \ldots, X_n$, we can consider their **joint probability distribution**: $\Pr(X_1, X_2, \ldots, X_n)$, which is read as "the probability of $X_1 X, 2, \ldots,$ *and* X_n". Note that these random variables can be a mixture of continuous and discrete. Regardless, the total probability distribution by summing over the discrete random variables, and integrating over the continuous ones should be 1:

$$\sum_{X_n} \cdots \sum_{X_{a+2}} \sum_{X_{a+1}} \Big(\int \ldots \int \int \Pr(X_1, X_2, \ldots, X_n)\, dX_1 dX_2 \ldots dX_a \Big) = 1$$

where the continuous random variables are from 1 to a and discrete ones are from $a+1$ to n.

✠ Given a *joint probability distribution*, **Marginalization** is the process of recovering the (joint) probability distribution of a subset of those random variables. The recovered probability distribution is called the **Marginal Distribution**. For example we can recover the *marginal distribution* $\Pr(X)$ from the *joint distribution* $\Pr(X, Y)$ by marginalizing over the random variable Y i.e. the probability distribution of X when we ignore Y:

$$\Pr(X, Y) = \sum_{W} \Big(\int \Pr(X, Y, W, Z)\, dZ \Big)$$

if W was a discrete random variable, and Z was a continuous one.

✠ The **Conditional probability** in a *joint distribution* is the relative probability of, let's say, X if Y is fixed to y^*. It is written as $\Pr(X|Y=y^*)$ and read as "the probability of X given $Y=y^*$".

It is computed by normalizing the probability distribution of a certain slice of the joint distribution:

$$\begin{aligned}\Pr(X|Y=y^*) &= \frac{\Pr \text{ of } X \text{ and } Y=y^*}{\text{total Pr when } Y=y^*} \\ &= \frac{\Pr(X,Y=y^*)}{\int \Pr(X,Y=y^*)dX} = \frac{\Pr(X,Y=y^*)}{\Pr(Y=y^*)} \\ \Pr(X|Y) &= \frac{\Pr(X,Y)}{\Pr(Y)} \qquad \textit{this is an alternate form of writing the eq. above}\end{aligned}$$

An important corollary of this formula is that in case of more than two variables, we may repeatedly take conditional probabilities to divide up the joint probability distribution into a product of terms:

$$\begin{aligned}\Pr(W,X,Y,Z=z^*) &= \Pr(W,X,Y|Z)\Pr(Z) \\ &= \Pr(W,X|Y,Z)\Pr(Y|Z)\Pr(Z) \\ &= \Pr(W|X,Y,Z)\Pr(X|Y,Z)\Pr(Y|Z)\Pr(Z)\end{aligned}$$

✠ The idea of **Independence** between random variables states that a certain random variable X tells us nothing about Y - they are completely un-correlated. This will be evident when we compute the conditional probability, let's say, of X given that $Y=y^*$, and find that it remains the same regardless of the values of y^*:

$$\begin{aligned}\Pr(X|Y) &= \Pr(X) \\ \Pr(Y|X) &= \Pr(Y)\end{aligned}$$

$$\begin{aligned}\textit{moreover, } \Pr(X,Y) &= \Pr(X|Y)\Pr(Y) \\ &= \Pr(X)\Pr(Y)\end{aligned}$$

✠ Using:

$$Pr(X,Y) = \Pr(Y|X)\Pr(X) = \Pr(X|Y)\Pr(Y)$$

the **Bayes' rule** refers to these steps:

$$\begin{aligned}\Pr(Y|X)\Pr(X) &= \Pr(X|Y)\Pr(Y) \\ \Pr(Y|X) &= \frac{\Pr(X|Y)\Pr(Y)}{\Pr(X)} && (3.1) \\ &= \frac{\Pr(X|Y)\Pr(Y)}{\int \Pr(X,Y)dY} && \text{using } \textit{marginalization} \quad (3.2) \\ &= \frac{\Pr(X|Y)\Pr(Y)}{\int \Pr(X|Y)\Pr(Y)dY} && \text{using } \textit{conditional probability} \quad (3.3)\end{aligned}$$

The steps in Equation 3.1 to 3.3 are known as the *Bayes rule*.

The terms are named as: $\Pr(Y|X)$ is the ***Posterior*** probability; $\Pr(X|Y)$ is the ***Likelihood***; $\Pr(Y)$ is the ***Prior*** (what we know about Y before knowing X); and $\Pr(X)$ is the ***Evidence***.

✠ The problems in Vision using the learning approach usually want to infer the state of the world $\mathbf{y}^*$ given some visual data $\mathbf{x}^*$. When the state of the world, $\mathbf{y}^*$ is continuous, we call this inference **Regression**. When it is discrete we call this **Classification**.

✠ A certain visual data $\mathbf{x}^*$ is still ambiguous to convincingly define state $\mathbf{y}^*$. Hence we work with $\Pr(\mathbf{y}^*|\mathbf{x}^*)$, that is the probability distribution of the state after observing the visual data. The prior here, $\Pr(\mathbf{y}^*)$ indicates that some states are observed more than others. This process of observing the data and refining our understanding about the state from the prior to the posterior probability is called ***Inference***.

✠ A **posterior function** takes $\mathbf{x}^*$ as the input and returns $\Pr(\mathbf{y}^*|\mathbf{x}^*)$, the posterior distribution as the output. Remember, *the posterior function is not a probability function.*

✠ Approaches to *inference* (in all images, each row belongs to a particular state $\mathbf{y}^*$, and each column belong to a particular visual data $\mathbf{x}^*$):

1. The ***Discriminative*** approach directly models the *posterior function* $\Pr(\mathbf{y}^*|\mathbf{x}^*)$. Given the visual data $\mathbf{x}^*$ it directly outputs $\Pr(\mathbf{y}^*|\mathbf{x}^*)$. In this model the columns of the image are probability distributions and sum to one.
2. The ***Generative joint*** approach models the *joint distribution* $\Pr(\mathbf{y}^*, \mathbf{x}^*)$. Given the visual data $\mathbf{x}^*$ it computes the posterior using Bayes' rule:

   $$\Pr(\mathbf{y}^*|\mathbf{x}^*) = \frac{\Pr(\mathbf{y}^*, \mathbf{x}^*)}{\Pr(\mathbf{x}^*)}$$

   Here the entire image $\Pr(\mathbf{y}^*, \mathbf{x}^*)$, of course, is a PDF. We create the posterior function indirectly by normalizing the columns of this joint distribution.
3. The ***Generative likelihood*** approach models the *likelihood* distribution $\Pr(\mathbf{x}^*|\mathbf{y}^*)$ and the *prior* distribution $\Pr(\mathbf{y}^*)$. Now given the visual data $\mathbf{x}^*$, it computes the posterior:

   $$\Pr(\mathbf{y}^*|\mathbf{x}^*) = \frac{\Pr(\mathbf{x}^*|\mathbf{y}^*)\Pr(\mathbf{y}^*)}{\Pr(\mathbf{x}^*)}$$

   In this model each row in the image of $\Pr(\mathbf{x}^*|\mathbf{y}^*)$ sums to 1. The posterior is generated by weighting the rows by the prior and then normalizing columns.

   Note that in case of *classification* we can model the **class conditional likelihoods**, $\Pr(\mathbf{x}^*|\mathbf{y}^* = k)$ for each class separately

✠ Regardless of the *inference* approach adopted, in *Learning* we estimate the posterior function over the parameters θ. Estimating these parameters is known as ***Learning***. The shape of the posterior function will change with the choice of parameters θ.

