

# Lecture 14: Feedback Loops in Camera: The 3A Algorithms

---

**Yuhao Zhu**

<http://yuhaozhu.com>

CSC 292/572, Fall 2022  
Mobile Visual Computing

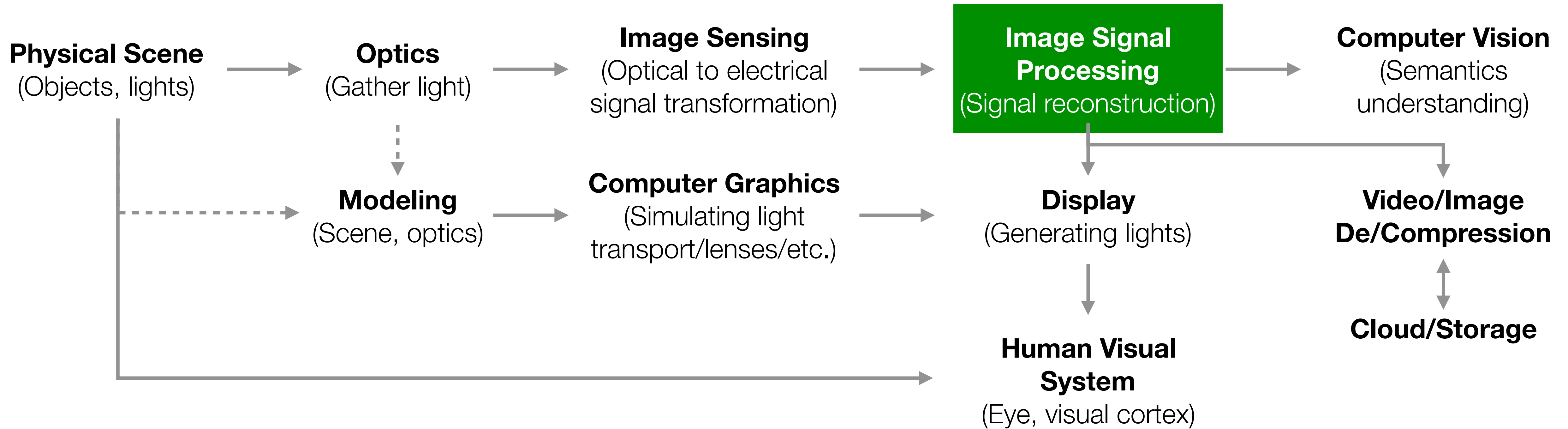
# Logistics

WA4 released. The last written assignment.

Will post PA2 soon.



