Epipolar geometry contd.
Estimating F – 8-point algorithm

• The fundamental matrix $F$ is defined by

$$x'^{T}Fx = 0$$

for any pair of matches $x$ and $x'$ in two images.

• Let $x=(u,v,1)^{T}$ and $x'=(u',v',1)^{T}$,

$$F = \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix}$$

each match gives a linear equation

$$uu'f_{11} + vu'f_{12} + u'f_{13} + uv'f_{21} + vv'f_{22} + v'f_{23} + uf_{31} + vf_{32} + f_{33} = 0$$
8-point algorithm

\[
\begin{bmatrix}
  u_1 u_1' & v_1 u_1' & u_1' & u_1 v_1' & v_1' & v_1 & u_1 & v_1 & 1 \\
  u_2 u_2' & v_2 u_2' & u_2' & u_2 v_2' & v_2' & v_2 & u_2 & v_2 & 1 \\
  \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
  u_n u_n' & v_n u_n' & u_n' & u_n v_n' & v_n' & v_n & u_n & v_n & 1 \\
\end{bmatrix}
\begin{bmatrix}
f_{11} \\
f_{12} \\
f_{13} \\
f_{21} \\
f_{22} \\
f_{23} \\
f_{31} \\
f_{32} \\
f_{33}
\end{bmatrix} = 0
\]

- In reality, instead of solving \(A f = 0\), we seek \(f\) to minimize \(\|A f\|\), least eigenvector of \(A^T A\).
8-point algorithm – Problem?

- \( \mathbf{F} \) should have rank 2
- To enforce that \( \mathbf{F} \) is of rank 2, \( \mathbf{F} \) is replaced by \( \mathbf{F}' \) that minimizes \( \| \mathbf{F} - \mathbf{F}' \| \) subject to the rank constraint.

- This is achieved by SVD. Let \( \mathbf{F} = \mathbf{U} \Sigma \mathbf{V}^T \), where

\[
\Sigma = \begin{bmatrix}
\sigma_1 & 0 & 0 \\
0 & \sigma_2 & 0 \\
0 & 0 & \sigma_3
\end{bmatrix}, \quad \text{let} \quad \Sigma' = \begin{bmatrix}
\sigma_1 & 0 & 0 \\
0 & \sigma_2 & 0 \\
0 & 0 & 0
\end{bmatrix}
\]

then \( \mathbf{F}' = \mathbf{U} \Sigma' \mathbf{V}^T \) is the solution.
Recovering camera parameters from F / E

• Can we recover R and t between the cameras from F?

\[ F = K_2^{-T}[t] \times RK_1^{-1} \]

• No: \( K_1 \) and \( K_2 \) are in principle arbitrary matrices

• What if we knew \( K_1 \) and \( K_2 \) to be identity?

\[ E = [t] \times R \]
Recovering camera parameters from $E$

\[ E = [t] \times R \]
\[ t^T E = t^T [t] \times R = 0 \]
\[ E^T t = 0 \]

- $t$ is a solution to $E^T x = 0$
- Can’t distinguish between $t$ and $ct$ for constant scalar $c$
- How do we recover $R$?
Recovering camera parameters from E

\[ E = [t]_\times R \]

- We know \( E \) and \( t \)
- Consider taking SVD of \( E \) and \([t]_\times\)

\[
[t]_\times = U\Sigma V^T
\]

\[
E = U'\Sigma' V'^T
\]

\[
U'\Sigma' V'^T = E = [t]_\times R = U\Sigma V^T R
\]

\[
U'\Sigma' V'^T = U\Sigma V^T R
\]

\[
V'^T = V^T R
\]
Recovering camera parameters from $E$

\[ E = [t] \times R \]
\[ t^T E = t^T [t] \times R = 0 \]
\[ E^T \mathbf{t} = 0 \]

- $\mathbf{t}$ is a solution to $E^T \mathbf{x} = 0$
- Can’t distinguish between $\mathbf{t}$ and $c\mathbf{t}$ for constant scalar $c$
8-point algorithm

- Pros: it is linear, easy to implement and fast
- Cons: susceptible to noise
- Degenerate: if points are on same plane

- Normalized 8-point algorithm: Hartley
  - Position origin at centroid of image points
  - Rescale coordinates so that center to farthest point is $\sqrt{2}$
Other approaches to obtaining 3D structure
Active stereo with structured light

- Project “structured” light patterns onto the object
  - simplifies the correspondence problem
  - Allows us to use only one camera

Active stereo with structured light

Microsoft Kinect

- 3D Depth Sensors
- RGB Camera
- Multi-Array Mic
- Motorized Tilt
Light and geometry
Till now: 3D structure from multiple cameras

• Problems:
  • requires calibrated cameras
  • requires correspondence

• Other cues to 3D structure?

Key Idea: use feature motion to understand shape
What does 3D structure mean?

• We have been talking about the depth of a pixel.
What does 3D structure mean?