✠ The estimation is of θ is done using training data $\mathbf{X}, \mathbf{Y}$. Since this training data might not be enough, the best way to estimate a posterior distribution $\Pr(\theta|\mathbf{X}, \mathbf{Y})$ over the parameters θ given the training data. In other words, in the Bayesian approach, we treat the model parameters θ as random variables that are themselves uncertain:

1. Discriminative:
$$\Pr(\theta|\mathbf{X}, \mathbf{Y}) = \frac{\Pr(\mathbf{Y}|\mathbf{X}, \theta)\Pr(\theta)}{\Pr(\mathbf{Y}|\mathbf{X})}$$
2. Generative joint:
$$\Pr(\theta|\mathbf{X}, \mathbf{Y}) = \frac{\Pr(\mathbf{Y}, \mathbf{X}|\theta)\Pr(\theta)}{\Pr(\mathbf{X}, \mathbf{Y})}$$
3. Generative likelihood:
$$\Pr(\theta|\mathbf{X}, \mathbf{Y}) = \frac{\Pr(\mathbf{X}|\mathbf{Y}, \theta)\Pr(\theta)}{\Pr(\mathbf{X}|\mathbf{Y})}$$

✠ Since each set of parameter θ will give a different posterior function, the **Bayesian approach** advocates marginalizing over all possible settings for the parameters θ:

1. Discriminative:
$$\Pr(\mathbf{y}^*|\mathbf{x}^*, \mathbf{X}, \mathbf{Y}) = \int \Pr(\mathbf{y}^*|\mathbf{x}^*, \theta)\Pr(\theta|\mathbf{X}, \mathbf{Y})d\theta$$
2. Generative joint:
$$\Pr(\mathbf{y}^*, \mathbf{x}^*|\mathbf{X}, \mathbf{Y}) = \int \Pr(\mathbf{y}^*, \mathbf{x}^*|\theta)\Pr(\theta|\mathbf{X}, \mathbf{Y})d\theta$$
3. Generative likelihood:
$$\Pr(\mathbf{x}^*|\mathbf{y}^*, \mathbf{X}, \mathbf{Y}) = \int \Pr(\mathbf{x}^*|\mathbf{y}^*, \theta)\Pr(\theta|\mathbf{X}, \mathbf{Y})d\theta$$

✠ Sometimes the *Bayesian approach* might be intractable, because the resulting equation might not be in closed form. Rather than integrating over all parameters, we can select the most probable parameters $\hat{\boldsymbol{\theta}} = \arg\max_\theta \Pr(\theta|\mathbf{X}, \mathbf{Y})$. The distributions over the chosen parameters is now:

$\Pr(\mathbf{y}^*|\mathbf{x}^*, \mathbf{X}, \mathbf{Y}, \hat{\boldsymbol{\theta}})$	*Discriminative*
$\Pr(\mathbf{y}^*, \mathbf{x}^*|\mathbf{X}, \mathbf{Y}, \hat{\boldsymbol{\theta}})$	*Generative joint*
$\Pr(\mathbf{x}^*|\mathbf{y}^*, \mathbf{X}, \mathbf{Y}, \hat{\boldsymbol{\theta}})$	*Generative likelihood*

This is the ***Maximum a posteriori (MAP)*** estimation of θ.

Chapter 4

Common Probability Distributions

✠ The common probability distributions:

Distributions to describe **data and world state** $\mathbf{x}^*, \mathbf{y}^*$	**Domain**	*Distributions* to model uncertainty in **Parameters** θ
Bernoulli	$z \in \{0, 1\}$	Beta
Categorical	$z \in \{1, 2, \ldots, K\}$	Dirichlet
Univariate normal	$z \in \mathbb{R}$	Normal inverse gamma
Multivariate normal	$\mathbf{z} \in \mathbb{R}^+$	Normal inverse Wishart

✠ **Bernoulli** distribution is a *discrete* distribution that models binary trials $y \in \{0, 1\}$ (1 is for success usually). Plus $\lambda \in [0, 1]$:

$$\Pr(y) = \lambda^y(1-\lambda)^{1-y} = \text{Bern}_y[\lambda]$$

It could be used to model data where pixels are thresholded, and states such as a binary classifier.

✠ The **Beta** distribution is defined on $\lambda \in [0, 1]$ and has parameters $(\alpha, \beta) \in [0, \infty)$ whose relative values determine the expectation $E[\lambda] = \alpha/(\alpha + \beta)$. As the absolute values of (α, β) increase the concentration around $E[\lambda]$ increases:

$$\Pr(\lambda) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}\lambda^{\alpha-1}(1-\lambda)^{\beta-1} = \text{Beta}_\lambda[\alpha, \beta]$$

✠ The **Categorical** distribution is a discrete distribution that determines the probability of observing one of K possible outcomes (*Bernoulli* is just a special case of this). This is useful when pixel data is banded into discrete levels, or we are dealing with a K-way classifier:

$$\Pr(y = k) = \lambda_k = \text{Cat}_y[\lambda_1, \lambda_2, \ldots, \lambda_K]$$

where $\sum_{k=1}^{K} \lambda_k = 1$.

✠ The **Dirichlet** distribution is defined over K continuous values $\lambda_1, \ldots, \lambda_K$, following the rules in the *Categorical* distribution. It is defined using K parameters $\alpha_1, \ldots, \alpha_K$ which can take positive values

(their relative values define expectations: $E[\lambda_1], \ldots, E[\lambda_k]$):

$$\Pr(\lambda_1, \ldots, \lambda_K) = \frac{\Gamma[\sum_{k=1}^{K} \alpha_k]}{\prod_{k=1}^{K} \Gamma[\alpha_k]} \prod_{k=1}^{K} \lambda_k^{\alpha_k - 1} = \text{Dir}_{\lambda_{1\ldots K}}[\alpha_1, \alpha_2, \ldots, \alpha_K]$$

The absolute values determine the concentration: the distribution is highly peaked around the expected value $E[\lambda_k]$ at high parameter values α_k.

✠ The **Univariate Normal** distribution (*Gaussian*) is defined over continuous values $x \in [-\infty, \infty]$. It is defined using 2 parameters mean μ (determines the position of the peak), and variance σ^2 (determines width of the distribution):

$$\Pr(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-0.5(x-\mu)^2/\sigma^2) = \text{Norm}_x[\mu, \sigma^2]$$

By ignoring the fact that pixels can only take discrete values, the normal distribution can be used to describe visual data. Similarly they can be used to describe the *state*.

✠ The **Normal Inverse Gamma** distribution is defined over $\mu \in \mathbb{R}$ and $\sigma^2 \geq 0$, following the rules in the *Univariate Normal* distribution. It is defined using $\alpha, \beta, \gamma > 0$, and $\delta \in \mathbb{R}$ parameters (varying α, β changes the spread of the distribution; γ defines the narrowness; and δ shifts the distribution left or right):

$$\Pr(\mu, \sigma^2) = \frac{\sqrt{\gamma}}{\sigma\sqrt{2\pi}} \frac{\beta^\alpha}{\Gamma(\alpha)} \Big(\frac{1}{\sigma^2}\Big)^{\alpha+1} \exp\Big[-\frac{2\beta + \gamma(\delta-\mu)^2}{2\sigma^2}\Big] = \text{NorSIG}_{\mu,\sigma^2}[\alpha, \beta, \gamma, \delta]$$

✠ The **Multivariate Normal** distribution (*Gaussian*) is defined over a vector of K values $\mathbf{x}$ where each $x \in [-\infty, \infty]$ (*Univariate Normal* is a special case of this distribution). It is defined using 2 parameters: mean $\boldsymbol{\mu}$ is $K \times 1$ vector (determines the position of the peak in $\mathbb{R}^K$ space), and variance $\boldsymbol{\Sigma}$ is a $K \times K$ symmetric positive definite matrix (determines width against each variable):

$$\Pr(\mathbf{x}) = \frac{1}{(2\pi)^{K/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\Big[-0.5(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\Big] = \text{Norm}_{\mathbf{x}}[\boldsymbol{\mu}, \boldsymbol{\Sigma}]$$

This distribution can be used to describe a joint distribution, of let us say, intensities of K pixels within a certain region of an image. Important properties of this distribution:

1. When $K = 2$, $\boldsymbol{\Sigma}$ can take the following forms:

$$\underbrace{\boldsymbol{\Sigma}_{\text{spher}} = \begin{bmatrix}\sigma^2 & 0\\ 0 & \sigma^2\end{bmatrix} \qquad \boldsymbol{\Sigma}_{\text{diag}} = \begin{bmatrix}\sigma_1^2 & 0\\ 0 & \sigma_2^2\end{bmatrix}}_{\text{here the two variables are } \textbf{independent}} \qquad \boldsymbol{\Sigma}_{\text{full}} = \begin{bmatrix}\sigma_1^2 & \sigma_{12}^2\\ \sigma_{12}^2 & \sigma_2^2\end{bmatrix}$$

   $\boldsymbol{\Sigma}_{\text{spher}}$ produces hyper-spherical (circular) distributions; $\boldsymbol{\Sigma}_{\text{diag}}$ gives hyper-ellipsoid (elliptical); and $\boldsymbol{\Sigma}_{\text{full}}$ gives rotated hyper-ellipsoid.