# Where Are We



# The Roadmap

Theoretical Preliminaries

Human Visual Systems

**Color in Nature, Arts, Tech**  
(a.k.a., the birth, life, and death of light)

**Digital Camera Imaging**

Modeling and Rendering

Applications



Optics in Camera

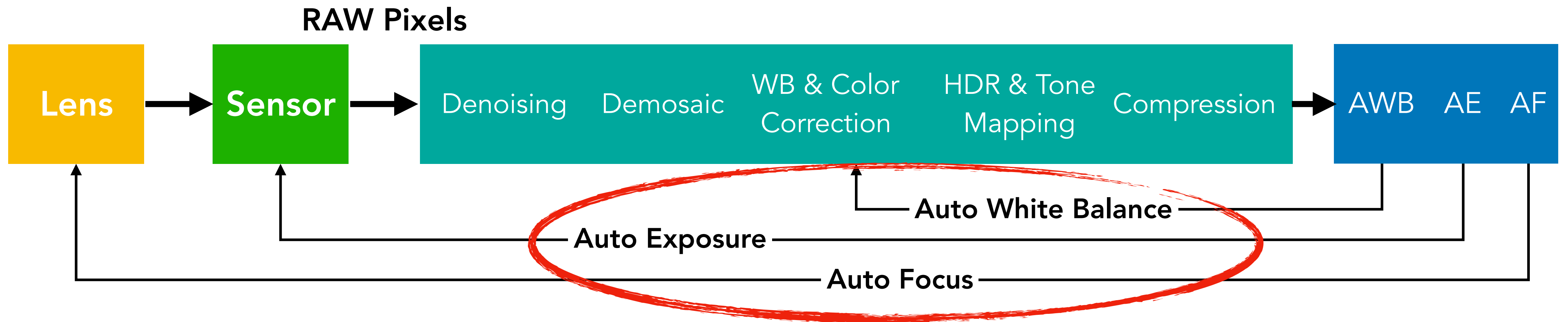
Image Sensor

