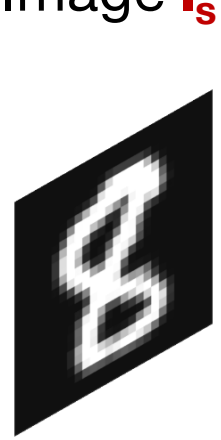
Machine Learning in Imaging

BME 590L
Roarke Horstmeyer

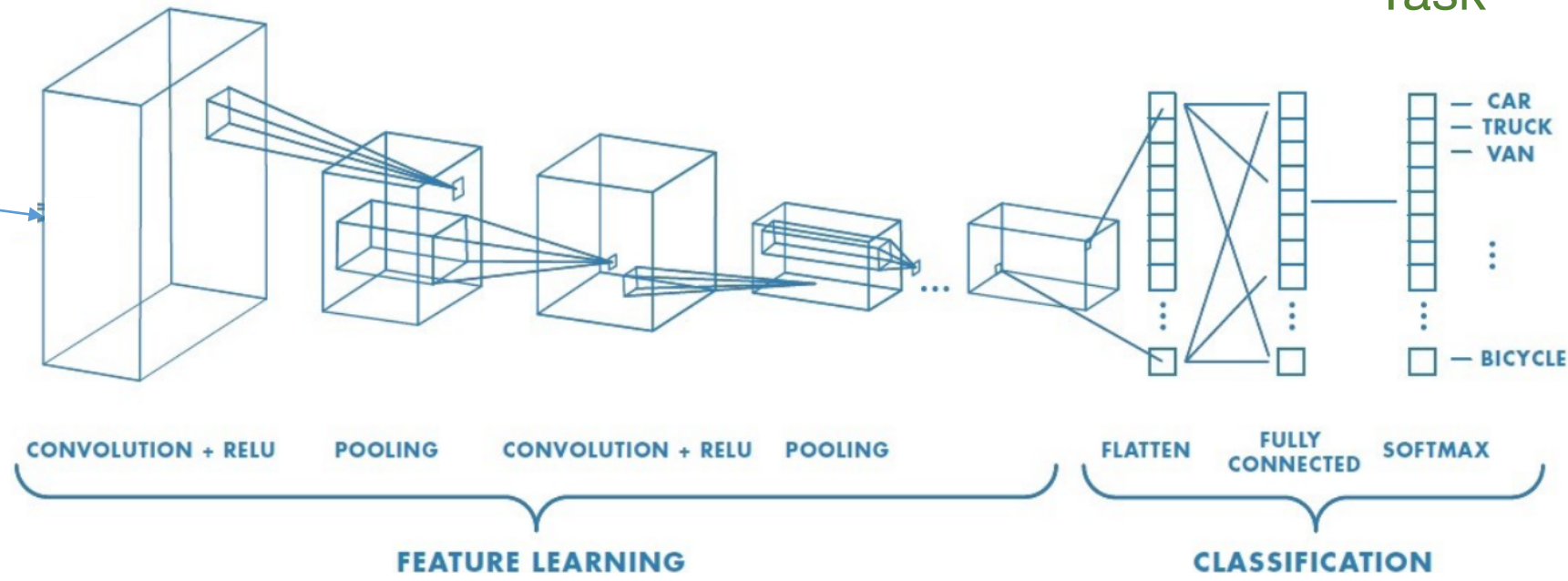
Lecture 16: Physical CNN examples & introduction to optics

Bringing together physical and digital image representations

Image $\mathbf{I_s}$

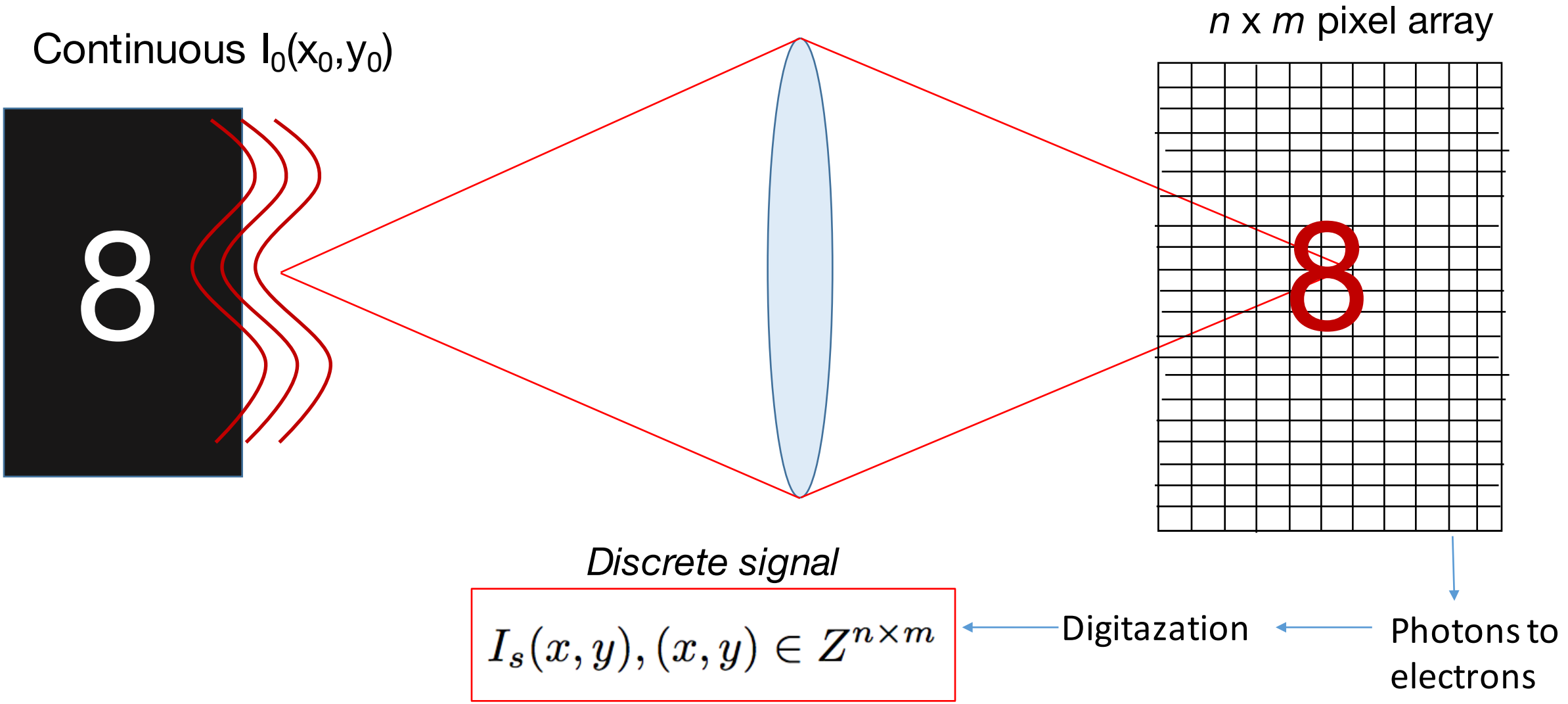


Task



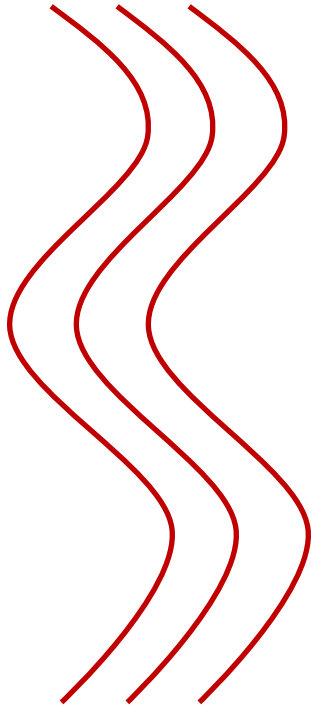
Task = $\mathbf{W_n}$ …ReLU[$\mathbf{W_1}$ ReLU[$\mathbf{W_0}$ $\mathbf{I_s}$]…]

Simple model of image formation



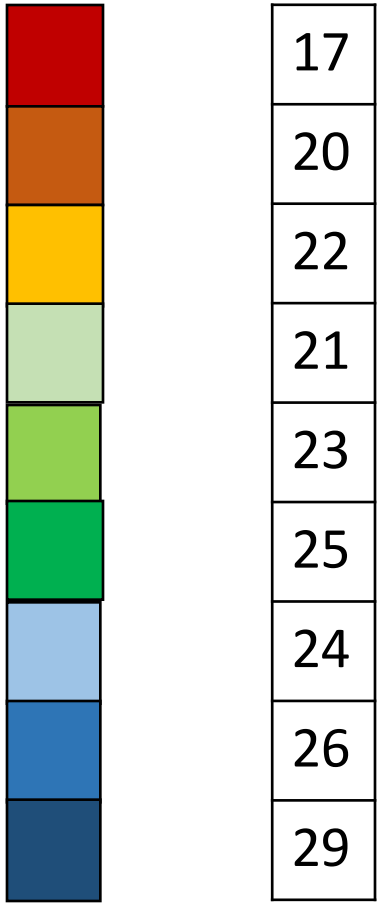
What does the Sampling Theorem mean for us?

Continuous functions



*conditions

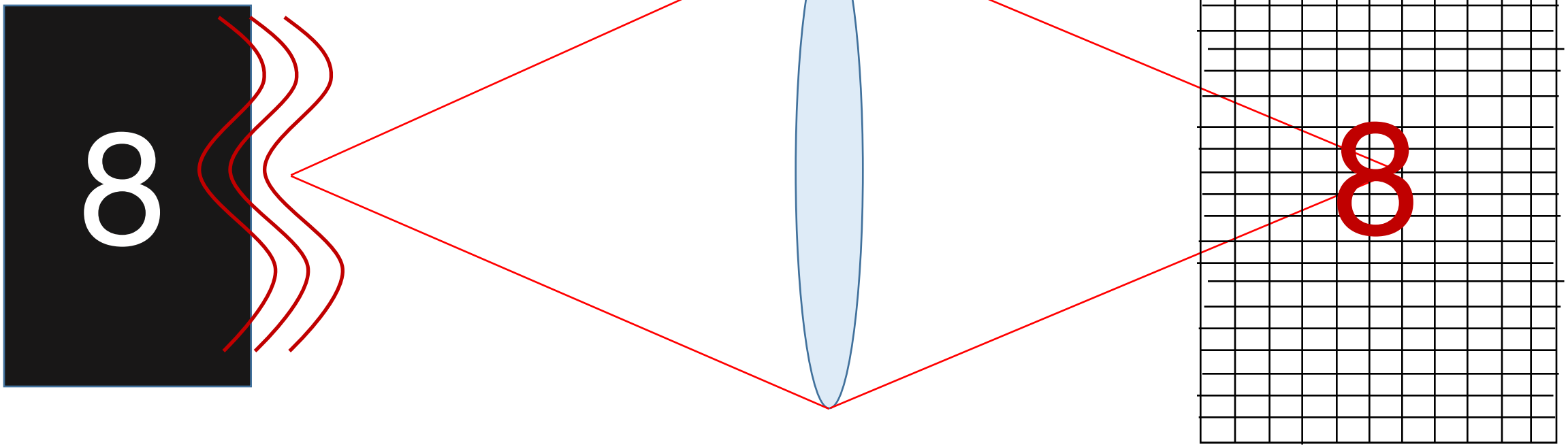
Discretize vectors
(and matrices)