2. For the $\boldsymbol{\Sigma}_{\text{full}}$, we can decompose the matrix by considering that they are simple rotations of $\boldsymbol{\Sigma}_{\text{diag}}$. By taking the transformation $\mathbf{x}' = \mathbf{R}\mathbf{x}$, we deduce: $\boldsymbol{\Sigma}_{\text{full}} = \mathbf{R}\boldsymbol{\Sigma}'_{\text{diag}}\mathbf{R}^{\mathrm{T}}$.

This can also be thought as the eigenvalue decomposition of the matrix $\boldsymbol{\Sigma}_{\text{full}}$ ($\boldsymbol{\Sigma}_{\text{full}} = \mathbf{U}\boldsymbol{\Sigma}'_{\text{diag}}\mathbf{U}^{\text{T}}$). The matrix $\mathbf{U}$ contains unit-length, orthogonal eigenvectors, and the matrix $\boldsymbol{\Sigma}'_{\text{diag}}$ is a diagonal matrix containing the associated eigenvalues. The rotation $\mathbf{U}$ just allows us to make the variables *independent*. This decomposition of the covariance matrix is known as **Principal Component Analysis**, where we only keep a k largest diagonal values from $\boldsymbol{\Sigma}'_{\text{diag}}$.

3. The multivariate normal distribution is preserved under linear transformations $\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{b}$:

$$\Pr(\mathbf{x}) = \text{Norm}_{\mathbf{x}}[\boldsymbol{\mu},\ \boldsymbol{\Sigma}] \quad \rightarrow \quad \Pr(\mathbf{y}) = \text{Norm}_{\mathbf{y}}[\mathbf{A}\boldsymbol{\mu} + \mathbf{b},\ \mathbf{A}\boldsymbol{\Sigma}\mathbf{A}^{\text{T}}]$$

4. If we marginalize over any subset of random variables in a multivariate normal distribution, the remaining distribution is also normally distributed. To find the mean and covariance of the marginal distribution of any subset of variables, one can simply extract the relevant entries from $\boldsymbol{\mu}, \boldsymbol{\Sigma}$.

5. The conditional distribution is also distributed as a multivariate normal; it is computed as:

$$\Pr(\mathbf{x}_1|\mathbf{x}_2) = \text{Norm}_{\mathbf{x}_1}[\boldsymbol{\mu}_1 + \boldsymbol{\Sigma}_{12}^{\text{T}}\boldsymbol{\Sigma}_{22}^{-1}(\mathbf{x}_2 - \boldsymbol{\mu}_2), \boldsymbol{\Sigma}_{11} - \boldsymbol{\Sigma}_{12}^{\text{T}}\boldsymbol{\Sigma}_{22}^{-1}\boldsymbol{\Sigma}_{12}]$$

Remember $\boldsymbol{\Sigma}_{\text{spher}}, \boldsymbol{\Sigma}_{\text{diag}}$ are for independent variables, hence their conditional distribution doesn't change.

✠ The **Normal Inverse Wishart** distribution is defined over $\boldsymbol{\mu}, \boldsymbol{\Sigma}$ and $\sigma^2 \geq 0$, following the rules in the *Multivariate Normal* distribution. It is defined using $\alpha > 0$ (controls the dispersion of covariances), $\boldsymbol{\Psi}$ is a $K \times K$ positive-definite matrix (controls the average covariance), $\gamma > 0$ (controls the dispersion of mean vectors), and $\boldsymbol{\delta}$ is a $K \times 1$ vector (controls the average values of the mean vectors):

$$\begin{aligned}\Pr(\boldsymbol{\mu}, \boldsymbol{\Sigma}) &= \frac{\boldsymbol{\Psi}^{\alpha/2}|\boldsymbol{\Sigma}|^{(\alpha+K+2)/2}\exp\big[-0.5\big(2\text{Tr}(\boldsymbol{\Psi}\boldsymbol{\Sigma}^{-1}) - \gamma(\boldsymbol{\mu}-\boldsymbol{\delta})^{\text{T}}\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}-\boldsymbol{\delta})\big)\big]}{2^{\alpha d/2}2\pi^{d/2}\Gamma(\alpha/2)} \\ &= \text{NorIWis}_{\boldsymbol{\mu},\boldsymbol{\Sigma}}[\alpha, \boldsymbol{\Psi}, \gamma, \boldsymbol{\delta}]\end{aligned}$$

this gives positive values for any valid $\boldsymbol{\mu}, \boldsymbol{\Sigma}$, such that if we integrate over all possible values of $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ it will give 1.

✠ The table 4 gives related distributions. The distribution on the rightmost column is the conjugate of the respective distribution on the left e.g. *Beta* is the conjugate distribution to *Bernoulli*.

The product of the distribution with its conjugate comes out in the form of the conjugate.

✠ Lets say we are given to learn the parameters for the *generative joint* model of inference:

$$\Pr(\theta|\mathbf{X}, \mathbf{Y}) = \frac{\Pr(\mathbf{Y}, \mathbf{X}|\theta)\Pr(\theta)}{\Pr(\mathbf{X}, \mathbf{Y})}$$

Now if the prior $\Pr(\theta)$ is a conjugate of the likelihood term $\Pr(\mathbf{Y}, \mathbf{X}|\theta)$, then the posterior $\Pr(\theta|\mathbf{X}, \mathbf{Y})$ will also have the conjugate form. The multiplicative constant (that comes out of multiplying a function with its conjugate) will get canceled with the evidence term $\Pr(\mathbf{X}, \mathbf{Y})$. This whole process will make it easier to get a closed form expression for the posterior.

Similarly, when we marginalize the result above over all the parameters we have:

$$\Pr(\mathbf{y}^*, \mathbf{x}^*|\mathbf{X}, \mathbf{Y}) = \int \Pr(\mathbf{y}^*, \mathbf{x}^*|\theta) \Pr(\theta|\mathbf{X}, \mathbf{Y}) d\theta$$

Here too, if the learned distribution over the parameters $\Pr(\theta|\mathbf{X}, \mathbf{Y})$ has a form that is conjugate to the term $\Pr(\mathbf{y}^*, \mathbf{x}^*|\theta)$; then the solution to the integral will simply be the constant term (which results from the multiplication of a function with its conjugate) κ. This is known as **exact inference**.

Chapter 5

Inference at a Single Pixel

✠ Aim is to classify each pixel into one of the K classes. Examples:

1. **Chroma-keying**: Used in compositing - assign binary label to indicate if pixel belongs to the bright-colored background or the foreground object.
2. **Background subtraction**: Assign pixels as being background or foreground (and its shadows).
3. **Skin segmentation**
4. **Scene understanding**: Assign pixels to one of the K classes each for a certain object.

Skin pixel data can be best viewed with this transformation $x = 0.5x_r - 0.4187x_g - 0.0813x_b$.

✠ For example, in skin segmentation we are looking for a binary classification from the above mentioned transformation, i.e. $f : x \in [0, 1] \rightarrow y^* \in \{0, 1\}$. If we discretize the input pixel data, we can have something like $f : x \in \{1, \ldots, K\} \rightarrow y^* \in \{0, 1\}$, where K are the number of discretization bins.

✠ In both of the above cases, choosing a classification model has far-reaching implications. Some choices will allow us to do *exact inference* (marginalizing over all parameters) using the *Bayesian approach*. In other cases we have to settle for *MAP*

✠ One advantage of using the *Generative Likelihood approach* over *Discriminative approach* is that adding new classes (world state) to the former is extremely simple (since we model each class separately anyways). Plus when adding new classes, we don't need to relearn the likelihood of the previous ones. Contrary to the generative approach, it is quite tedious to add new classes when using a discriminative approach.

✠ For generative approach we model the class-conditional data for each class (world state) separately. Problems occur when the input data is in large dimensional space / not all of the density function might be relevant in making the distinction between the classes. In contrast, disriminative models actually try

to refine the boundary between the classes and devote more time where the discrimination is a hard problem.

5.1 Generative Likelihood Approach

✠ Supposing the prior $\Pr(\mathbf{y}^*)$ is known.