**Image Signal Processing**

Image/Video Compression

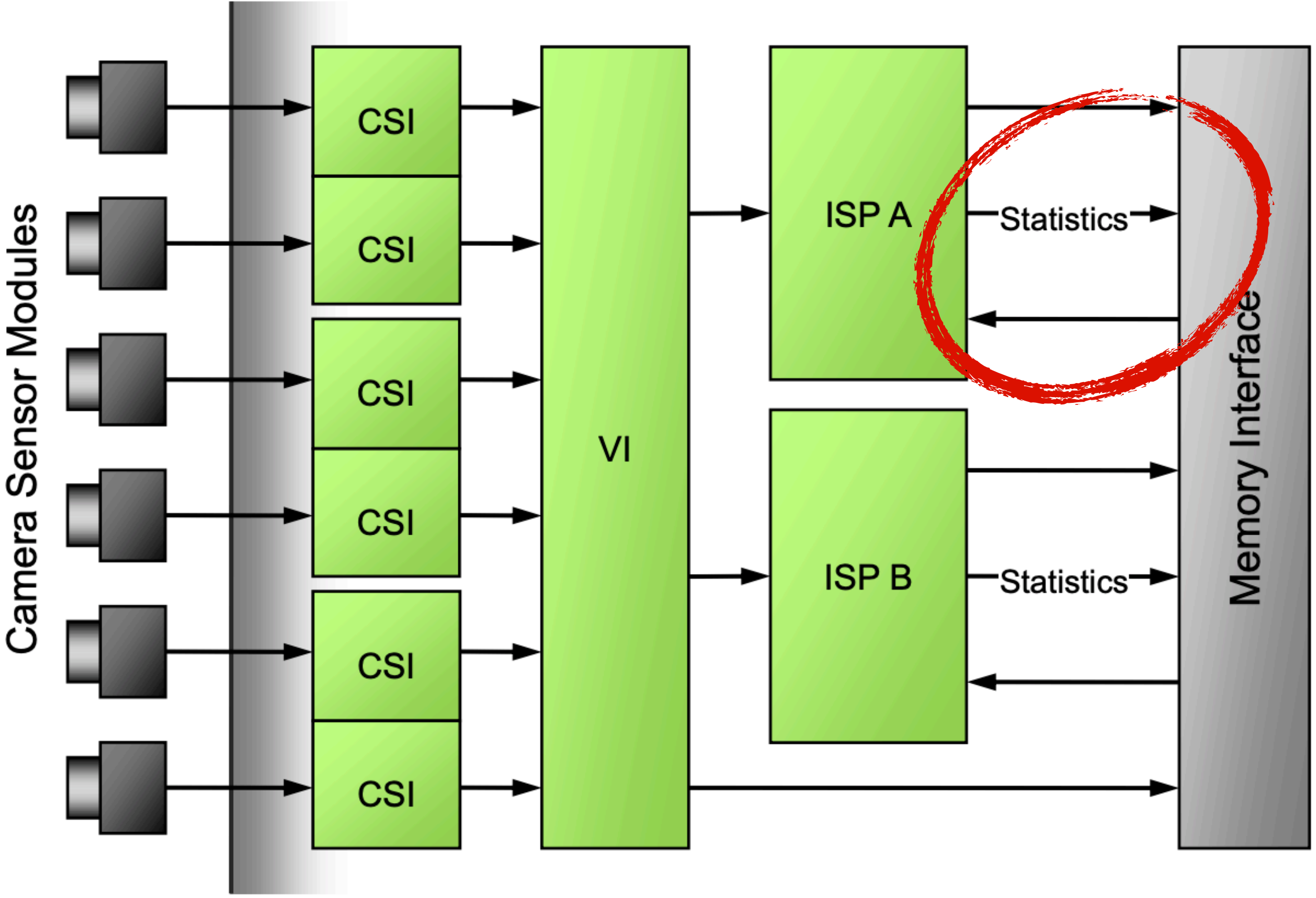
Immersive Content

# The Feedback Loops



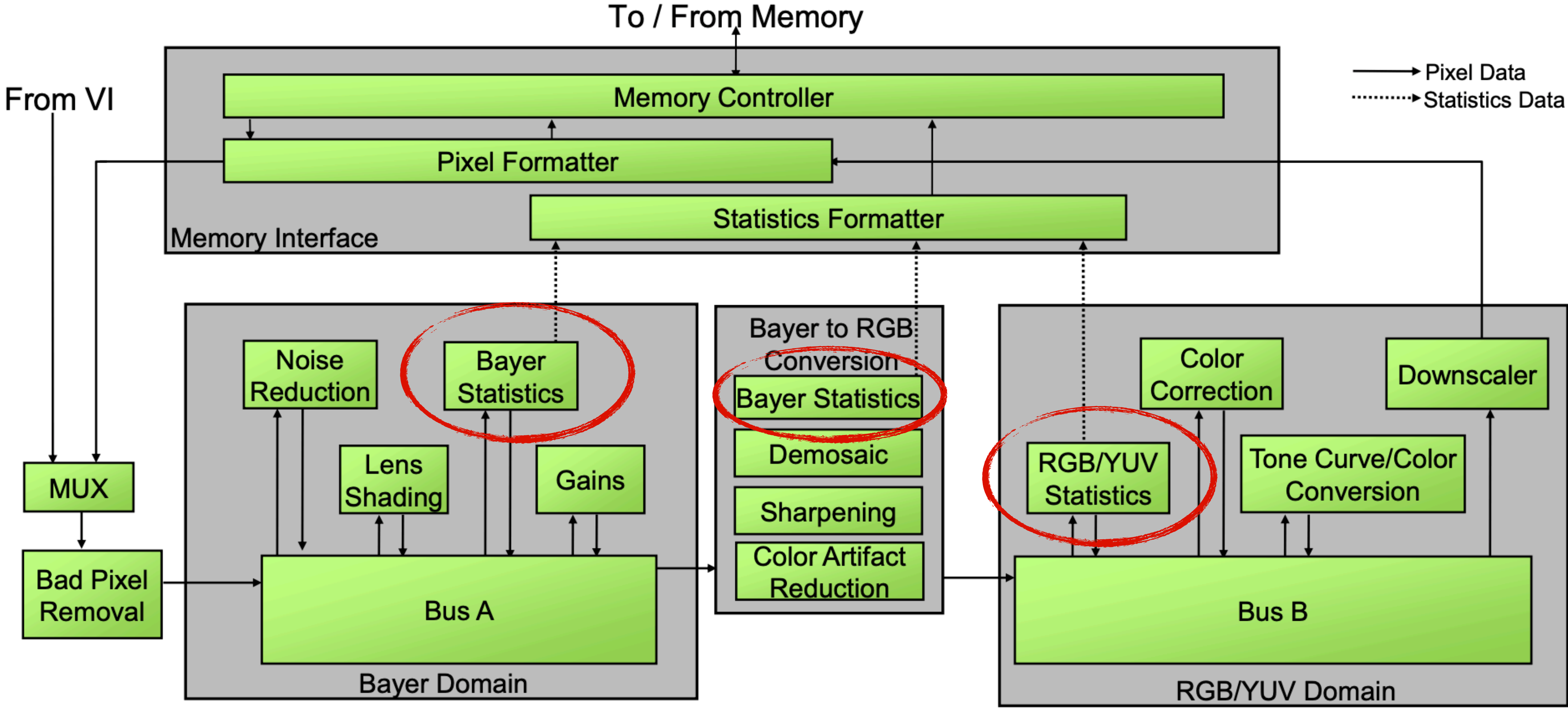
# Feedback

Camera subsystem in Nvidia TX1 SoC



<https://www.servethehome.com/nvidia-jetson-agx-xavier-module-for-robotics-development/nvidia-jetson-agx-xavier-soc/>