17
20
22
21
23
25
24
26
29

Simple model of image formation

Discrete $I_0(x_0,y_0)$



Discrete signal

$$I_s(x,y),(x,y) \in Z^{n \times m}$$

← Digitazation ← Photons to electrons

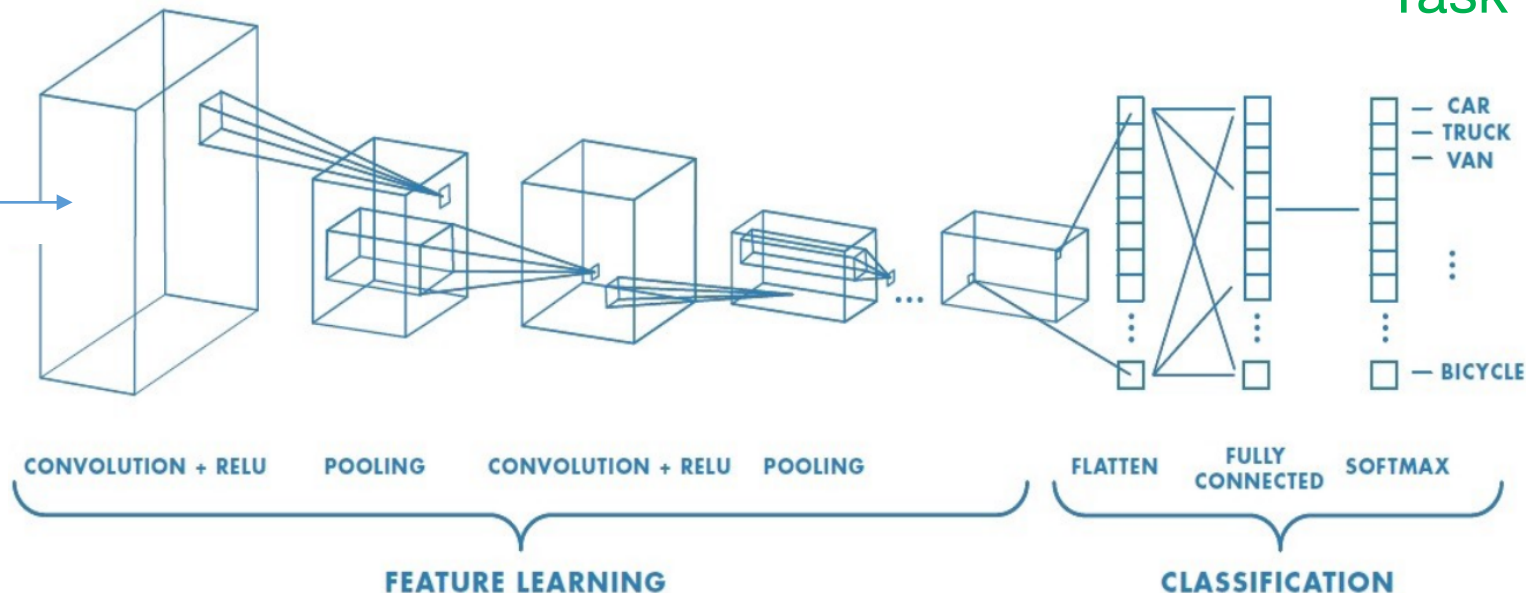
Bringing together physical and digital image representations

Physical Layers

Digital Layers

Task

Physical Object I_0 → f[ ] → Digitized image I_s →



$$I_s = f[I_0]$$

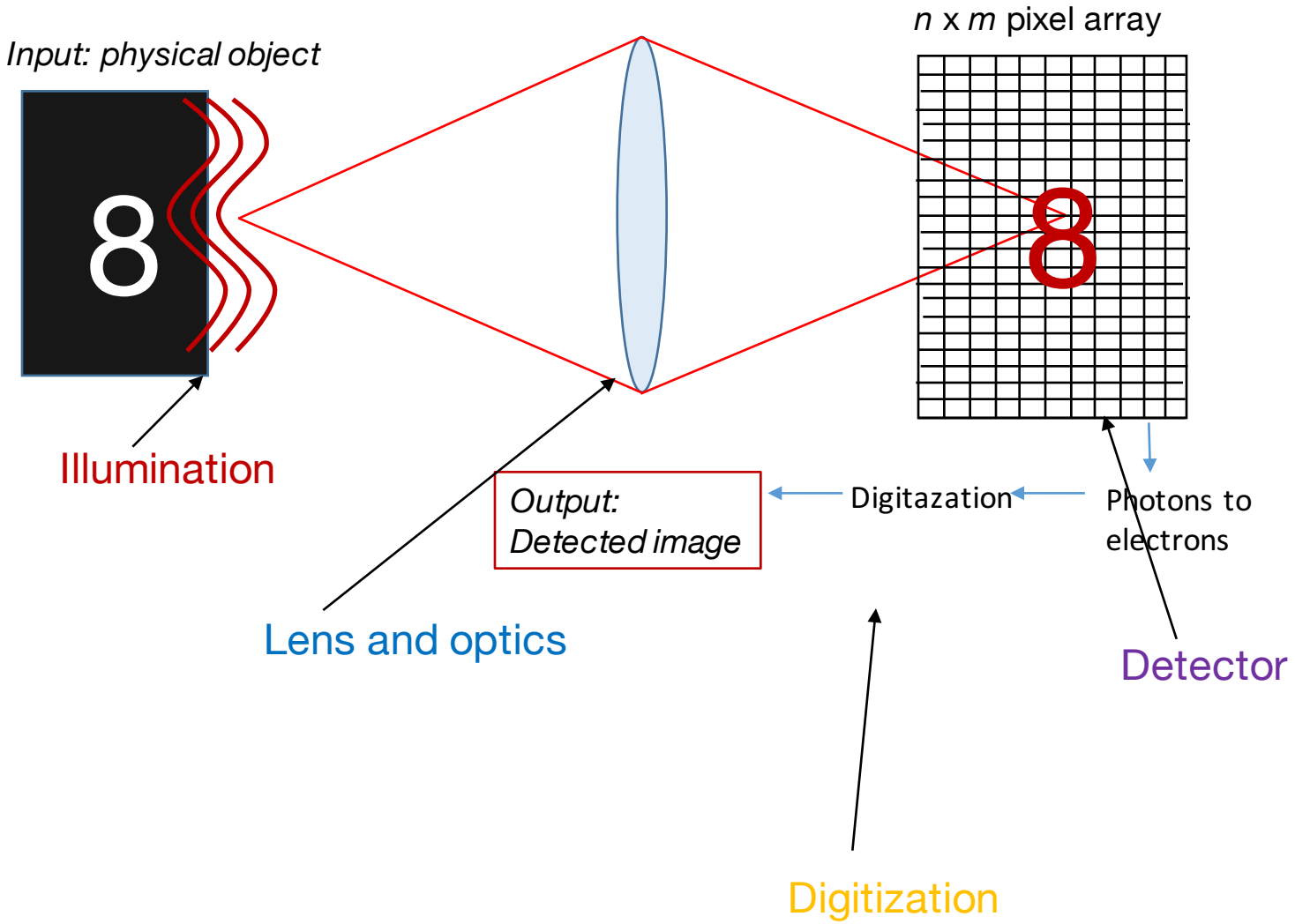
Digital layers

Physical layers

$$\text{Task} = W_n \ldots \text{ReLU}[W_1 \, \text{ReLU}[W_0 \, f[I_0]] \ldots]$$

What physical parameters effect image formation f[ ]?

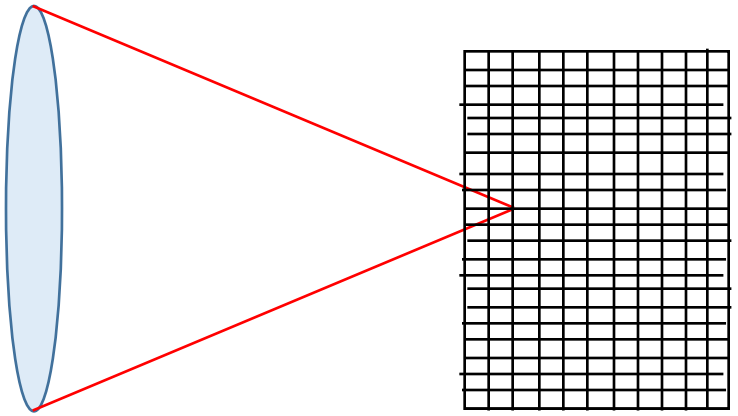
- Illumination
  - Spatial pattern
  - Angle of incidence
  - Color, polarization
- Lens and optics
  - Position/orientation
  - Shape
  - Focus
  - Transparency
- Detector
  - Pixel size
  - Pixel shape & fill factor
  - Color filters
  - Other filters
- Digitization
  - E to P curves
  - Digitization schemes/thresholds
  - Data transmission, multiplexing
- Physical object



Example #1: Optimized color filter for a grayscale camera

“Ground truth” object:
$I_0(x, y, \lambda)$
100 x 100 pix. x 30 spectral channels

Example #1: Optimized color filter for a grayscale camera

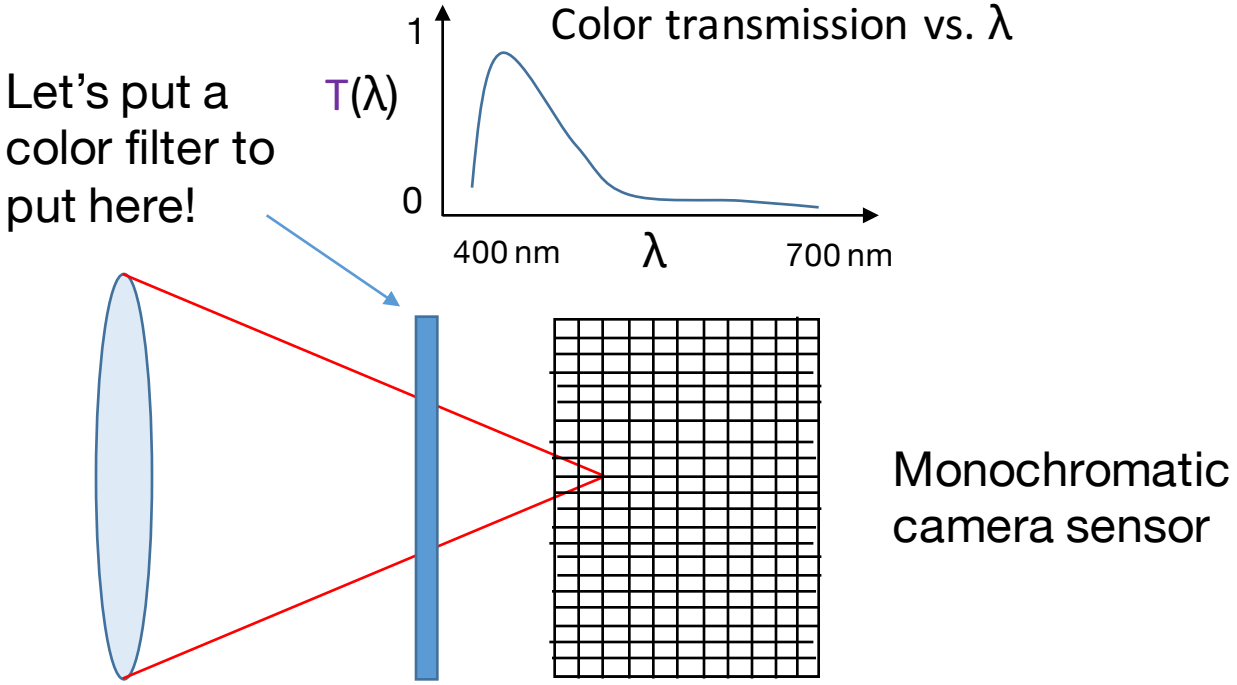


Monochromatic
camera sensor

"Ground truth" object:
$I_0(x, y, \lambda)$
100 x 100 pix. x 30 spectral channels