1. **Model Specification** - Choose a mathematical model for the likelihood $\Pr(\mathbf{x}|\mathbf{y} = m, \boldsymbol{\theta}_m)$ for each world state m. Usually the same distribution model is chosen for all states. Also, distributions are chosen for parameters associated with each world state, $\Pr(\boldsymbol{\theta}_m)$.
2. **Learn the Model** - This stage learns over the parameters $\boldsymbol{\theta}_m$, for each world state of $\mathbf{y} = m$. The result of this step would be a tighter estimate of the likelihood, either by taking a weighted sum over all the parameters (*Bayesian approach*) or by simply choosing the likelihood from the best set of parameters (*MAP approach*):

   (a) *Bayesian Approach* - Using *Bayes' Rule* we learn a distribution $\Pr(\boldsymbol{\theta}_m|\mathbf{X}, \mathbf{Y} = m)$. This is done by taking a subset of the training data $\mathcal{S}$ where we know the state is m:

   $$\Pr(\boldsymbol{\theta}_m|\mathbf{X}, \mathbf{Y} = m) = \frac{\prod_{i\in\mathcal{S}} \Pr(\mathbf{x}_i|\boldsymbol{\theta}_m) \Pr(\boldsymbol{\theta}_m)}{\prod_{i\in\mathcal{S}} \Pr(\mathbf{x}_i)}$$

   The likelihood for this state ($\mathbf{y} = m$) is found by marginalizing over all $\boldsymbol{\theta}_m$:

   $$\Pr(\mathbf{x}^*|\mathbf{y}^* = m) = \int \Pr(\mathbf{x}^*|\mathbf{y}^* = m, \boldsymbol{\theta}_m) \Pr(\boldsymbol{\theta}_m|\mathbf{X}, \mathbf{Y} = m) d\boldsymbol{\theta}_m$$

   (b) *MAP Approach* - For each world state, we find the values for the parameters $\boldsymbol{\theta}_m$ that maximize the posterior:

   $$\hat{\boldsymbol{\theta}}_m = \arg\max_{\boldsymbol{\theta}_m} \Pr(\boldsymbol{\theta}_m|\mathbf{X}, \mathbf{Y} = m)$$

   Giving us the likelihood for state m as $\Pr(\mathbf{x}^*|\mathbf{y}^* = m, \boldsymbol{\theta}_m)$
3. **Inference** - Calculate the posterior for each world state m using *Bayes' Rule*:

   $$\Pr(\mathbf{y}^* = m|\mathbf{x}^*) = \frac{\Pr(\mathbf{x}^*|\mathbf{y}^* = m) \Pr(\mathbf{y}^* = m)}{\Pr(\mathbf{x}^*)}$$

   Note that the denominator is simply the sum of the products of the likelihood and the prior for all the world states:

   $$\Pr(\mathbf{x}^*) = \sum_{m=1}^{M} \Pr(\mathbf{x}^*|\mathbf{y}^* = m) \Pr(\mathbf{y}^* = m)$$

5.1.1 Discrete data

✠ Taking the skin classification example, model the likelihoods $\Pr(x|y=0)$ (non-skin) and $\Pr(x|y=1)$ (skin) with binning $x \in \{1 \dots 20\}$). We then indirectly infer the posterior $\Pr(y^*|x^*)$ over any given colour value x^*. The prior probability is $\Pr(y=1) = 0.2$

1. **Model Specification** - Use two *categorical distributions* to model the skin and non-skin likelihoods. The two respective parameters $\boldsymbol{\lambda}_m$ are modelled each with a *Dirichlet distribution* (the parameters for the dirichlet are set to $\alpha_k = 1$ for all k to give a uniform prior to the parameters).
2. **Learn the Model** - Do for both skin and non-skin data ($m \in \{0, 1\}$):
   (a) *Bayesian Approach* - Using *Bayes' Rule* we get:
   $$\Pr(\boldsymbol{\lambda}_m|\mathbf{X}, \mathbf{Y} = m) = \text{Dir}_{\boldsymbol{\lambda}_m}[N_1 + \alpha_1, \dots, N_{20} + \alpha_{20}]$$
   where N_k is the number of trials where we observed the k'th quantized value. The likelihood for both states ($m \in \{0, 1\}$) come out in this form:
   $$\Pr(\mathbf{x}^* = k|\mathbf{y}^* = m) = \frac{N_k + \alpha_k}{\sum_{j=1}^{20}(N_j + \alpha_j)}$$
   (b) *MAP Approach* - The following parameters maximize the posterior:
   $$\Pr(\mathbf{x}^* = k|\mathbf{y}^* = m) = \hat{\boldsymbol{\lambda}}_{k,m} = \frac{N_k + \alpha_k - 1}{\sum_{j=1}^{K}(N_j + \alpha_j) - K}$$
   where $K = 20$.
3. **Inference** - Compute the posterior as prescribed in 5.1.

5.1.2 Continuous data

✠ Taking the skin classification example, model the likelihoods $\Pr(x|y=0)$ and $\Pr(x|y=1)$ supposing x is now continuous). We then indirectly infer the posterior $\Pr(y^*|x^*)$ over any given colour value x^*.

1. **Model Specification** - Use two *univariate normal distributions* to model the skin and non-skin likelihoods. The two respective parameters are modeled each with a *normal inverse gamma distribution* (the parameters for the NorSIG are set to $\alpha = \beta = \gamma = 1$ and $\delta = 0$ allowing for wide variety of means and variances).
2. **Learn the Model** - Do for both skin and non-skin data ($m \in \{0, 1\}$):
   (a) *Bayesian Approach* - Using *Bayes' Rule* we get:
   $$\Pr(\boldsymbol{\mu}_m, \boldsymbol{\sigma}_m^2|\mathbf{X}, \mathbf{Y} = m) = \text{NorSIG}_{\boldsymbol{\mu}_m, \boldsymbol{\sigma}_m^2}[\tilde{\alpha}, \tilde{\beta}, \tilde{\gamma}, \tilde{\delta}]$$

where $\tilde{\alpha}, \tilde{\beta}, \tilde{\gamma}, \tilde{\delta}$ are decided by the conjugate relationship between Norm_x and NorSIG. Marginalizing over both $\boldsymbol{\mu}_m, \boldsymbol{\sigma}_m^2$, the likelihood for both states ($m \in \{0, 1\}$) comes out in this form:

$$\Pr(\mathbf{x}^* = k|\mathbf{y}^* = m) = \frac{1}{\sqrt{2\pi}} \frac{\sqrt{\tilde{\gamma}}\tilde{\beta}^{\tilde{\alpha}}\Gamma[\breve{\alpha}]}{\sqrt{\breve{\gamma}}\breve{\beta}^{\breve{\alpha}}\Gamma[\tilde{\alpha}]}$$

where $\breve{\alpha}, \breve{\beta}, \breve{\gamma}, \breve{\delta}$ are again decided by the conjugate relationship.

(b) *MAP Approach* - The following parameters maximize the posterior:

$$\hat{\mu}_m = \tilde{\delta}, \qquad \hat{\sigma}_m^2 = \frac{\tilde{\beta}}{\tilde{\alpha} + 3/2}$$

3. **Inference** - Compute the posterior as prescribed in 5.1.

✠ In case our input data is not a scalar value (data in $\mathbb{R}$), rather 3 RGB values (data in $\mathbb{R}^3$), we can model the likelihood by: (1) if we discretize into 20 bins (a) as a product of the categorical distributions for each colour channel (supposes independence in the 3 channels), (b) model a single categorical distribution with 20^3 parameters (needs too much data for training); (2) if we consider data is continuous, model the likelihood by: (1) product of 3 univariate normals (supposes independence); (3) by using a multivariate normal distribution (which would have 6 parameters for covariance and 3 for mean).

5.2 Discriminative Approach

✠ Need to model the posterior directly $\Pr(\mathbf{y}^*|\mathbf{x}^* = k)$ i.e. to give a distribution of the world state for every possible k.