# Feedback





# What Are They?

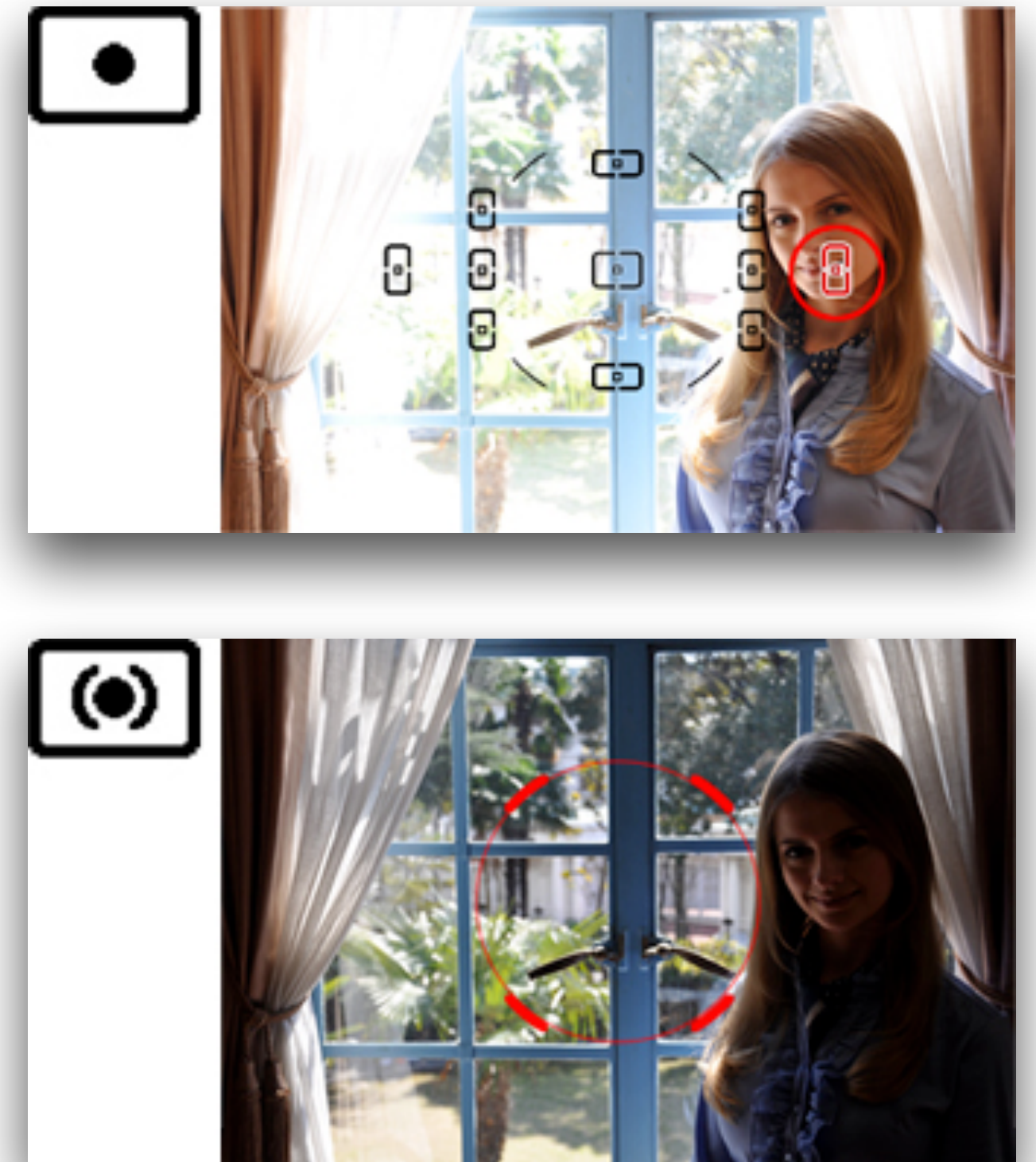
Auto White Balance



Auto Focus



Auto Exposure





# What Are They?

Word of caution: 3A are very **subjective!**

## Auto White Balance



## Auto Focus

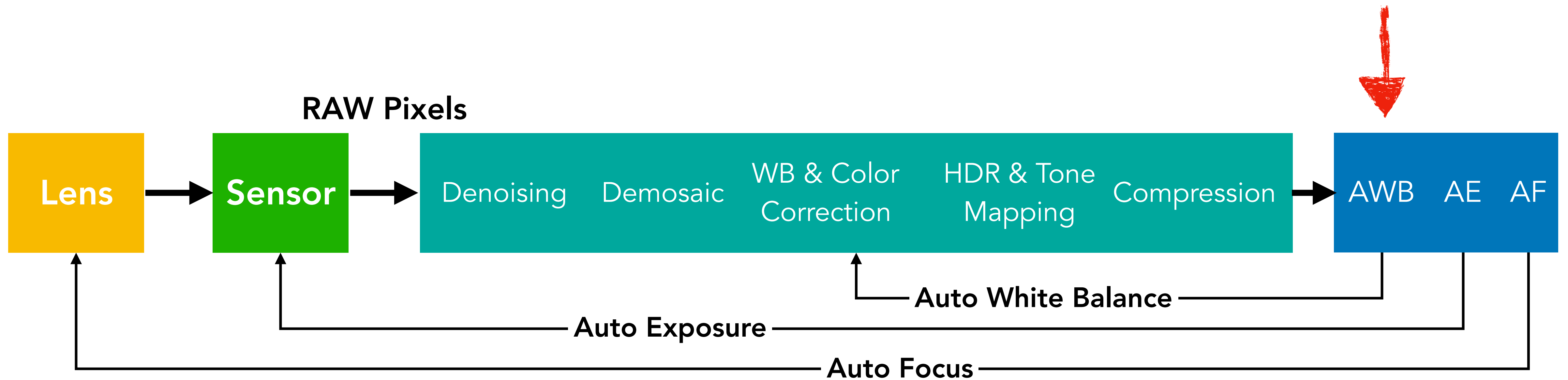


## Auto Exposure





# Auto White Balance







Does it look white?  
Should it look white?