$$I_s(x, y) = \sum_{\lambda} I_0(x, y, \lambda)$$

Example #1: Optimized color filter for a grayscale camera

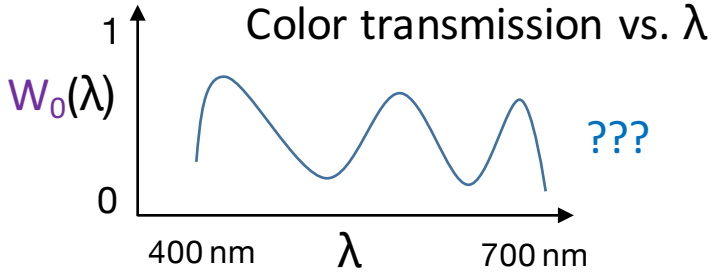


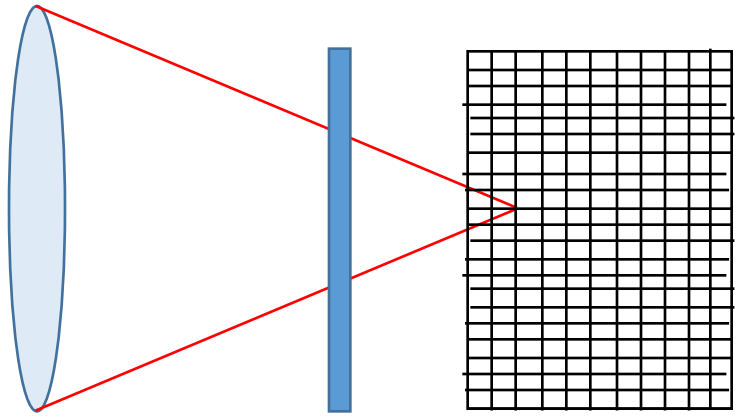
"Ground truth" object:
$I_0(x, y, \lambda)$
100 x 100 pix. x 30 spectral channels

$$I_s(x, y) = \sum_{\lambda} T(\lambda)\, I_0(x, y, \lambda)$$

Example #1: Optimized color filter for a grayscale camera

Design optimal color filter for classification:





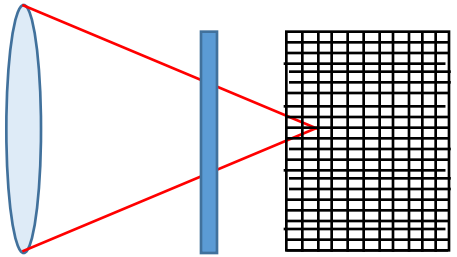
Monochromatic camera sensor

Training data:
$[I_0(x, y, \lambda), y]$
I_0:100 x 100 pix. x 30
Label y: 1x3 - pepper, broccoli, green beans

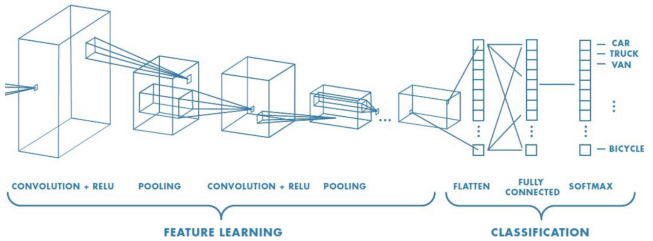
$$I_s(x, y) = \sum_{\lambda} W_0(\lambda)\, I_0(x, y, \lambda)$$

Physical Layer

Example #1: Optimized color filter for a grayscale camera



$[I_s(x, y), y]$



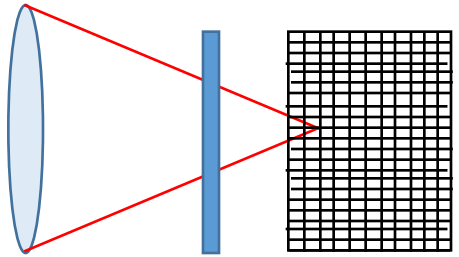
Training data:
$[I_0(x, y, \lambda), y]$
I_0:100 x 100 x 30
Label y: 1x3

$$I_s(x, y) = \sum_{\lambda} W_0(\lambda)\, I_0(x, y, \lambda)$$

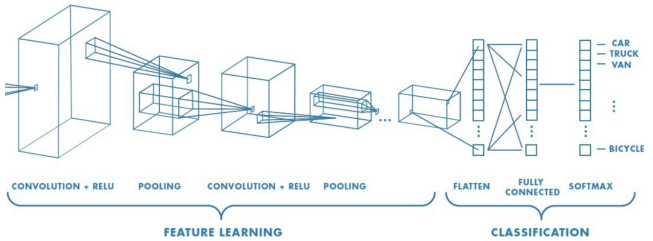
Physical Layer

Task = $\mathbf{W_n} \ldots \mathrm{ReLU}[\mathbf{W_1}\, \mathrm{ReLU}[\mathbf{W_0}\, I_s] \ldots]$

Example #1: Optimized color filter for a grayscale camera



$[I_s(x, y), y]$



Training data:
$[I_0(x, y, \lambda), y]$
I_0:100 x 100 x 30
Label y: 1x3

$$I_s(x, y) = \sum_{\lambda} W_0(\lambda)\, I_0(x, y, \lambda)$$

Physical Layer

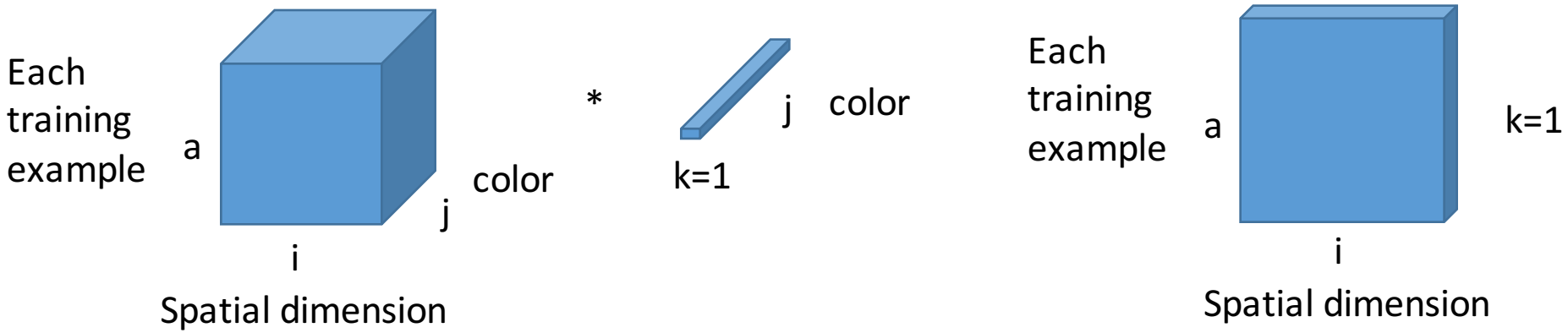
Task = $\mathbf{W_n}$ …ReLU[$\mathbf{W_1}$ ReLU[$\mathbf{W_0}$ I_s] …]

```
multispectral_data = tf.placeholder(tf.float32, [None, num_colors, image_size])
veg_labels = tf.placeholder(tf.float32, [None, 3])
filter_weights = tf.truncated_normal([num_colors, 1], stddev = 0.1)
filtered_images = tf.einsum('aij,jk->aik', multispectral_data, filter_weights)

#Now, train CNN of your choice using [filtered_images, veg_labels]
#Output of training/testing will be CNN weights, classification performance AND filter_weights!
```

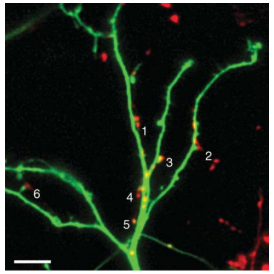
Tensorflow einsum – pretty useful for quick linear combo's

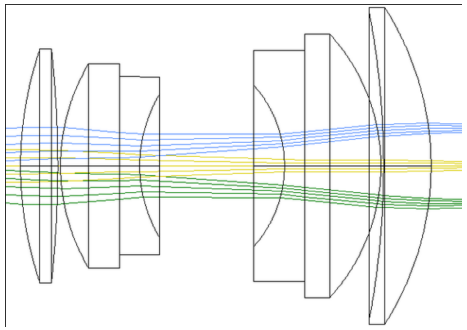
```
filtered_images = tf.einsum('aij,jk->aik', multispectral_data, filter_weights)
```