1. **Define Posterior distribution** - Choose a model for the world state $\mathbf{y}^*$ with *distribution parameters* Θ, $\Pr(\mathbf{y}^*|\Theta)$ (for example, if the world state is continuous $\mathbb{R}$ we can use a univariate normal distribution with $\Theta = \{\mu, \sigma^2\}$).
2. **Define Posterior function** - To make the posterior function, we first need to define the *distribution parameters* as a function of the input data $\mathbf{x}^*$, $\Theta(\mathbf{x}^*, \boldsymbol{\theta})$. The *function parameters* $\boldsymbol{\theta}$ modulate the effect of $\mathbf{x}^*$ on Θ. So the posterior function can be written either as $\Pr(\mathbf{y}^*|\Theta(\mathbf{x}^*, \boldsymbol{\theta}))$ or $\Pr(\mathbf{y}^*|\mathbf{x}^*, \boldsymbol{\theta})$. Also we define a prior distribution for each function parameter $\Pr(\boldsymbol{\theta})$.
3. **Learn Posterior function parameters $\boldsymbol{\theta}$** - Here we find a probability distribution over each possible *function parameter* (and hence possible functions):

   (a) *Bayesian Approach* - Using *Bayes' Rule* we learn a distribution $\Pr(\boldsymbol{\theta}|\mathbf{X}, \mathbf{Y})$ (assuming I independent training examples):

$$\Pr(\boldsymbol{\theta}|\mathbf{X}, \mathbf{Y}) = \frac{\prod_{i=1}^{I} \Pr(\mathbf{y}_i|\mathbf{x}_i, \boldsymbol{\theta}) \Pr(\boldsymbol{\theta})}{\Pr(\mathbf{Y})}$$

(b) *MAP Approach* - We simply seek the most probable set of parameters:

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \Pr(\boldsymbol{\theta}|\mathbf{X}, \mathbf{Y})$$

4. **Inference** - In this step we finally get the posterior function (*you need to follow the same approach you followed in the previous step*):

   (a) *Bayesian Approach* - Marginalize over all parameters:

$$\Pr(\mathbf{y}^*|\mathbf{x}^*) = \int \Pr(\mathbf{y}^*|\mathbf{x}^*, \boldsymbol{\theta}) \Pr(\boldsymbol{\theta}|\mathbf{X}, \mathbf{Y}) d\boldsymbol{\theta}$$

   (b) *MAP Approach* - Use the *MAP* parameters $\hat{\boldsymbol{\theta}}$ to get the posterior function, $\Pr(\mathbf{y}^*|\mathbf{x}^*, \hat{\boldsymbol{\theta}})$.

5.2.1 Discrete data

✠ Taking the skin classification example, directly model the posterior $\Pr(y^* = m|x^*)$ (i.e. two posterior distributions) having seen training data with K possible values (binning). Eventually we get a Bernoulli distribution $\Pr(y^*|x^* = k)$ which gives the distribution for each input state $k \in \{1, \ldots, K\}$.

1. **Define Posterior distribution** - Use Bernouilli distribution because the world state is binary:

$$\Pr(y^*|\lambda) = \text{Bern}_{y^*}[\lambda]$$

2. **Define Posterior function** - Since the data values have K options, there will be K parameters λ_k, each giving a different bernoulli distribution:

$$\Pr(y^*|x^*, \lambda_{1,\ldots,K}) = \text{Bern}_{y^*}\left[\prod_{k=1}^{K} \lambda_k^{x_k^*}\right]$$

   Each of these λ_k *function parameters* will be associated with a prior distribution:

$$\Pr(\lambda_k) = \text{Beta}_{\lambda_k}[\alpha, \beta]$$

   Note that we use the same α and β for all prior distributions.

3. **Learn Posterior function parameters $\boldsymbol{\theta}$** - Using the training data $\mathbf{X}, \mathbf{Y}$, we learn the parameters $\lambda_{1,\ldots,K}$:

   (a) *Bayesian Approach* - Using only a subset of the training data $\mathcal{S}_k$ where we know that the data belonged to the k'th bin:

$$\begin{aligned}\Pr(\lambda_k|\mathbf{X}, \mathbf{Y}) &= \frac{\prod_{i \in \mathcal{S}_k} \Pr(\mathbf{y}_i|\mathbf{x}_i, \lambda_k) \Pr(\lambda_k)}{\Pr(\mathbf{Y})} \\ &= \text{Beta}_{\lambda_k}[\tilde{\alpha}_k, \tilde{\beta}_k]\end{aligned}$$

   where $\tilde{\alpha}_k = \alpha + N_k$, $\tilde{\beta}_k = \beta + M_k$, where the N_k is the number of skin training examples when $x = k$, and M_k is the number of non-skin training examples when $x = k$

(b) *MAP Approach* - The most probable parameter for each bin k comes out to be:

$$\hat{\lambda}_k = \arg\max_{\lambda_k} \Pr(\lambda_k|\mathbf{X}, \mathbf{Y}) = \frac{\tilde{\alpha}_k - 1}{\tilde{\alpha}_k + \tilde{\beta}_k - 2}$$

4. **Inference** - In this step we finally get the posterior function:

   (a) *Bayesian Approach* - Marginalizing over all parameters:

$$\begin{aligned}\Pr(\mathbf{y}^*|\mathbf{x}^* = k) &= \int \Pr(y^*|x^* = k, \lambda_k) \Pr(\lambda_k|\mathbf{X}, \mathbf{Y}) d\lambda_k \\ &= \text{Bern}_{y^*}\left[\frac{\tilde{\alpha}_k}{\tilde{\alpha}_k + \tilde{\beta}_k}\right]\end{aligned}$$

   (b) *MAP Approach* - Using the *MAP* parameters $\hat{\lambda}_k$, we get the following posterior:

$$\Pr(\mathbf{y}^*|\mathbf{x}^* = k, \hat{\lambda}_k) = \text{Bern}_{y^*}\left[\frac{\tilde{\alpha}_k - 1}{\tilde{\alpha}_k + \tilde{\beta}_k - 2}\right]$$

5.2.2 Continuous data

✠ Taking the skin classification example, directly model the posterior $\Pr(y^*|x^*)$ with a Bernoulli distribution where the parameters for the distribution changes continuously with the values of x^*. This model is called **logistic regression**:

1. **Define Posterior distribution** - Use Bernoulli distribution because the world state is binary (remember only a single parameter λ is required):

$$\Pr(y^*|\lambda) = \text{Bern}_{y^*}[\lambda]$$

2. **Define Posterior function** - To make the parameter λ dependent on the input data x^*, we use a linear function $a(x^*) = w_0 + w_1 x^*$ (this step is called **Activation**). To constrain the values that a can take from $a \in [-\infty, \infty]$ to $\lambda \in [0, 1]$, we use a logistic sigmoid function:

$$\lambda = \frac{1}{1 + \exp(-a)}$$

   This helps us build a distribution over any world state:

$$\Pr(y^*|x^*, w_0, w_1) = \text{Bern}_{y^*}\left[\frac{1}{1 + \exp(-(w_0 + w_1 x^*))}\right]$$

   Since there is no conjugate prior which directly relates to λ, we use:

$$\Pr(w_0, w_1) = \text{Norm}_{w_0}[0, \sigma_0^2]\text{Norm}_{w_1}[0, \sigma_1^2]$$

3. **Learn Posterior function parameters $\boldsymbol{\theta}$** - Using the training data $\mathbf{X}, \mathbf{Y}$, we learn the parameters w_0, w_1:

   (a) *Bayesian Approach* - The function used to *learn* a distribution $\Pr(w_0, w_1|\mathbf{X}, \mathbf{Y})$ over the parameters, doesn't have a closed form because we didn't have a conjugate prior. This causes problems in using the *Bayesian Approach*.

   (b) *MAP Approach* - The expression for the most probable parameters:

   $$\hat{w}_0, \hat{w}_1 = \arg\max_{w_0, w_1} \Pr(w_0, w_1|\mathbf{X}, \mathbf{Y})$$

   also don't have a closed form. Use gradient descent for this nonlinear optimization: (1) select a sensible direction in the parameter space and move in the direction of the gradient, (2) iteratively improve results.

4. **Inference** - In this step we finally get the posterior function:

   (a) *Bayesian Approach* - The function to marginalize over both parameters doesn't have a closed form.

   (b) *MAP Approach* - Using the *MAP* parameters $\hat{w}_0, \hat{w}_1$, we get the following posterior:

   $$\Pr(y^*|x^*, \hat{w}_0, \hat{w}_1) = \text{Bern}_{y^*}\left[\frac{1}{1+\exp(-(\hat{w}_0 + \hat{w}_1 x^*))}\right]$$

✠ If the world states y^* we are looking are multidimensional (rather than binary in case of skin classification), then model is called **multiclass logistic regression**. The posterior distribution is modeled as a *categorical distribution*. Here we will have one *Activation* for each of the K classes/world states. To constrain the values a_k can take, we use a **softmax function**, which is sort of the multiclass variant of the *logistic sigmoid function*. The problems in *multiclass logistic regression* are that we cannot treat the dimensions as independent, but when we build a joint distribution over them, there can be an explosion of parameters.