It doesn't look white as we are looking at the photo now, but would look white to our eyes when we were sitting in front of it at the moment.

Does it look white?  
Should it look white?





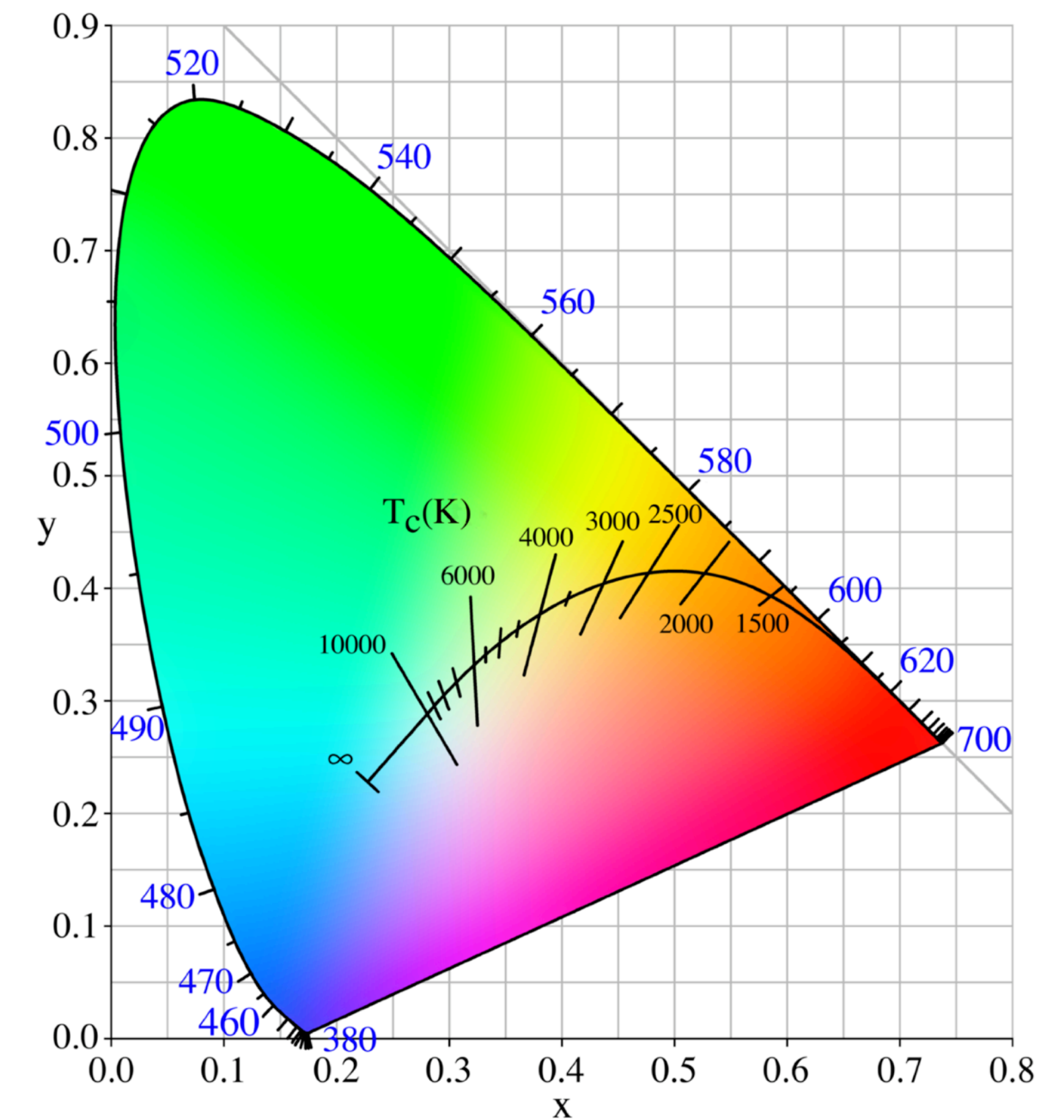
# "Color Constancy"

Clear blue poleward sky

Overcast daylight sky



Objective colors under different illuminants are very different, but our subjective color perception is not nearly that different.