First model of light we'll consider – Incoherent light

- Interpretation #1: Radiation (*Incoherent*)
- Model: Rays

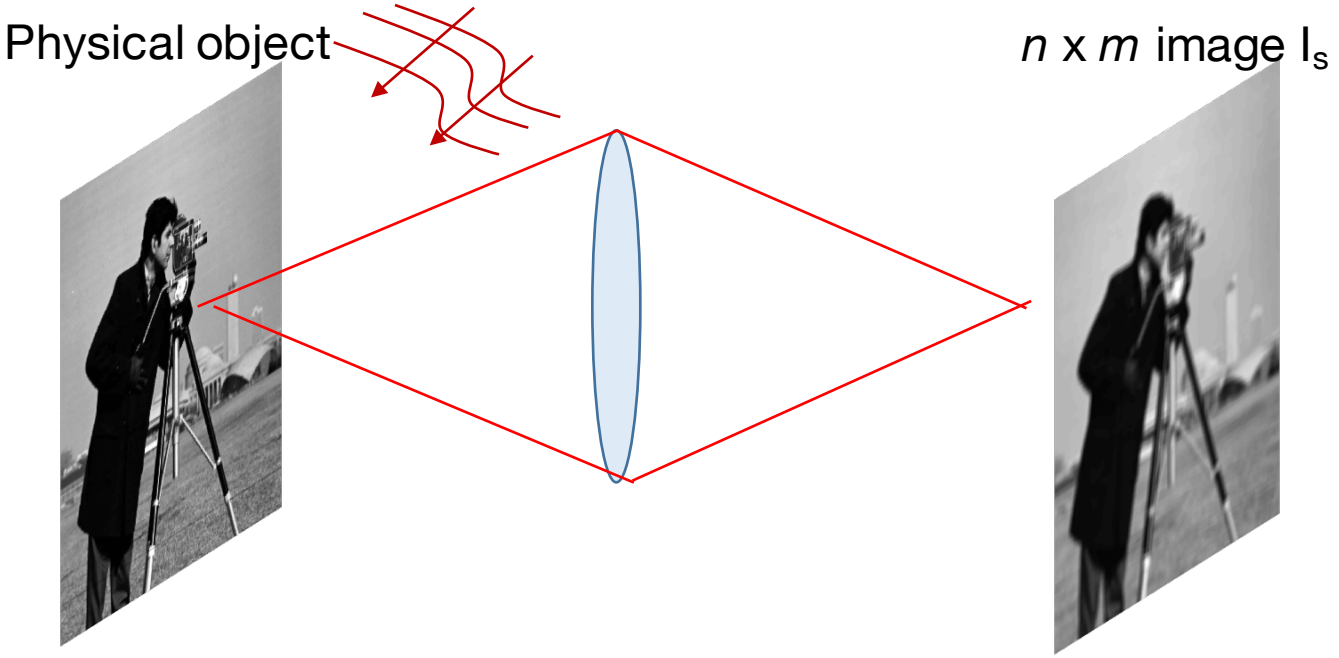




- Real, non-negative
- Models absorption and brightness

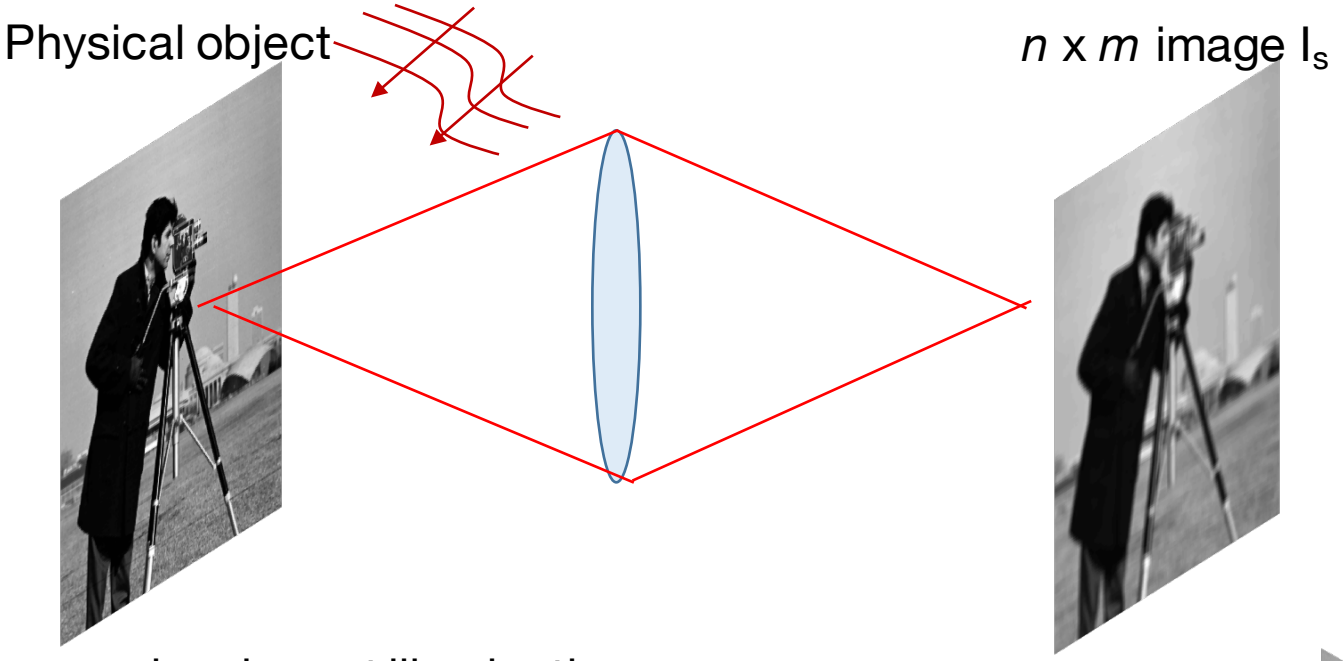
$$I_{tot} = I_1 + I_2$$

Example #2: Optimized illumination pattern (one color)

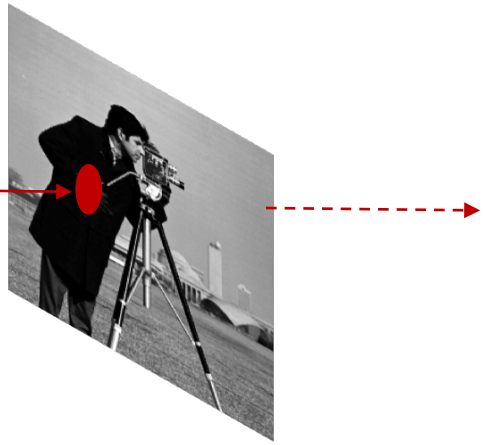


- Assume incoherent illumination
- Assume thin 2D object
- Object is real, non-negative map of absorption/reflectivity

Example #2: Optimized illumination pattern (one color)

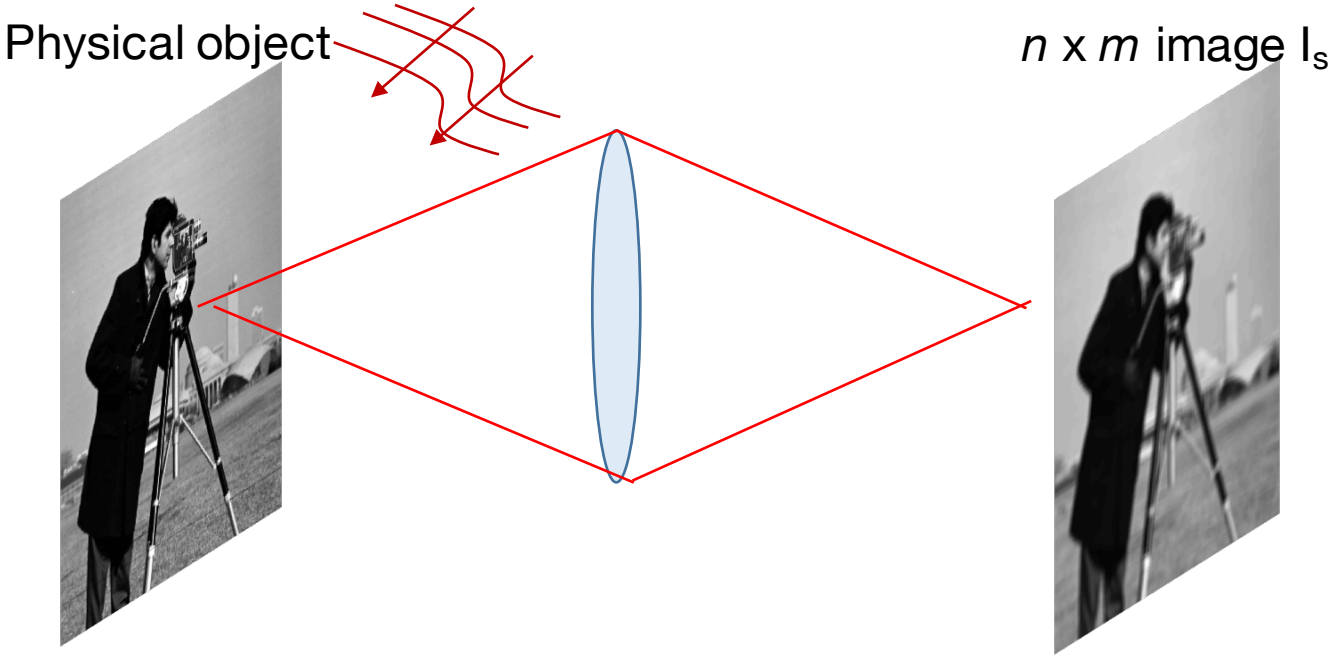


- Assume incoherent illumination
- Assume thin 2D object
- Object is real, non-negative map of absorption/reflectivity



Object absorption: $\mathbf{I_0}(x,y)$
Illumination pattern: $\mathbf{s}(x,y)$
Light exiting object surface: $\mathbf{I_e}(x,y) = \mathbf{I_0}(x,y) \circ \mathbf{s}(x,y)$

Example #2: Optimized illumination pattern (one color)

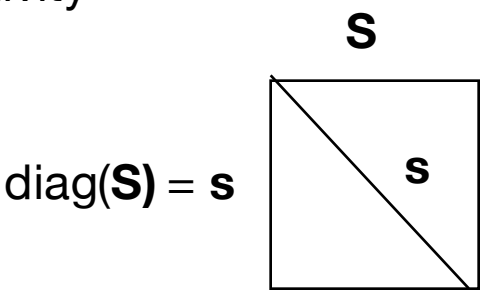


- Assume incoherent illumination
- Assume thin 2D object
- Object is real, non-negative map of absorption/reflectivity

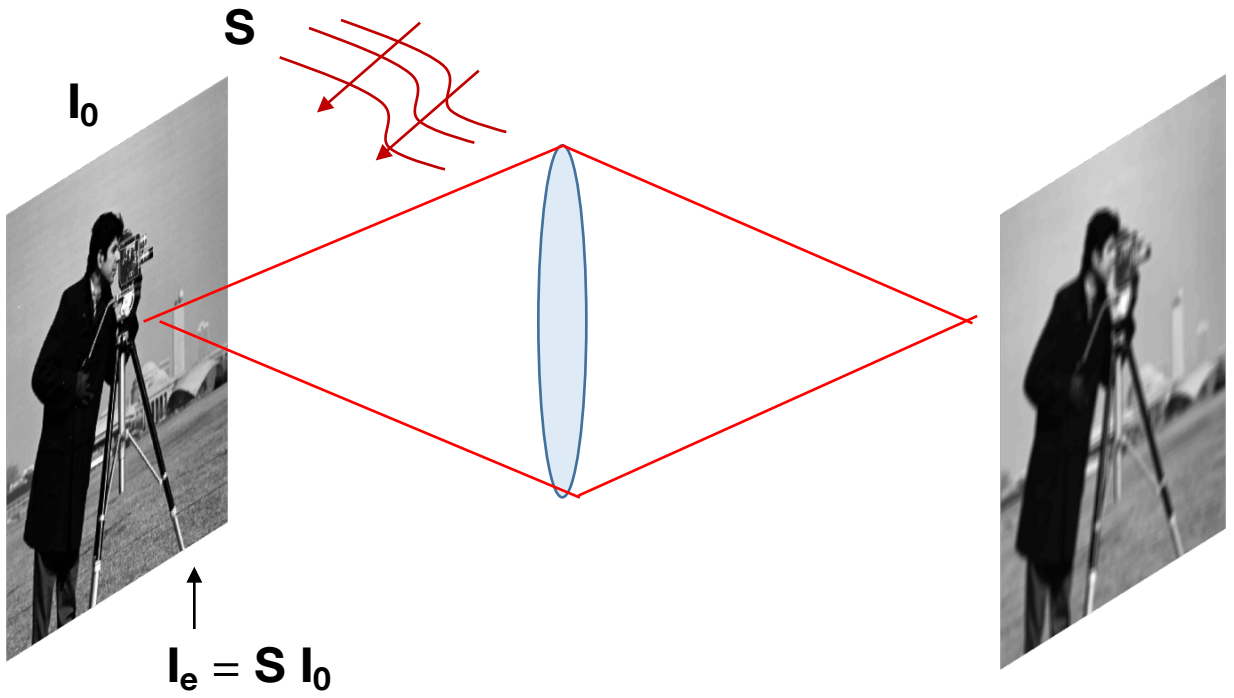
Modeling incoherent illumination

$$\mathbf{I_e}(x,y) = \mathbf{I_0}(x,y) \circ \mathbf{s}(x,y)$$

$$\mathbf{I_e} = \mathbf{S}\,\mathbf{I_0}$$

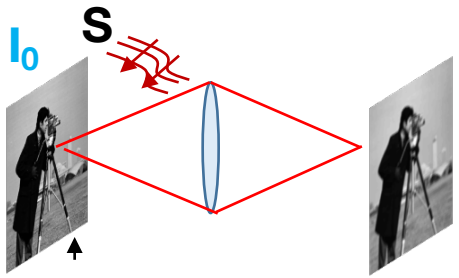


Example #2: Optimized illumination pattern (one color)