✠ *Bayesian approach* is more preferable to *MAP approach* when the amount of training data available is less relative to the number of model parameters - here the Bayesian approach will build a posterior which take this uncertainty into account. When the training dataset is large both methods have little difference. Although, MAP makes over-confident assumptions, because it takes hard estimates on parameters, which in a proper formulation should be treated as uncertain. On the other hand, Bayesian estimation gives us flatter posteriors. Moreover, in case some data value didn't occur in the training set, the posterior probability wouldn't be defined in MAP (*Bayes' Rule* gives zero divided by zero); as compared to Bayesian approach which will assign some small probability even to unseen data.

✠ The **Receiver Operator Characteristic (ROC)** is a graph of true positives against false positives. By thresholding the resulting posterior functions at different probabilities, we can get different set of performance values. This allows us to build a whole curve. The more the curve is toward the top left corner, the more perfect the performance of the algorithm is. The area under the curve $A \in [0, 1]$ is a good metric for judging the performance.

With smaller training sets, *Bayesian approach* is usually superior to *MAP* (has more area under the

curve); and with larger training sets, the performance given by both is similar. The lag of MAP catching up to the *Bayesian approach* is lesser in continuous input data cases.

Chapter 6

Markov Random Fields, MCMC, and Graph Cuts

✠ The aim is to establish relationship between the labels of adjacent regions, rather than looking at them individually which creates noisy results. The right step in this direction is to express the relationship in a neighborhood using a prior $\Pr(\mathbf{l})$ over labellings $\mathbf{l} = \{l_1, \ldots, l_p\}$ for the whole image of p pixels. This can enforce *smoothing*. But the problem is that we are forced to calculate the posterior probability $\Pr(\mathbf{l}|\mathbf{X})$ of all of the labels at once given the image data $\mathbf{X} = \{x_1, \ldots, x_p\}$ (the joint probability of all of the labels is high-dimensional).

6.1 Dense Stereo Matching

✠ *Dense stereo matching* problem for each pixel in the *rectified* image I_x, find the corresponding pixel in I_y. We take the *generative approach*: finding the unknown/*hidden* label l at each image pixel I_x, which represents the value of the *disparity* (offset in the horizontal direction); taking l as a discrete random variable for representing integer pixel values.

✠ To make the problem realistic, we describe the dependency of x on l $\Pr(x|l)$ with Gaussian noise on the offset pixel y's intensity value. Then the likelihood that the observed image data $x_{i,j}$ is described by the disparity $l_{i,j}$ of k pixels:

$$\Pr(x_{i,j}|\mathbf{y}, l_{i,j} = k, \sigma^2) = \mathcal{G}_{x_{i,j}}[y_{i,j+k}, \sigma^2]$$

The prior on the hidden labels/disparity is given by (where K is the largest acceptable disparity):

$$\Pr(l_{i,j} = k) = \begin{cases} \frac{1}{K+1} & \text{if } 0 \leq k \leq K \\ 0 & \text{otherwise} \end{cases}$$

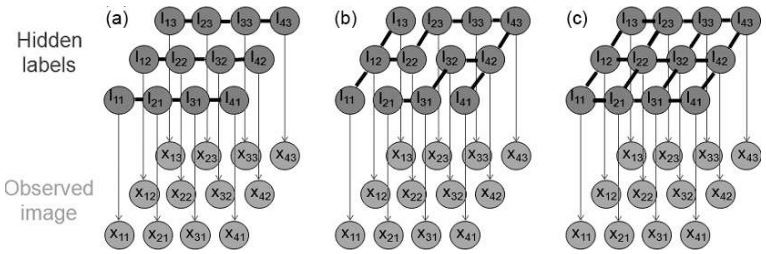

Figure 6.1: *Graphical model* showing relationship between the observed image pixel $x_{i,j}$ and hidden disparity label $l_{i,j}$. The thing to note is the connection between the labels: **(a)** In the *Markov chain* model smoothing occurs across the scan lines and is solved by *dynamic programming*; **(b)** in the *Markov tree* model smoothing occurs across links which form a tree and is solved by *dynamic programming over a tree*; **(c)** in a *Markov Random Field* all label nodes are connected to its neighbors and is solved by *Gibbs sampling* or *Graph cuts*.

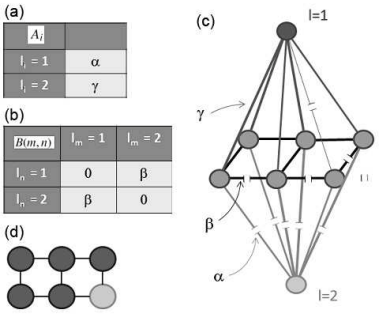

Figure 6.2: Graph setup in **(c)** such that the min-cut is the *MAP* estimation in a binary *MRF* (two labels). **(a)** and **(b)** show *unary* and *binary* costs respectively. The *t-links* to sink takes unary cost α and links to source takes γ. The *n-links* have the cost β for choosing a different label in the neighborhood. **(d)** shows the final segmentation.

We can invert the links of the graphical model (see Figure 6.1) and find the posterior distribution over the label values $l_{i,j} = k$ at pixel (i, j) using *Bayes' Rule*:

$$\Pr(l_{i,j} = k | x_{i,j}, \mathbf{y}, \sigma^2) = \frac{\Pr(x_{i,j} | \mathbf{y}, l_{i,j} = k, \sigma^2) \Pr(l_{i,j} = k)}{\sum_{m=1}^{K} \Pr(x_{i,j} | \mathbf{y}, l_{i,j} = m, \sigma^2) \Pr(l_{i,j} = m)}$$

Essentially we are assuming solution for each pixel is completely independent. Using this we take the *MAP* solution.

✠ We want a *smoothness* constraint over the field of hidden labels, so that neighboring labels tend to similar values. In each developed model, we compute the *MAP* solution:

$$l_i^{(\text{MAP})} = \arg\max_{l_i} \Pr(x_i | \mathbf{y}, \sigma^2, l_i) \Pr(l_i)$$

6.2 Markov chain (Dynamic Programming)

✠ Here we consider each scan-line as a separate problem - and the crucial characteristic that lets us solve this problem is that this graph formulation has no loops. The likelihood term remains the same as before, although the prior changes. We now treat the scan-line as chain of pixels where the prior for

seeing the disparity label $l_{i,j}$ is dependent on the label $l_{i,j-1}$:

$$\Pr(l_{i,j} = k | l_{i,j-1} = m) = \begin{cases} c_1 & \text{if } k = m \\ c_2 & \text{if } |k - m| = 1 \\ c_3 & \text{otherwise} \end{cases}$$

where $c_1 \geq c_2 \geq c_3$. Figure 6.1 (a) shows the graphical model. Now the solution is to use *Bayes' Rule* to establish the posterior over the hidden/unknown labels:

$$\Pr(l_i | x_i, \mathbf{y}, \sigma^2) = \frac{\Pr(x_i | \mathbf{y}, \sigma^2, l_i) \Pr(l_i)}{\Pr(x_i)} = \frac{\left(\prod_{j=1}^{J} \Pr(x_{i,j} | \mathbf{y}, \sigma^2, l_{i,j}) \right) \Pr(l_i)}{\Pr(x_i)}$$

where J are the number of pixels on a scan-line. Note that we can break down the likelihood since all observations are independent. Now what about the prior. We mentioned above that adjacent pixels are related by a cost, hence:

$$\Pr(l_i) = \Pr(l_{i,1}) \times \Pr(l_{i,2} | l_{i,1}) \times \Pr(l_{i,3} | l_{i,2}) \ldots \Pr(l_{i,J} | l_{i,J-1}) = \Pr(l_{i,1}) \prod_{j=2}^{J} \Pr(l_{i,j} | l_{i,j-1})$$

this is exactly what a **Markov chain** is (having the *Markov property*) - i.e. the hidden label of a particular pixel only depends on its immediate neighbor.