# "Color Constancy"

The objective color of the scene. Color is reddish because the illuminant is ~tungsten.



The color you actually see when in the scene.



# Color Constancy and Chromatic Adaption

Human visual system adjusts to changes in illumination to preserve the relatively constant appearance of "white".





# Color Constancy and Chromatic Adaption

Human visual system adjusts to changes in illumination to preserve the relatively constant appearance of “**white**”.

Informally, a white paper is seen as white under many light sources. But different light sources actually have very different colors.





# Color Constancy and Chromatic Adaption

Human visual system adjusts to changes in illumination to preserve the relatively constant appearance of “**white**”.

Informally, a white paper is seen as white under many light sources. But different light sources actually have very different colors.

Appearances of other colors adapt too, but don't adapt fully.

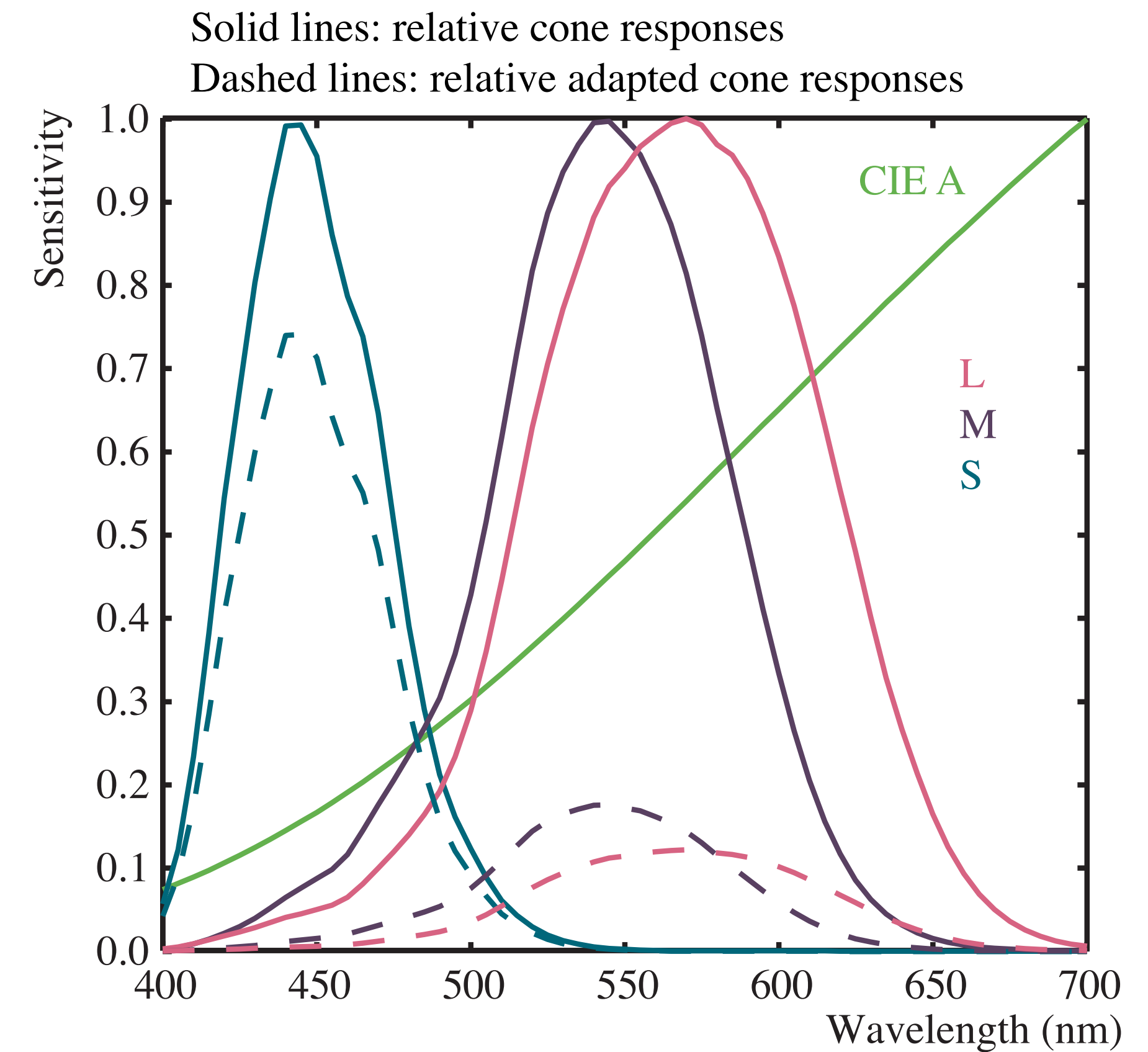


# Von Kries Chromatic Adaption Model

Hypothesis: The **cone responses** adapt to the illuminant. Cones become more/less sensitive depending on the illuminant.

- Each cone adapts independently.
- Each cone's "adaption factor" is inversely proportional to the cone's responses of the illuminant itself.

$$\begin{bmatrix} L_a \\ M_a \\ S_a \end{bmatrix} = \begin{bmatrix} \frac{1}{L_w}, 0, 0 \\ 0, \frac{1}{M_w}, 0 \\ 0, 0, \frac{1}{S_w} \end{bmatrix} \times \begin{bmatrix} L \\ M \\ S \end{bmatrix}, \text{ where } \begin{bmatrix} L_w \\ M_w \\ S_w \end{bmatrix} = \begin{bmatrix} \int \Phi(\lambda)L(\lambda)d\lambda \\ \int \Phi(\lambda)M(\lambda)d\lambda \\ \int \Phi(\lambda)S(\lambda)d\lambda \end{bmatrix}$$






# Von Kries Chromatic Adaption Model

Neutral points all look as the same color under different illuminants.