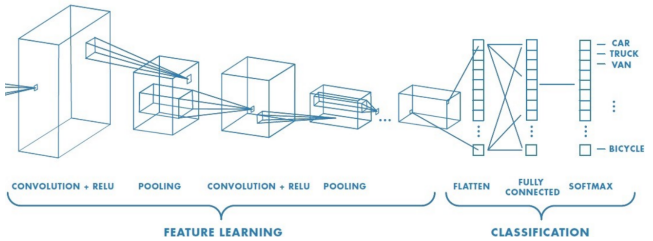


First, assume perfect camera:
intensity at image plane $I_p = I_e = S\ I_0$

Example #2: Optimized illumination pattern (one color)



$[I_e(x, y), y]$



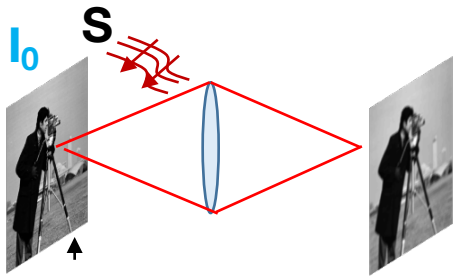
Training data:
$[I_0(x, y), y]$
I_0:100 x 100
Label y: 1x2

$I_e(x,y) = \mathbf{S}\ I_0(x,y)$

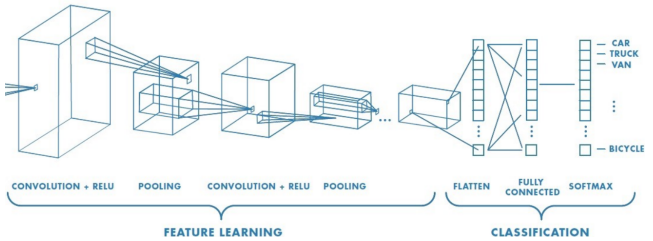
Physical Layer

Task = $\mathbf{W_n} \ldots \mathrm{ReLU}[\mathbf{W_1}\ \mathrm{ReLU}[\mathbf{W_0}\ I_e] \ldots]$

Example #2: Optimized illumination pattern (one color)



$[I_e(x, y), y]$



Training data:
$[I_0(x, y), y]$
I_0:100 x 100
Label y: 1x2

$I_e(x,y) = \mathbf{S}\ I_0(x,y)$

Physical Layer

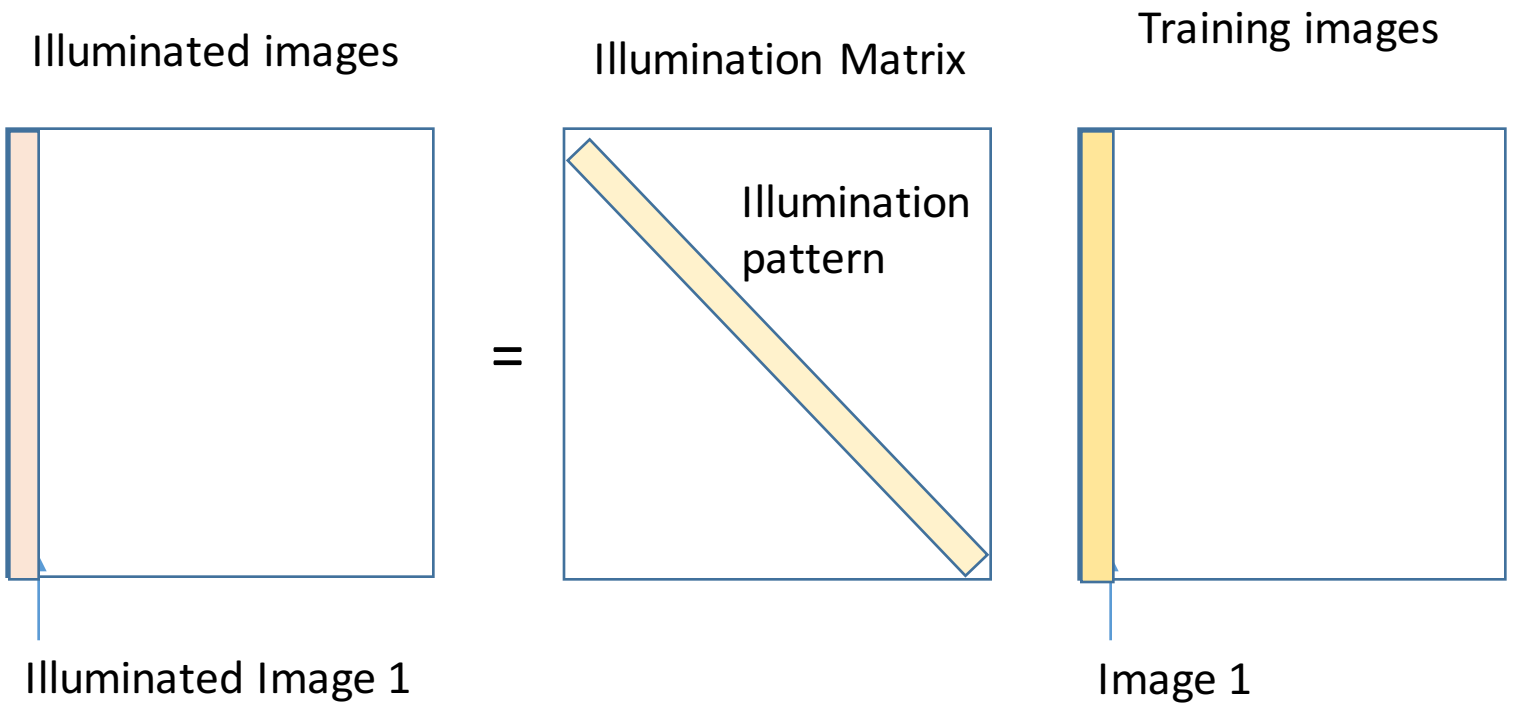
Task = $\mathbf{W_n}$ …ReLU[$\mathbf{W_1}$ ReLU[$\mathbf{W_0}\ I_e$] …]

```
training_images = tf.placeholder(tf.float32, [image_size, None])
training_labels = tf.placeholder(tf.float32, [None, 3])
illumination_pattern = tf.truncated_normal([image_size, 1], stddev = 0.1)
illumination_matrix = tf.linalg.diag(illumination_pattern)
illumianted_images = tf.matmul(illumination_matrix, training_images)

#Now, train CNN of your choice using [illumianted_images, training_labels]
#Output of train/test will be CNN weights, classification performance AND illumination_pattern!
```

```
training_images = tf.placeholder(tf.float32, [image_size, None])
training_labels = tf.placeholder(tf.float32, [None, 3])
illumination_pattern = tf.truncated_normal([image_size, 1], stddev = 0.1)
illumination_matrix = tf.linalg.diag(illumination_pattern)
illumianted_images = tf.matmul(illumination_matrix, training_images)

#Now, train CNN of your choice using [illumianted_images, training_labels]
#Output of train/test will be CNN weights, classification performance AND illumination_pattern!
```



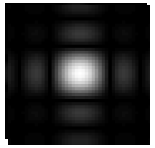
We can also add in some lens blur

Lenses blur and rescale images:
(We'll learn how exactly next few weeks)

Input intensity

Convolution filter h

$*$



$=$

Output intensity

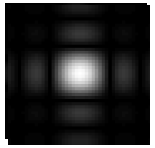
We can also add in some lens blur

Lenses blur and rescale images:
(We'll learn how exactly next few weeks)