✠ We aim to find the *MAP* labeling through a dynamic programming approach:

$$\begin{aligned} l_i^{(\text{MAP})} &= \arg\max_{l_i} \Pr(x_i | \mathbf{y}, \sigma^2, l_i) \Pr(l_i) \\ &= \arg\max_{l_i} \left(\prod_{j=1}^{J} \Pr(x_{i,j} | \mathbf{y}, \sigma^2, l_{i,j}) \right) \left(\prod_{j=2}^{J} \Pr(l_{i,j} | l_{i,j-1}) \right) \Pr(l_{i,1}) \end{aligned}$$

$$l_i^{(\text{MAP})} = \arg\min_{l_i} \underbrace{\sum_{j=1}^{J} A_j}_{\text{node costs}} + \underbrace{\sum_{j=2}^{J} B_{j,j-1}}_{\text{link costs}} + \underbrace{B_1}_{\text{constant}}$$

where in the last equation $A_j = -\log \Pr(x_{i,j} | \mathbf{y}, \sigma^2, l_{i,j})$ and $B_{j,j-1} = -\log \Pr(l_{i,j} | l_{i,j-1})$. This last equation is just the conversion of the problem to a negative log minimization (remember the log is a monotonic transformation and location of extremas don't change).

✠ Now using the **Viterbi algorithm** (*dynamic programming/max-product algorithm*), get the *MAP* solution. The minimum cost path would correspond to the *MAP* disparity for a scan-line. Smoothing across a scan-line can be increased by increasing c_1 as compared to c_2 and c_3. Lowering this c_1 is equivalent to enforcing independent pixel solutions.

6.3 Markov tree (Dynamic Programming on a tree)

✠ It should be noted that these dynamic programming approaches can be used whenever there are no

loops in a graph. Veksler (CVPR '05) proposed starting with a fully connected graph (as given in Figure 6.1 (c)) and pruning links until there are no loops (as given in Figure 6.1 (b)). Pruning is done wherever it is thought that the disparity map shouldn't be smooth anyways - i.e. pixels with large intensity changes are good candidates for such pruning.

6.4 Markov Random Fields

✠ A *Markov Random Field* (*MRF*) defines the prior probability distribution (the $\Pr(l_i)$ term) over a set of hidden random variables $\mathbf{l}$. A *MRF* is determined by three aspects:

1. A set of site $S = \{S_1 \dots S_P\}$ (pixels).
2. A set of random variables $L = \{l_1 \dots l_P\}$ associated with each of the P sites.
3. A set of neighbors $N_{1\dots P}$ for each of the P sites. They define the extent of the probabilistic connections between sites.

✠ For our problem the hidden labels can only take one of the 2 labels (i.e. we will form a binary MRF). If there are P pixels, the total number of possible configurations would be 2^P. The goal is to specify a probability distribution over these 2^P states - we can incorporate our understanding of images into the probability distribution or simply use a training set.

✠ To be a *Markov Random Field*, it should obey the *Markov property*:

$$\Pr(l_i|l_{S\setminus i}) = \Pr(l_i|l_{N_i}) \quad \forall i \in S$$

i.e. only labels in a small distance of each other will have a direct probabilistic relationship. We will define our neighborhood to be the 4 connected pixels to each *site*. This prior will be set such that fields of labels that are smooth would tend to have a higher probability.

✠ The key thought here is the **Hammersley-Clifford theorem**. It states that any system that follows the above mentioned *Markov property* for an *MRF*, can be written as a **Gibbs' distribution**:

$$\Pr(\mathbf{l} = \frac{1}{Z} \exp\left[-\frac{1}{T}U(\mathbf{l})\right]$$

where Z is the partition function/normalization constant (an unknowable quantity), T is the temperature (usually set to $T = 1$), and U is the cost of the MRF. Also note as U increases, the probability decreases.

$U(\mathbf{l})$ is essentially the sum of **clique potentials**, which are, effectively, costs:

$$U(\mathbf{I}) = \sum_{c \in C} V_c(\mathbf{l})$$

As *clique potential* increases, the probability decreases. There is one clique potential for every maximal clique C in the graph. Hence, the hidden variable l_{22} would contribute to the cliques C_{12}^{22}, C_{23}^{22}, C_{32}^{22}, and

C_{21}^{22} (for example the last one denotes a clique between labels $l_{2,1}$ and $l_{2,2}$ - see Figure 6.1). The clique potential takes the form of the 2×2 table given in Figure 6.2 (b), which denotes the cost w_{ij} for site 1 being in state i and site 2 being in state j.

✠ There could be situations where you know that a certain pixel is more likely to adopt a certain label. Let's say, the borders of the image are more likely to be the background. To incorporate such a bias we use the **Unary Term**:

$$\Pr(\mathbf{l}) = \frac{1}{Z} \exp\left[-\frac{1}{T} U(\mathbf{l})\right] \prod_{i=1}^{P} \Pr(l_i)$$

✠ Once more we aim to do *inference*. Using *Bayes' Rule* we have:

$$\Pr(\mathbf{l}|\mathbf{X}) = \frac{\Pr(\mathbf{X}|\mathbf{l}) \Pr(\mathbf{l})}{\Pr(\mathbf{X})}$$

Although $\Pr(\mathbf{l})$ contains the uncomputable Z term, we can still aim to compute the **MAP estimation for the MRF** because Z is constant. According to *Bayes' Rule*, the posterior probability over the states is proportional to the product of the likelihood terms and the prior:

$$\mathbf{l}^{(\text{MAP})} = \arg\max_{\mathbf{l}} \Pr(\mathbf{l}|\mathbf{X}) \;\propto\; \frac{1}{Z} \exp\left[-\frac{1}{T} U(\mathbf{l})\right] \prod_{i=1}^{P} \Pr(x_i|l_i) \Pr(l_i) \tag{6.1}$$

Its important to note that even finding this *MAP* solution is not trivial, since this distribution turns out to be quite complex. *MCMC* methods give an approximation by sampling from the posterior distribution. *Graph Cuts* find the exact MAP solution for the binary labeling problem.

✠ *MRF*s have an interesting application in **image denoising**. We take the same graphical model given in Figure 6.1, but now the observed noisy image is $\mathbf{X} = \{x_1, \ldots, x_p\}$, and we will try to infer the denoised/uncorrupted image $\mathbf{l} = \{l_1, \ldots, l_p\}$. Hence, we develop this posterior probability:

$$\Pr(\mathbf{l}|\mathbf{X}) = \frac{\prod_{i=1}^{P} \Pr(x_i|l_i)\ \Pr(\mathbf{l})}{\Pr(\mathbf{X})}$$

as you can see, we have assumed the values of x_i only depend on the label l_i. If we consider that the noise process flips the pixel polarity with probability ρ we would have:

$$\Pr(x_i|l_i = 0) = \text{Bern}_{x_n}[\rho]$$
$$\Pr(x_i|l_i = 1) = \text{Bern}_{x_n}[1 - \rho]$$

We would define a prior probability over world state to be an *MRF*, with cliques C. The cliques here will be only with the four neighbors:

$$\frac{1}{Z} \exp\left[-U(\mathbf{l})\right] = \frac{1}{Z} \exp\left[-\sum_{(m,n) \in C} \Psi(l_m, l_n, \theta)\right]$$

where $\Psi(l_m, l_n, \theta)$ is the cost $\theta_{j,k}$ for having labels at $l_m = j$ and $l_n = k$. The cost $\theta_{j,k}$ would be set to encourage smooth solutions.

✠ It can be argued that the interactions imposed through the prior are too local. To fix this, **Field of Experts** (Roth and Black, CVPR '05) was proposed. In this the label depends on a 5×5 neighborhood. In the binary MRF formulation there was some prior probability of observing labels $l_i = 1$ and $l_i = 0$. Now there is a prior probability over any configuration of the 25 pixels. The particular probability distribution that was selected was a **Product of Experts** (*mixture models* use weighted sums of simple probability distribution to create a complex distribution, whereas *product of experts* use products of simpler distributions).