- Neural points: points whose spectral reflectance is 1 everywhere. Informally we call it the **white** point.
- The color of a neural point is the color of the illuminant **without** illumination.

$$\begin{bmatrix} \frac{1}{L_{w1}} & 0 & 0 \\ 0 & \frac{1}{M_{w1}} & 0 \\ 0 & 0 & \frac{1}{S_{w1}} \end{bmatrix} \times \begin{bmatrix} L_{w1} \\ M_{w1} \\ S_{w1} \end{bmatrix} = \begin{bmatrix} \frac{1}{L_{w2}} & 0 & 0 \\ 0 & \frac{1}{M_{w2}} & 0 \\ 0 & 0 & \frac{1}{S_{w2}} \end{bmatrix} \times \begin{bmatrix} L_{w2} \\ M_{w2} \\ S_{w2} \end{bmatrix}$$



**Unadapted** responses of **neutral point** under the first illuminant      **Unadapted** responses of **neutral point** under the second illuminant

# White Balance, a.k.a., Camera Chromatic Adaptation

Human evolved over millions of years to learn to adapt to illuminants. But cameras don't have this luxury — camera SSFs are fixed once fabricated.

**Goal:** adapt colors so that the resultant image *appears* as the photographers "remember" based upon their adaptation state.

Photo without AWB

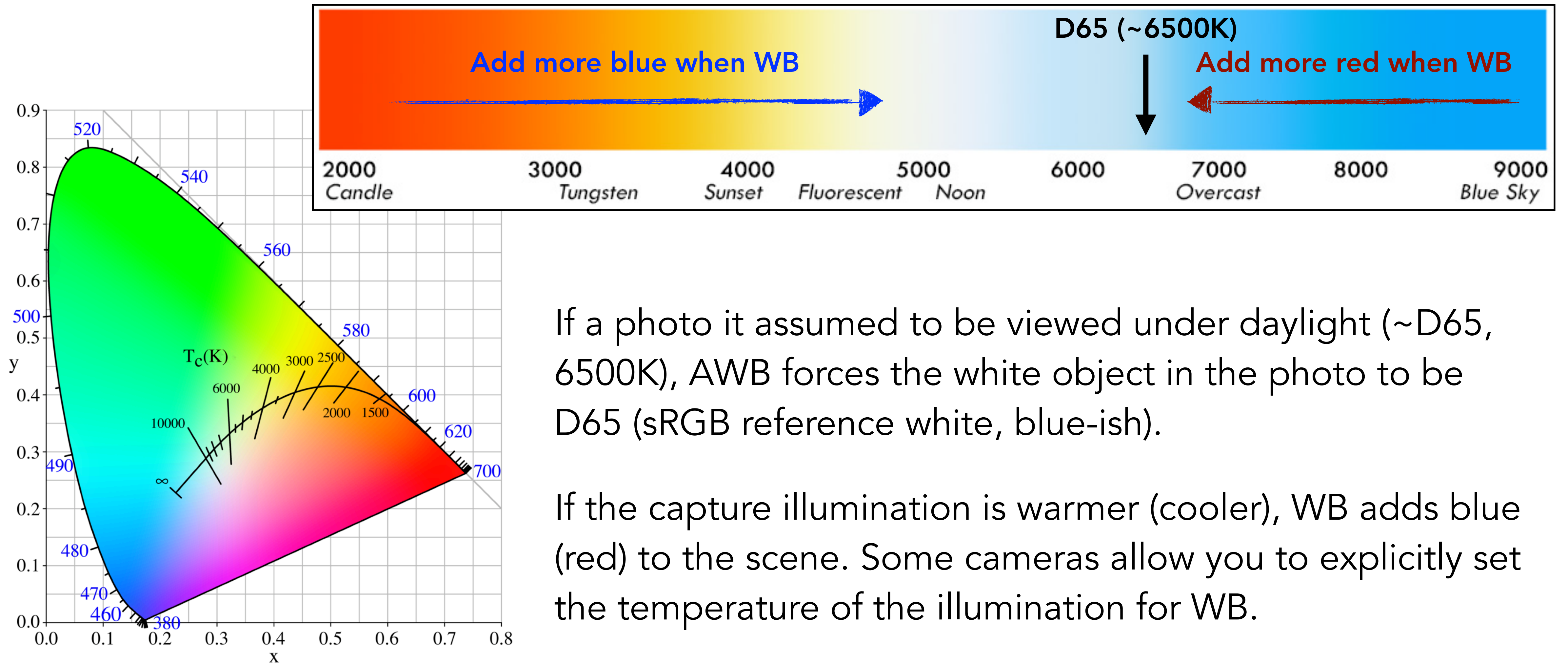


Photo with AWB





# White Balance and Color Temperature

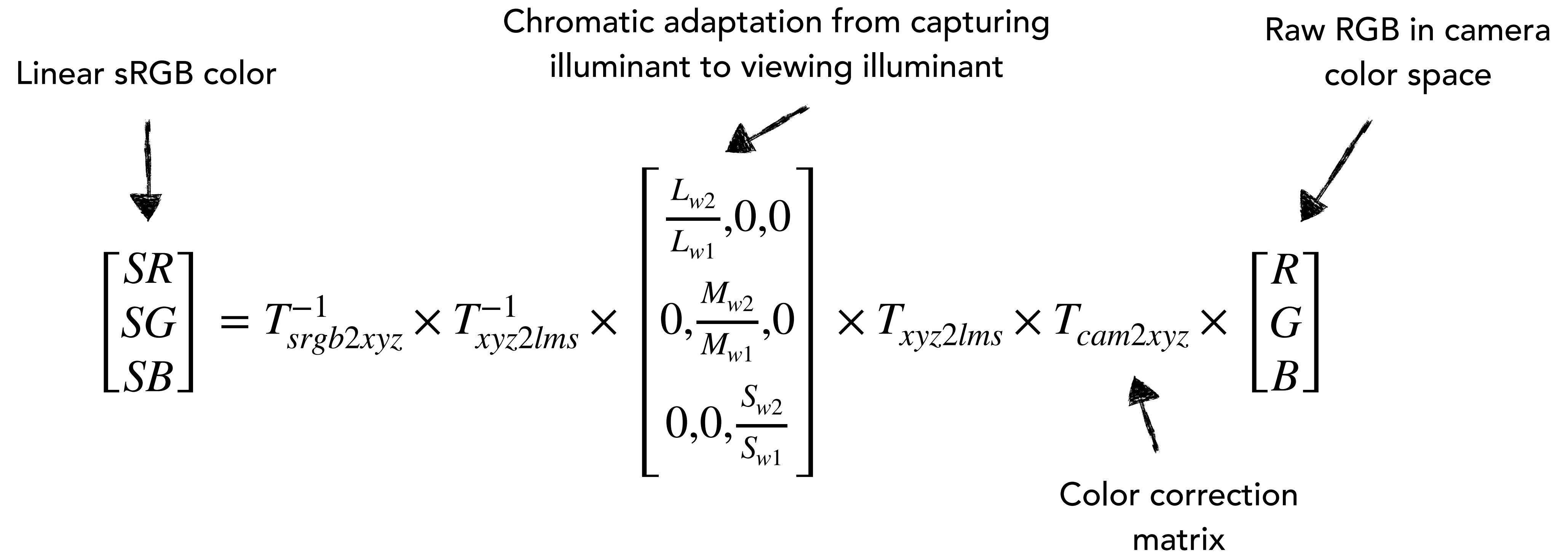


If a photo is assumed to be viewed under daylight (~D65, 6500K), AWB forces the white object in the photo to be D65 (sRGB reference white, blue-ish).

If the capture illumination is warmer (cooler), WB adds blue (red) to the scene. Some cameras allow you to explicitly set the temperature of the illumination for WB.

# White Balance in Cameras

Read: <http://yuhaozhu.com/blog/chromatic-adaptation.html>



# Ideal vs. Practical White Balance

Ideal white balance is easy if we know:

- the illuminant of the capturing scene.
- the illuminant of the viewing scene.

If so, we can simply calculate the scaling factors of the LMS cone responses.

- Remember, RGB/XYZ is just one linear transformation away from LMS. So knowing how to scale LMS basically means we know how to scale RGB values too.

In reality: capturing illuminant is unknown.

The real work of auto white balance is to **estimate the capturing illuminant.**



# (Semi-)Auto White Balance

How to know where a neutral point is (i.e., which pixel *should* look white under the capturing illuminant)?