- But we can also look at the orientation of the surface at each pixel: *surface normal*

Not enough by itself to reveal absolute locations, but gives enough of a clue to object shape
Shading is a cue to surface orientation
Modeling Image Formation

Now we need to reason about:

- How light interacts with the scene
- How a pixel value is related to light energy in the world

Track a “ray” of light all the way from light source to the sensor
How does light interact with the scene?

• Light is a bunch of photons
• Photons are energy packets
• Light starts from the light source, is reflected / absorbed by surfaces and lands on the camera

• Two key quantities:
  • Irradiance
  • Radiance
Radiance

• How do we measure the “strength” of a beam of light?
• Idea: put a sensor and see how much energy it gets

If the sensor is slanted it gets less energy

A larger sensor captures more energy
Radiance

• How do we measure the “strength” of a beam of light?

• Radiance: power in a particular direction per unit area when surface is orthogonal to direction.

If the sensor is slanted it gets less energy

A larger sensor captures more energy
Radiance

• Pixels measure radiance
Where do the rays come from?

• Rays from the light source “reflect” off a surface and reach camera

• Reflection: Surface absorbs light energy and radiates it back
Irradiance

- Radiance measures the energy of a light beam.
- But what is the energy received by a surface?
- Depends on the area of the surface and the orientation.

\[
A \cdot \cos \theta
\]
Irradiance

• Power received by a surface patch
  • of area $A$
  • from a beam of radiance $L$
  • coming at angle $\theta = L A \cos \theta$
**Irradiance**

- Power received by a surface patch of unit area
  - from a beam of radiance $L$
  - coming at angle $\theta = L\cos\theta$
- Called Irradiance
- Irradiance = Radiance of ray $\times \cos\theta$
Light rays interacting with a surface

• Light of radiance $L_i$ comes from light source at an incoming direction $\theta_i$

• It sends out a ray of radiance $L_r$ in the outgoing direction $\theta_r$

• How does $L_r$ relate to $L_i$?

- $\mathbf{N}$ is surface normal
- $\mathbf{L}$ is direction of light, making $\theta_i$ with normal
- $\mathbf{V}$ is viewing direction, making $\theta_r$ with normal
Light rays interacting with a surface

- \( \mathbf{N} \) is surface normal
- \( \mathbf{L} \) is direction of light, making \( \theta_i \) with normal
- \( \mathbf{V} \) is viewing direction, making \( \theta_r \) with normal

Output radiance along \( \mathbf{V} \)

\[
L_r = \rho(\theta_i, \theta_r) L_i \cos \theta_i
\]

Incoming irradiance along \( \mathbf{L} \)

Bi-directional reflectance function (BRDF)
Light rays interacting with a surface

\[ L_r = \rho(\theta_i, \theta_r)L_i \cos \theta_i \]

- Special case 1: Perfect mirror
  - \( \rho(\theta_i, \theta_r) = 0 \) unless \( \theta_i = \theta_r \)
- Special case 2: Matte surface
  - \( \rho(\theta_i, \theta_r) = \rho_0 \) (constant)
Special case 1: Perfect mirror

• \( \rho(\theta_i, \theta_r) = 0 \) unless \( \theta_i = \theta_r \)
• Also called “Specular surfaces”
• Reflects light in a single, particular direction
Special case 2: Matte surface

- $\rho(\theta_i, \theta_r) = \rho_0$
- Also called “Lambertian surfaces”
- Reflected light is independent of viewing direction
Lambertian surfaces

• For a lambertian surface:

\[ L_r = \rho L_i \cos \theta_i \]

\[ \Rightarrow L_r = \rho L_i \mathbf{L} \cdot \mathbf{N} \]

• \( \rho \) is called albedo
  • Think of this as paint
  • High albedo: white colored surface
  • Low albedo: black surface
  • Varies from point to point
Lambertian surfaces

• Assume the light is directional: all rays from light source are parallel.
  • Equivalent to a light source infinitely far away

• All pixels get light from the same direction \( \mathbf{L} \) and of the same intensity \( L_i \)
Lambertian surfaces

\[ I(x, y) = \rho(x, y) L_i L \cdot N(x, y) \]

Reflectance image

Shading image

Intrinsic Image Decomposition
Lambertian surfaces
Lambertian surfaces

Far
Near

$Z$  
shape / depth

$R$
Reflectance

$I = R \odot S(Z, L)$  
Lambertian reflectance

$S(Z, L)$  
Shading image of $Z$ and $L$

$L$
Illumination
Reconstructing Lambertian surfaces

\[ I(x, y) = \rho(x, y) L_i L \cdot N(x, y) \]

• Equation is a constraint on albedo and normals
• Can we solve for albedo and normals?
Solution 1: Shape from Shading

\[ I(x, y) = \rho(x, y)L_i \mathbf{L} \cdot \mathbf{N}(x, y) \]

- Assume \( L_i \) is 1
- Assume \( \mathbf{L} \) is known
- Assume some normals known
- Assume surface smooth: normals change slowly

In practice, SFS doesn’t work very well: assumptions are too restrictive, too much ambiguity in nontrivial scenes.