✠ The particular choice of the cost of the MRF, Ψ, is of the form $\log f(l_i)$. The term $f(l_i)$ defines the *product of experts*:

$$f(l_i) = \frac{1}{Z_{\text{POE}}} \prod_{e=1}^{E} \Pr(\mathbf{j}_e^{\mathrm{T}} l_i | \theta_e)$$

where $\mathbf{j}_e^{\mathrm{T}} l_i$ is used to project the 25 pixel patch data onto a particular axis. 1D t-distributions are used for these projections $\mathbf{j}_e$ whose parameter θ_e are learned from training data. Now taking $\Pr(x_i|l_i) = \mathcal{G}_{x_i}[l_i, \sigma^2]$, we can find the *MAP* solution: $\arg\max_{\mathbf{l}} \Pr(\mathbf{l}|\mathbf{X})$

6.4.1 Markov Chain Monte Carlo

✠ One naïve way to get the *MAP* solution would be to keep drawing random samples from the posterior distribution, and one with the highest probability will be the approximate *MAP* solution. But note that most of the states should have a low probability as they don't fall in the domain of natural images. Hence this requires some intelligent sampling where we only sample from high probability areas. Such class of algorithms are known as **Markov Chain Monte Carlo** methods (MCMC). The key idea is that the current sampling will only depend on the previous sample (*Markov Chain* part), yet the choice of sampling is not deterministic (*Monte Carlo* part).

Gibbs' Sampling

✠ **Gibbs' Sampling** is an example of a *MCMC* method. To illustrate the idea, consider the distribution $\Pr(\mathbf{l})$. We first choose a random starting location $\mathbf{l}^{[0]}$. We generate the next sample by taking turns on the different dimensions (in the MRF each dimension corresponds to the pixel site). At each step we sample from the conditional probability $\Pr(l_i|l_{1\ldots P\setminus i})$ so that we only change values for a single dimension while keeping others constant. Once we have done this for every dimension we will have $\mathbf{l}^{[1]}$. We can proceed similarly and get $\mathbf{l}^{[n]}$. Note that at each step we are more likely to change the current label to a configuration with overall higher probability (although the stochastic nature will result in some sampling from less likely regions). Similarly this can be done for the posterior distribution:

$$\Pr(l_i = k|l_{1\ldots P\setminus i}, \mathbf{X}) = \frac{\Pr(l_i = k, l_{1\ldots P\setminus i}, \mathbf{X})}{\Pr(l_{1\ldots P\setminus i}, \mathbf{X})} = \frac{\Pr(l_i = k, l_{1\ldots P\setminus i}, \mathbf{X})}{\sum_{a=1}^{2} \Pr(l_i = a, l_{1\ldots P\setminus i}, \mathbf{X})}$$

6.4.2 Graph Cuts

✠ To find the exact *MAP* solution for an *MRF* we use **Graph Cuts** with the following steps:

1. Write the maximization of the posterior probability in terms of energy minimization.
2. Express this minimization in terms of finding the min-cut on a graph.
3. Solve the min-cut by solving an equivalent problem of finding the max-flow from source to sink of the graph.

✠ Remember that we wanted to maximize the posterior probability (see Equation 6.1). Since the logarithm is a monotonic transformation, we can equivalently maximize the logarithm of this equation. Now rather than maximizing this function we can minimize the negative of this term, hence we have:

$$\mathbf{l}^{(\text{MAP})} = \arg\min_{\mathbf{l}} \underbrace{\sum_{i=1}^{P} A_i}_{\text{unary term}} + \underbrace{\sum_{(m,n)\in C} B(l_m, l_n)}_{\text{binary term}}$$

where $A_i = -\log \Pr(x_i|l_i)$ and $B(l_m, l_n)$ is the cost of the neighboring pixels taking their respective label values. The binary term is also called the *pairwise term*. The graph (see Figure 6.2) is set up in a way that the edge-weights on the source t-links is given by the *unary cost* for being label 2 and the edge-weights on the sink t-links is given by the *unary cost* for being label 1.

✠ The min-cut (by *saturating* the flow using the **augmenting paths algorithm**) corresponds to the minimum of the energy function. For each node we should break a t-link to the source or t-link to the sink (or otherwise some path from source to sink would not have been saturated)

6.4.3 Multi-label Problem

✠ Now, given $K > 2$ labels, how can we solve using an *MRF* formulation. Now the *clique potential*/cost table will be a $K \times K$ matrix. It can be shown that the *MAP* solution can only be achieved in this case when cost function is convex. But the problem is that convex function tend blur boundaries. **Pott's model** is an example which has a discontinuity preserving cost function: no cost when labels are the same; constant cost regardless of the level of disparity in labels.

✠ **Alpha Expansion** is an algorithm provides an approximate solution to non-convex cost functions. It breaks down the multi-label problem into a binary *MRF* formulation. On each iteration it randomly selects a label $\alpha \in \{1, \ldots, K\}$. Then it takes all pixels which don't have the label α. Now each pixel is formulated as a binary cut problem, where it considers changing the pixel to label α or keeping its current label. The process is iterated until no choice of α triggers any change in labeling. Each expansion move (the change in label) is guaranteed to lower the objective function - although it is not guaranteed that the final result will be the overall minimum.

Chapter 7

Appendix

✠ Given a vector of multivariate gaussian random variables $\mathbf{z}$:

$$\Pr(\mathbf{z}) = \Pr\left(\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}\right) = \mathcal{G}_{\mathbf{z}}\left(\begin{bmatrix}\mathbf{a}\\ \mathbf{b}\end{bmatrix},\ \begin{bmatrix}\mathbf{A} & \mathbf{C}\\ \mathbf{C}^{\mathrm{T}} & \mathbf{B}\end{bmatrix}\right)$$

Conditional distributions of Gaussian are also Gaussian:

$$\Pr(\mathbf{x}|\mathbf{y}) = \mathcal{G}_{\mathbf{x}}(\mathbf{a} + \mathbf{C}\mathbf{B}^{-1}(\mathbf{y} - \mathbf{b}),\ \mathbf{A} - \mathbf{C}\mathbf{B}^{-1}\mathbf{C}^{\mathrm{T}})$$
$$\Pr(\mathbf{y}|\mathbf{x}) = \mathcal{G}_{\mathbf{y}}(\mathbf{b} + \mathbf{C}^{\mathrm{T}}\mathbf{A}^{-1}(\mathbf{x} - \mathbf{a}),\ \mathbf{B} - \mathbf{C}^{\mathrm{T}}\mathbf{A}^{-1}\mathbf{C})$$

✠ Product of two Gaussians is also a Gaussian:

$$\mathcal{G}_{\mathbf{x}}(\mathbf{a}, \mathbf{A})\mathcal{G}_{\mathbf{x}}(\mathbf{b}, \mathbf{B}) \propto \mathcal{G}_{\mathbf{x}}[(\mathbf{A}^{-1} + \mathbf{B}^{-1})^{-1}(\mathbf{A}^{-1}\mathbf{a} + \mathbf{B}^{-1}\mathbf{b}),\ (\mathbf{A}^{-1} + \mathbf{B}^{-1})^{-1}]$$

✠ Consider a Gaussian in $\mathbf{x}$ with a mean that is a linear function $\mathbf{H}$ of $\mathbf{y}$

$$\mathcal{G}_{\mathbf{x}}(\mathbf{H}\mathbf{y}, \boldsymbol{\Sigma}) \propto \mathcal{G}_{\mathbf{x}}[(\mathbf{H}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\mathbf{H})^{-1}\mathbf{H}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\mathbf{x},\ (\mathbf{H}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\mathbf{H})^{-1}]$$

✠ Any linear function of a Gaussian variable is also distributed as a Gaussian. If $\mathbf{x}$ as $G_x(\mu, \Sigma)$, then the random variable $\mathbf{A}\mathbf{x} + \mathbf{y}$ is also a Gaussian:

$$\Pr(\mathbf{A}\mathbf{x} + \mathbf{y}) = \mathcal{G}_{\mathbf{x}}\left(\mathbf{A}\mu + \mathbf{y},\ \mathbf{A}\Sigma\mathbf{A}^{\mathrm{T}}\right)$$

Setting $\mathbf{A} = \Sigma^{-1/2}$ and the vector $\mathbf{y} = -\Sigma^{-1/2}\mu$, gives us the probability distribution $G_{\mathbf{x}}(0, 1)$. This is known as **whitening transform**.

✠ The convolution of two Gaussians is also a Gaussian:

$$\mathcal{G}_{\mathbf{x}}(\mathbf{a}, \mathbf{A}) * \mathcal{G}_{\mathbf{x}}(\mathbf{b}, \mathbf{B}) = \mathcal{G}_{\mathbf{x}}(\mathbf{a} + \mathbf{b}, \mathbf{A} + \mathbf{B})$$

This happens in practice when we draw from one distribution and from a second distribution and add the results. The distribution of the sum is equal to the convolution of the original distributions.