Input intensity

Convolution filter h

$*$



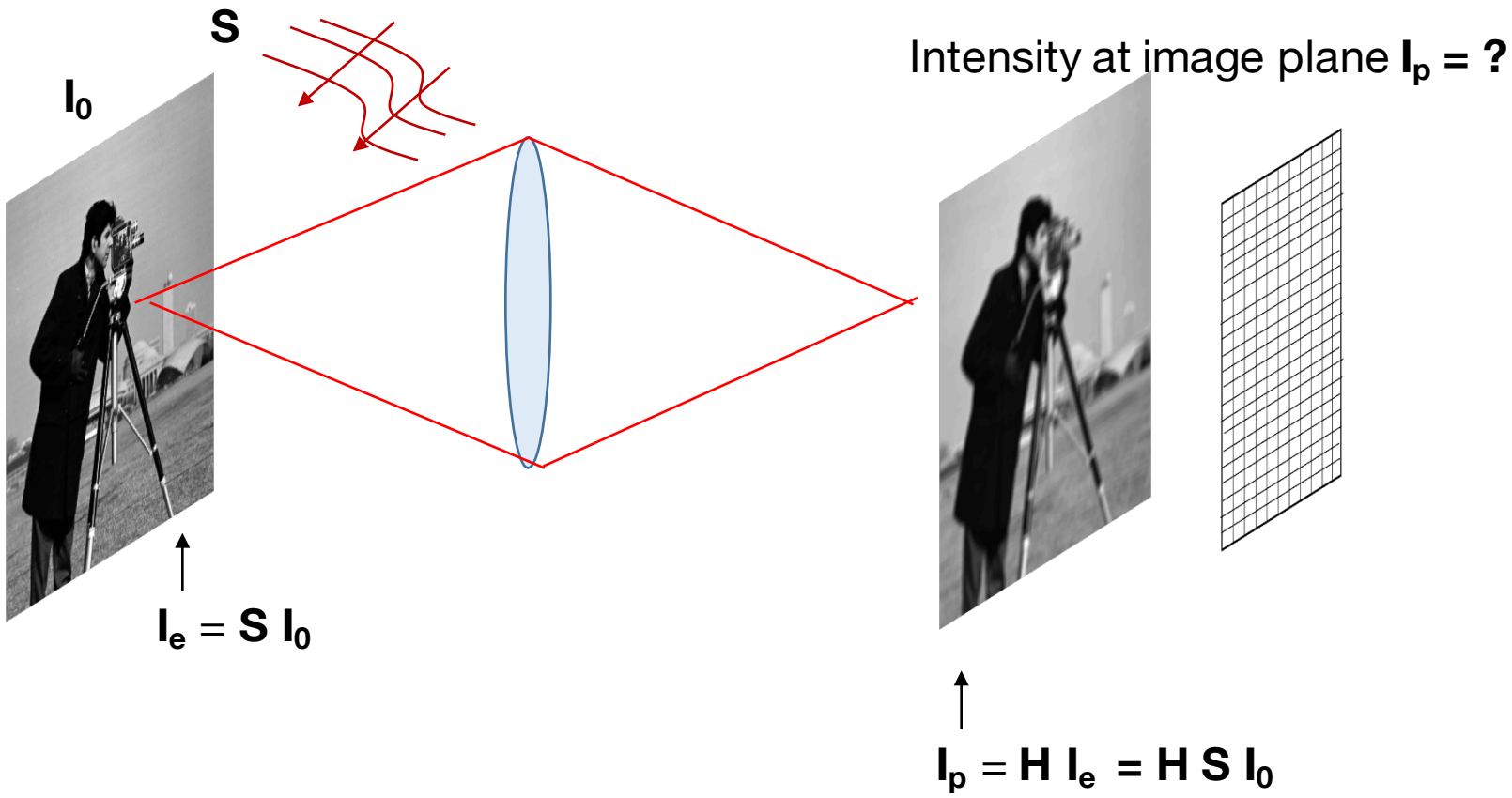
$=$

Output intensity

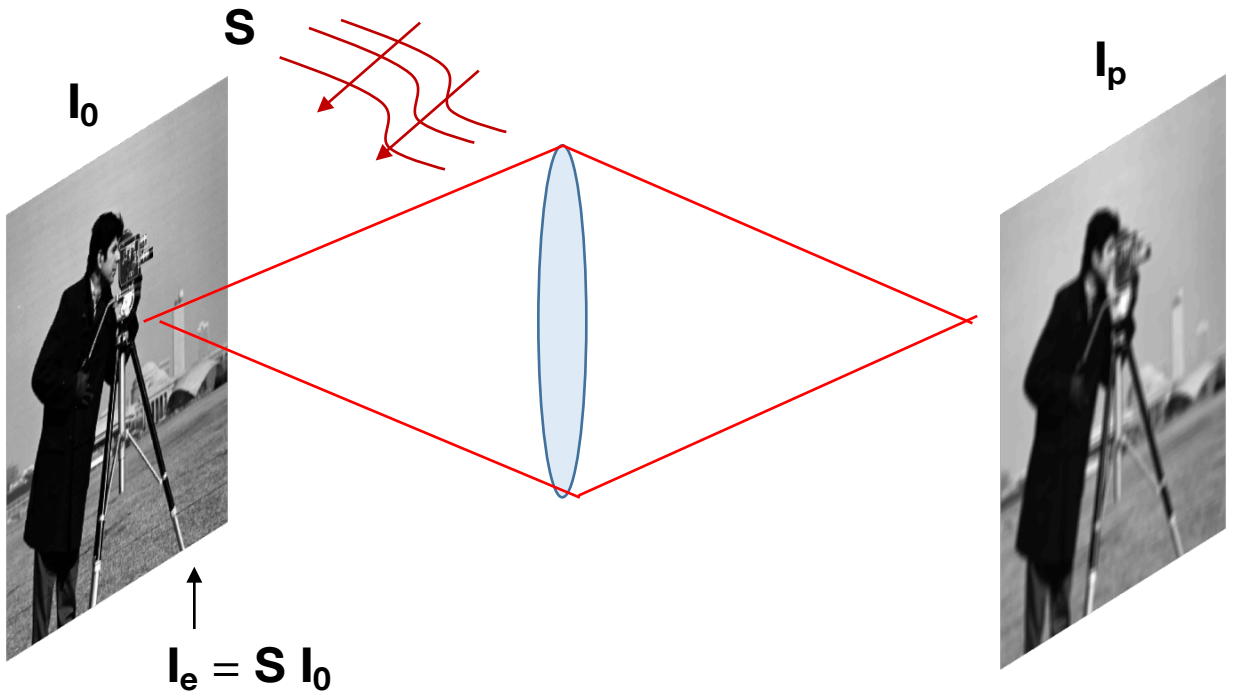
$\mathbf{I_e}(x/M,y/M) \quad * \quad \mathbf{h}(x,y) \quad = \quad \mathbf{I_p}(x,y)$

Assuming we've resized by M, $\mathbf{I_p = I_e * h = H I_e}$

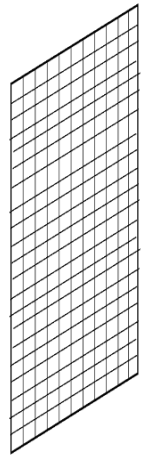
Simple mathematical model of image formation



Simple mathematical model of image formation



Discretization by detector ?



$$I_s = D\ I_p = D\ H\ S\ I_0$$

Use downsampling matrix (sum-pooling)

$$D = \begin{bmatrix} 0.5\ 0.5\ 0\ 0\ 0\ \ldots. & & \\ & 0.5\ 0.5\ 0\ 0\ 0\ \ldots. & \\ & & 0.5\ 0.5\ 0\ 0\ 0\ \ldots. \\ \vdots & & \vdots \end{bmatrix}$$

Physical Layers

Digital Layers

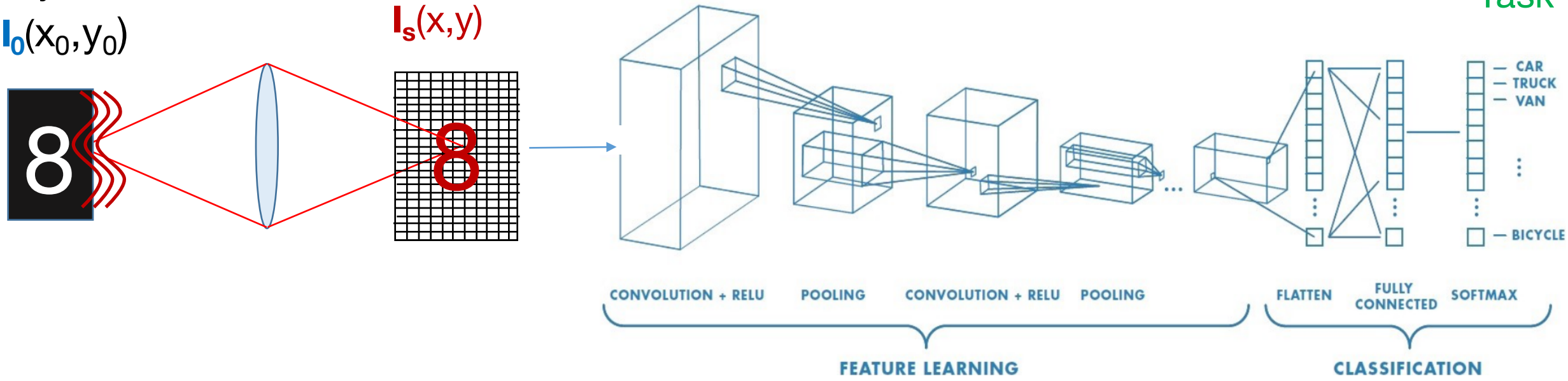
Physical world

$I_0(x_0,y_0)$

Digital Image

$I_s(x,y)$

Task



Digital layers

Physical layers

$$\text{Task} = \mathbf{W}_n \ldots \text{ReLU}[\mathbf{W}_1\ \text{ReLU}[\mathbf{W}_0\ f[\mathbf{I}_0]]\ldots]$$

$$\text{Task} = \mathbf{W}_n \ldots \text{ReLU}[\mathbf{W}_1\ \text{ReLU}[\mathbf{W}_0\ \mathbf{D}\ \mathbf{H}\ \mathbf{S}\ \mathbf{I}_0]\ldots]$$

Summary: simple physical layers for incoherent imaging

- Deal with sample/image intensities I, real and non-negative
- Different colors add linearly

$$I_s(x, y) = \sum_{\lambda} I_0(x, y, \lambda)$$

- Effect of illumination is element-wise multiplication

$$I_e(x,y) = \mathbf{S}\, I_0(x,y)$$

- Imaging systems blur the object via point-spread function matrix **H**

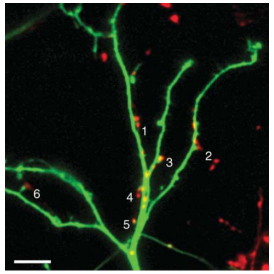
$$I_b(x,y) = \mathbf{H}\, I_0(x,y)$$

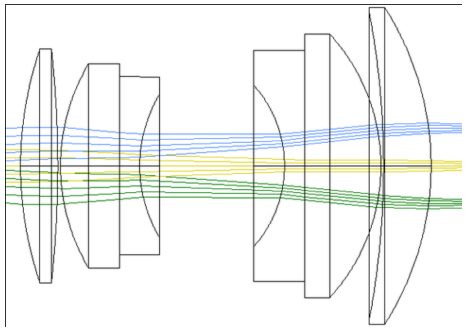
- Discrete pixels down-sample the object via

$$I_d(x,y) = \mathbf{D}\, I_0(x,y)$$

First - what is light and how can we model it?

- Interpretation #1: Radiation (*Incoherent*)
- Model: Rays

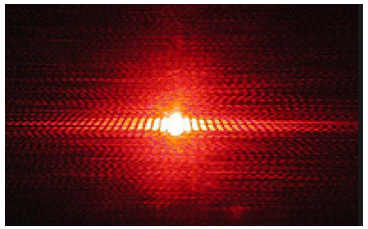


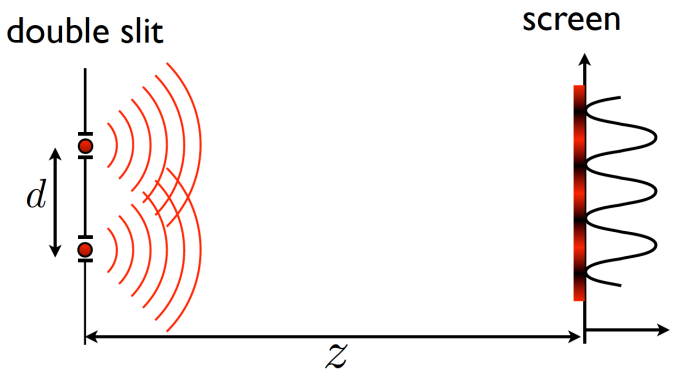


- Real, non-negative
- Models absorption and brightness

$$I_{tot} = I_1 + I_2$$

- Interpretation #2: Electromagnetic wave (*Coherent*)
- Model: Waves





- Complex field
- Models Interference

$$E_{tot} = E_1 + E_2$$

Let's take a step back: how does light propagate?

Maxwell's equations without any charge

$$\nabla \times \vec{\mathcal{E}} = -\mu \frac{\partial \vec{\mathcal{H}}}{\partial t}$$

$$\nabla \times \vec{\mathcal{H}} = \epsilon \frac{\partial \vec{\mathcal{E}}}{\partial t}$$

$$\nabla \cdot \epsilon \vec{\mathcal{E}} = 0$$

$$\nabla \cdot \mu \vec{\mathcal{H}} = 0.$$

Let's take a step back: how does light propagate?

Maxwell's equations without any charge

$$\nabla \times \vec{\mathcal{E}} = -\mu \frac{\partial \vec{\mathcal{H}}}{\partial t}$$

$$\nabla \times \vec{\mathcal{H}} = \epsilon \frac{\partial \vec{\mathcal{E}}}{\partial t}$$

$$\nabla \cdot \epsilon \vec{\mathcal{E}} = 0$$

$$\nabla \cdot \mu \vec{\mathcal{H}} = 0.$$

1. Take the curl of both sides of first equation
2. Substitute 2nd and 3rd equation
3. Arrive at the wave equation:

$$\nabla^2 \vec{\mathcal{E}} - \frac{n^2}{c^2} \frac{\partial^2 \vec{\mathcal{E}}}{\partial t^2} = 0 \qquad n = \left(\frac{\epsilon}{\epsilon_0} \right)^{1/2} \qquad c = \frac{1}{\sqrt{\mu_0 \epsilon_0}}.$$

Let's take a step back: how does light propagate?

Considering light that isn't pulsed over time, we can use the following solution:

$$u(P, t) = A(P) \cos[2\pi\nu t + \phi(P)]$$

$$u(P, t) = Re\{U(P) \exp(-j2\pi\nu t)\},$$

Let's take a step back: how does light propagate?

Considering light that isn't pulsed over time, we can use the following solution:

$$u(P,t) = A(P)\cos[2\pi\nu t + \phi(P)]$$

$$u(P,t) = Re\{U(P)\exp(-j2\pi\nu t)\},$$

With this particular solution, we get the following important time-independent equation:

Helmholtz Equation

$$(\nabla^2 + k^2)U = 0.$$

$$k = 2\pi n\frac{\nu}{c} = \frac{2\pi}{\lambda},$$

Let's take a step back: how does light propagate?

Considering light that isn't pulsed over time, we can use the following solution:

$$u(P,t) = A(P)\cos[2\pi\nu t + \phi(P)]$$

$$u(P,t) = Re\{U(P)\exp(-j2\pi\nu t)\},$$

With this particular solution, we get the following important time-independent equation:

Helmholtz Equation

$$(\nabla^2 + k^2)U = 0.$$

$$k = 2\pi n\frac{\nu}{c} = \frac{2\pi}{\lambda},$$

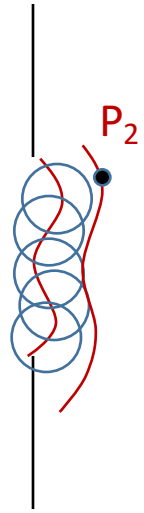
This is an important equation in physics. We won't go into the details, but it leads to the Huygen-Fresnel principle:

$$U(P_2) = \frac{1}{j\lambda}\iint_{\Sigma} U(P_1)\frac{\exp(jkr_{21})}{r_{21}}\cos\theta\, ds$$

Plane-to-plane light propagation via the ”paraxial approximation”

The Huygens-Fresnel Equation

$$U(P_2) = \frac{1}{j\lambda} \iint_{\Sigma} U(P_1) \frac{\exp(jkr_{21})}{r_{21}} \cos\theta \, ds$$

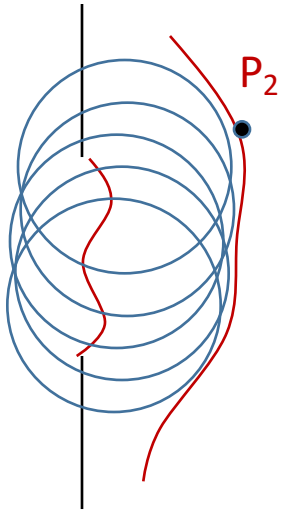


Aperture

Plane-to-plane light propagation via the ”paraxial approximation”

The Huygens-Fresnel Equation

$$U(P_2) = \frac{1}{j\lambda} \iint_{\Sigma} U(P_1) \frac{\exp(jkr_{21})}{r_{21}} \cos\theta \, ds$$



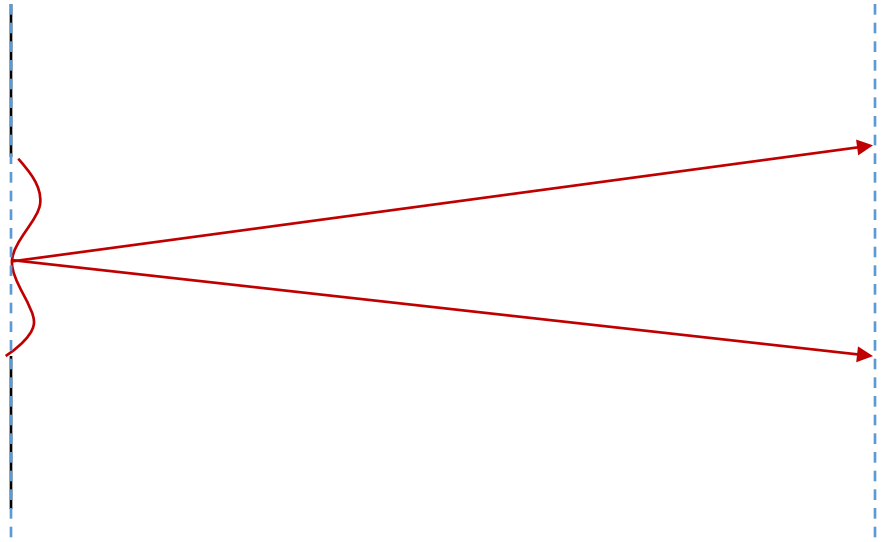
Generally connects two points in 3D:

$$U(P_1) = U(x_1, y_1, z_1)$$

$$U(P_2) = U(x_2, y_2, z_2)$$

Plane-to-plane light propagation via the ”paraxial approximation”

We are usually concerned about propagation between two planes (almost always in an optical system):

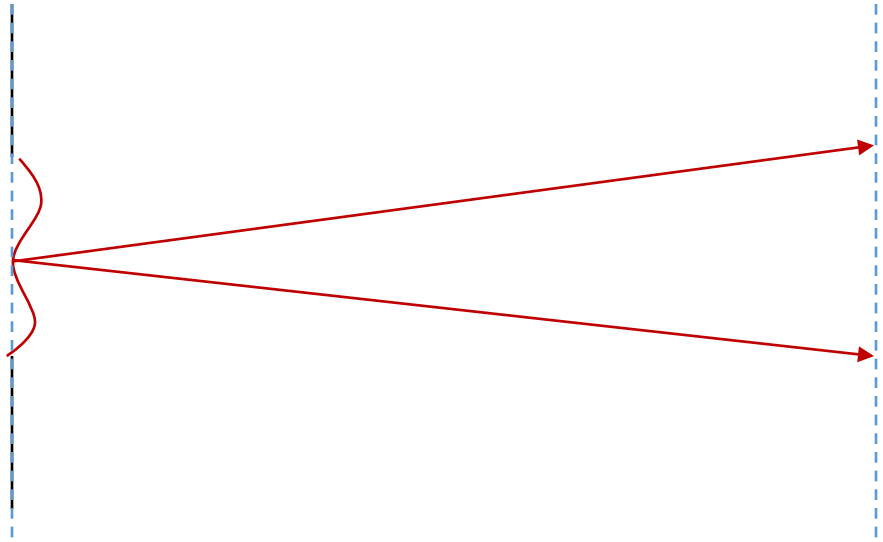


$$U(P_1) = U(x_1, y_1, z_1 = z_{p1}) \qquad U(P_2) = U(x_2, y_2, z_2 = z_{p2})$$

$$U(P) = E(x, y, z)e^{ikz}$$

Plane-to-plane light propagation via the ”paraxial approximation”

We are usually concerned about propagation between two planes (almost always in an optical system):



Paraxial approximation:

$$\nabla_{\perp}^2 U + 2ik\frac{dU}{dz} = 0$$

$$\nabla_{\perp}^2 \stackrel{\text{def}}{=} \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$$

$$U(P_1) = U(x_1, y_1, z_1 = z_{p1}) \qquad U(P_2) = U(x_2, y_2, z_2 = z_{p2})$$

$$U(P) = E(x, y, z)e^{ikz}$$

Plane-to-plane light propagation via the ”paraxial approximation”

We are usually concerned about propagation between two planes (almost always in an optical system):

Paraxial approximation:

$$\nabla_\perp^2 U + 2ik\frac{dU}{dz} = 0$$

Substitute in $U(P) = E(x, y, z)e^{ikz}$ and crank the wheel,

$$\nabla_\perp^2 E + 2ik\frac{dE}{dz} + 2k^2 E = 0$$

Paraxial Helmholtz Equation. This has an exact integral solution:

Plane-to-plane light propagation via the ”paraxial approximation”

We are usually concerned about propagation between two planes (almost always in an optical system):

Paraxial approximation:

$$\nabla_\perp^2 U + 2ik\frac{dU}{dz} = 0$$

Substitute in $U(P) = E(x,y,z)e^{ikz}$ and crank the wheel,

$$\nabla_\perp^2 E + 2ik\frac{dE}{dz} + 2k^2 E = 0$$

Paraxial Helmholtz Equation. This has an exact integral solution:

$$E(x,y,z) = \frac{e^{ikz}}{i\lambda z}\iint_{-\infty}^{+\infty} E(x',y',0)e^{\frac{ik}{2z}\left[(x-x')^2+(y-y')^2\right]}dx'\,dy'$$

Fresnel diffraction integral

This is how light propagates from one plane to the next. It’s a convolution!

Fresnel light propagation as a convolution

$$E(x,y,z) = \frac{e^{ikz}}{i\lambda z} \iint_{-\infty}^{+\infty} E(x',y',0) e^{\frac{ik}{2z}\left[(x-x')^2+(y-y')^2\right]} dx'\,dy'$$

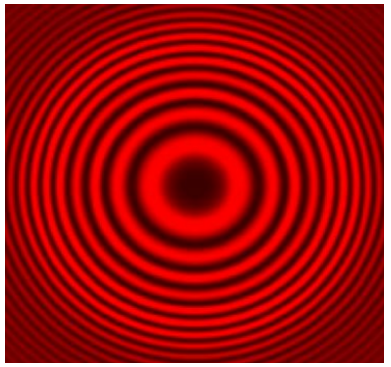
$$h(x,y,z) = \frac{e^{ikz}}{i\lambda z} e^{i\frac{k}{2z}(x^2+y^2)}$$

$$E(x,y,z) = E(x,y,0) * h(x,y,z)$$

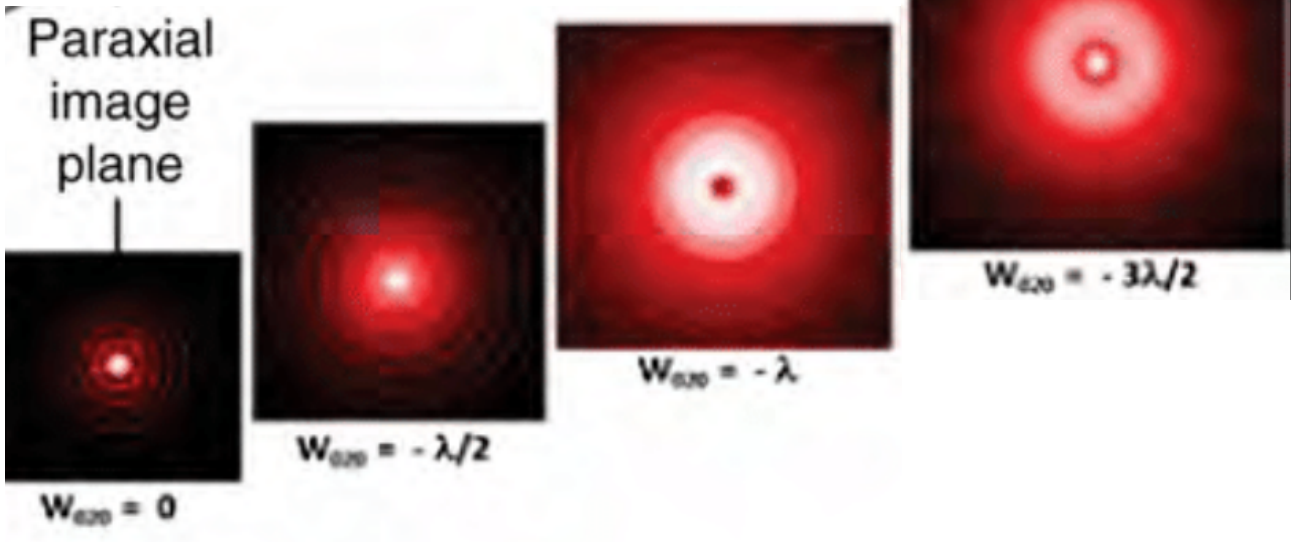
Fresnel light propagation as a convolution

$$E(x,y,z) = \frac{e^{ikz}}{i\lambda z} \iint_{-\infty}^{+\infty} E(x',y',0) e^{\frac{ik}{2z}\left[(x-x')^2+(y-y')^2\right]} dx'\,dy'$$

$$h(x,y,z) = \frac{e^{ikz}}{i\lambda z} e^{i\frac{k}{2z}(x^2+y^2)}$$



$$E(x,y,z) = E(x,y,0) * h(x,y,z)$$



From the Fresnel approximation to the Fraunhofer approximation

Fresnel Approximation:

$$E(x,y,z) = \frac{e^{ikz}}{i\lambda z}\iint_{-\infty}^{+\infty} E(x',y',0)e^{\frac{ik}{2z}\left[(x-x')^2+(y-y')^2\right]}dx'\,dy'$$

Lets assume that the second plane is “pretty far away” from the first plane. Then,

$$z > \frac{2D^2}{\lambda}$$

From the Fresnel approximation to the Fraunhofer approximation

Fresnel Approximation:

$$E(x,y,z) = \frac{e^{ikz}}{i\lambda z} \iint_{-\infty}^{+\infty} E(x',y',0) e^{\frac{ik}{2z}\left[(x-x')^2+(y-y')^2\right]} dx'\,dy'$$

Lets assume that the second plane is “pretty far away” from the first plane. Then,

$$z > \frac{2D^2}{\lambda}$$

1. Expand the squaring

$$E(x,y,z) = \frac{e^{ikz}}{i\lambda z} \iint E(x',y',0) e^{\frac{ik}{2z}(x^2+y^2)} e^{\frac{ik}{2z}(x'^2+y'^2)} e^{\frac{ik}{2z}(xx'+yy')} dx'\,dy'$$

From the Fresnel approximation to the Fraunhofer approximation

Fresnel Approximation:

$$E(x,y,z) = \frac{e^{ikz}}{i\lambda z}\iint_{-\infty}^{+\infty} E(x',y',0)e^{\frac{ik}{2z}\left[(x-x')^2+(y-y')^2\right]}dx'dy'$$

Lets assume that the second plane is "pretty far away" from the first plane. Then,

$$z > \frac{2D^2}{\lambda}$$

1. Expand the squaring

$$E(x,y,z) = \frac{e^{ikz}}{i\lambda z}\iint E(x',y',0)e^{\frac{ik}{2z}(x^2+y^2)}e^{\frac{ik}{2z}(x'^2+y'^2)}e^{\frac{ik}{2z}(xx'+yy')}dx'dy'$$

2. Front term comes out, assume second term goes away, then,

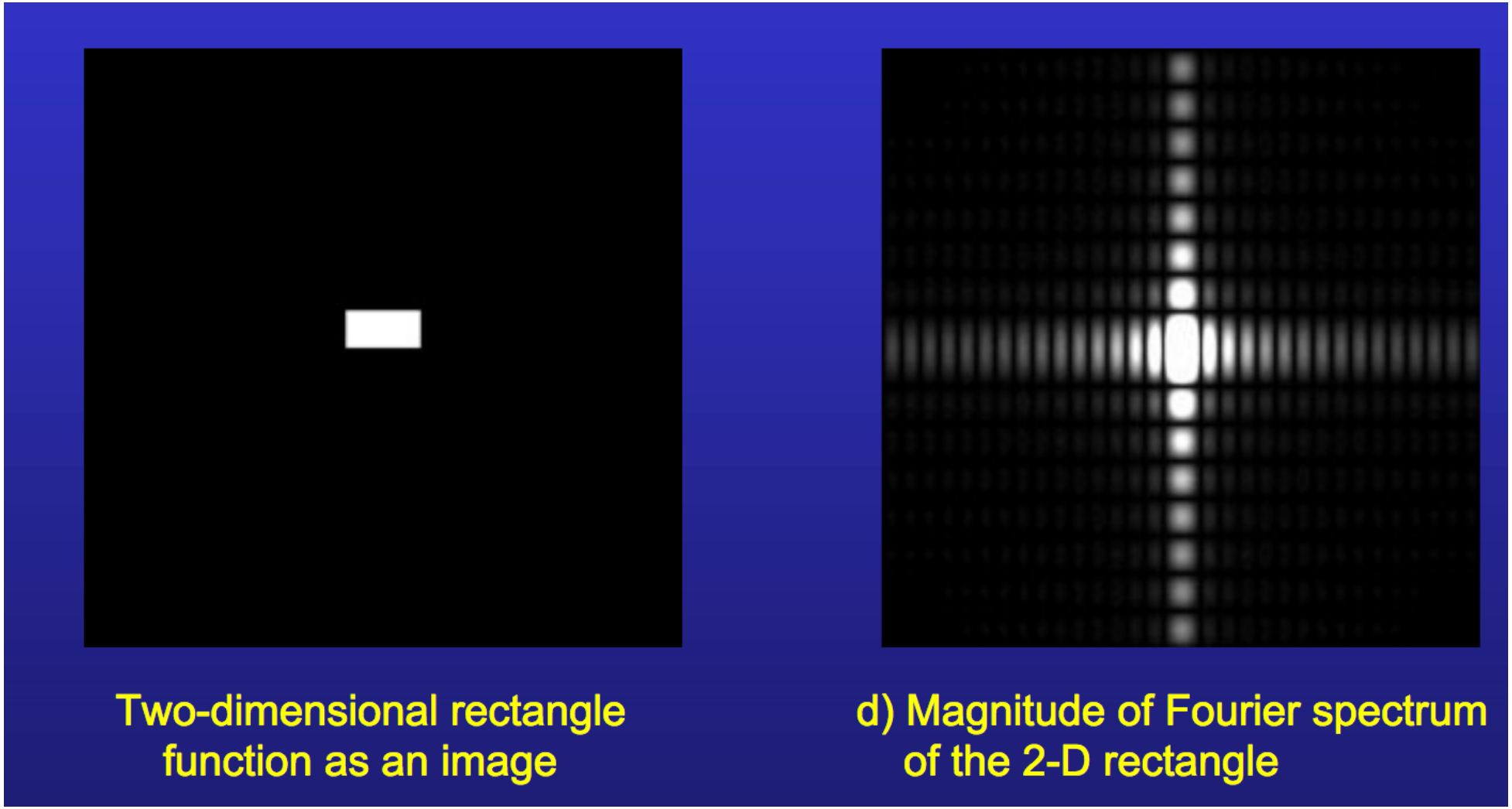
$$E(x,y,z) = C\iint E(x',y',0)e^{\frac{ik}{2z}(xx'+yy')}dx'dy'$$

$$C = \frac{e^{ikz}}{i\lambda z}e^{\frac{ik}{2z}(x^2+y^2)}$$

Fraunhofer diffraction is a Fourier transform!!!!!!!

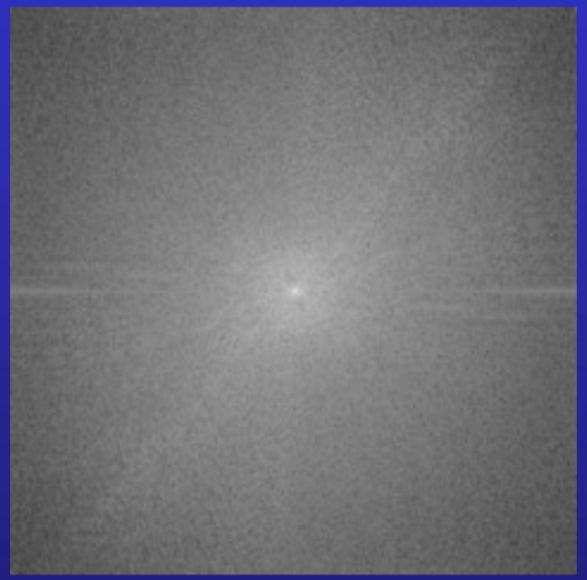
This is the aperture

This is what you see far away

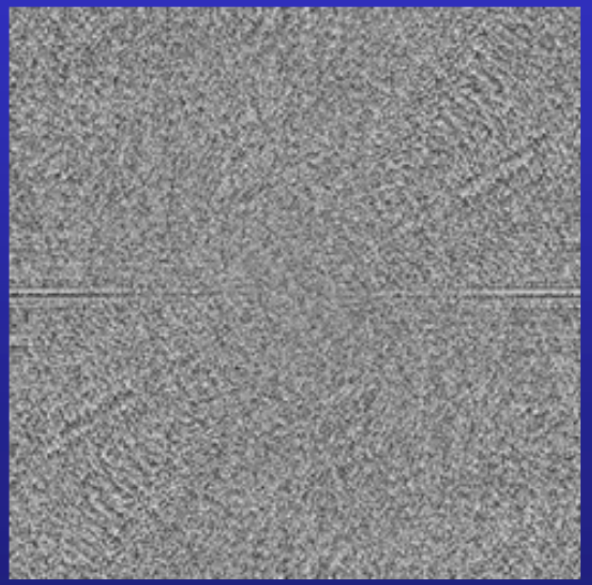


Cheetah

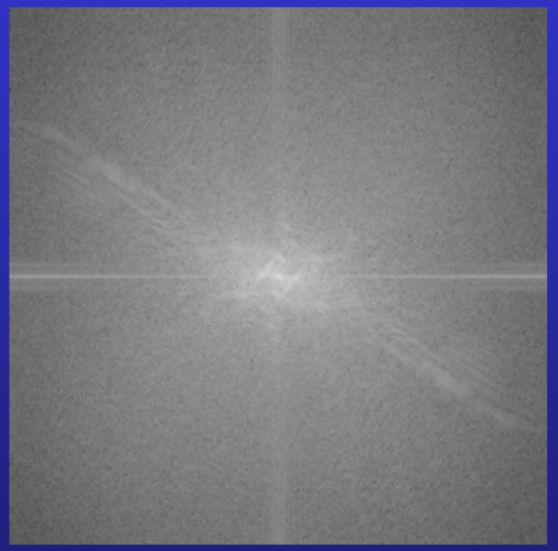
Zebra



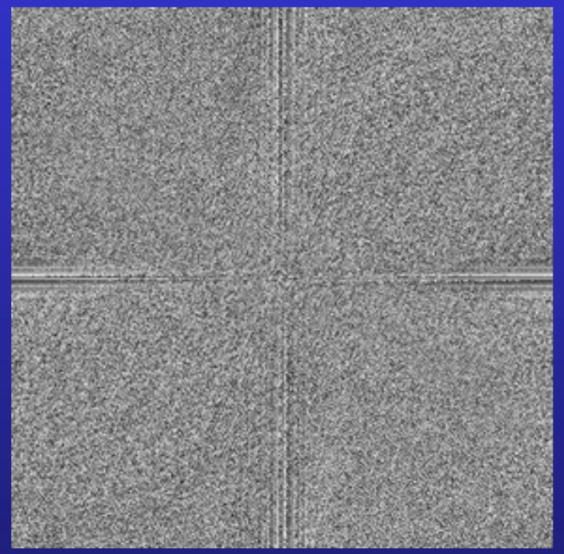
magnitude of cheetah



phase of cheetah



magnitude of zebra



phase of zebra