Pre-made reference white card

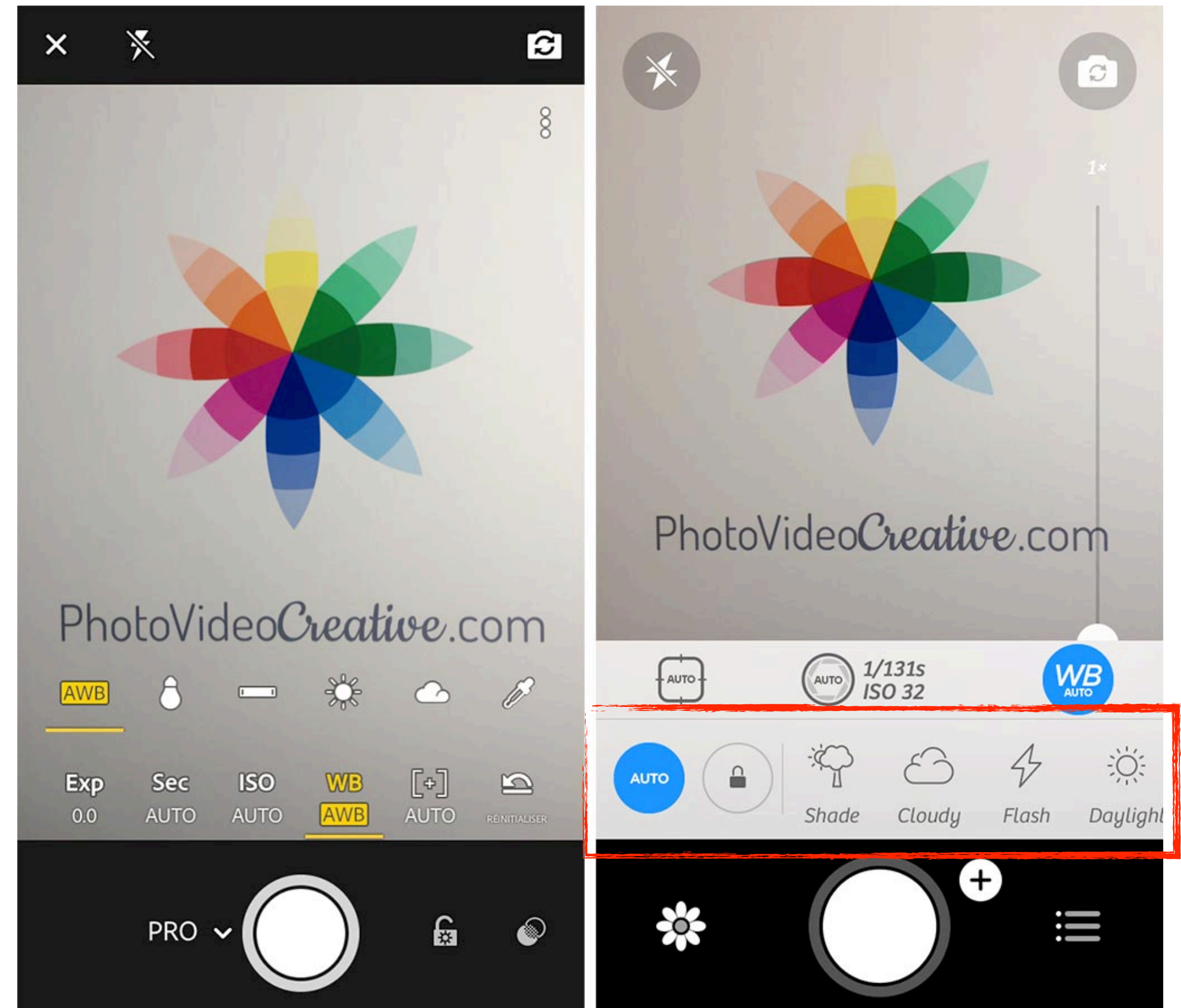
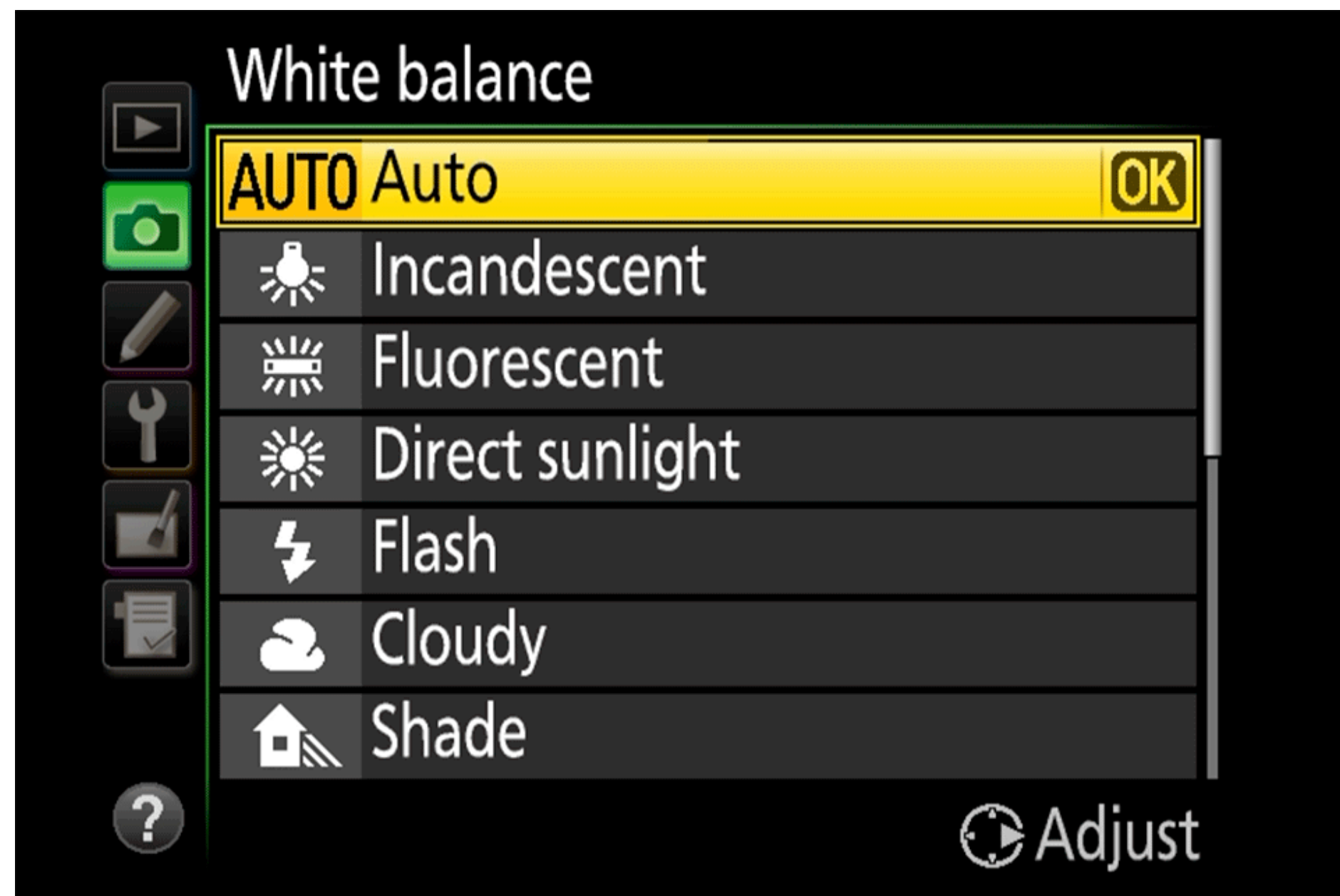
Can be used as a neutral point (lucky!)





# (Semi-)Auto White Balance

If a white point is unavailable, specify the capture illuminant and use a pre-calculated adaptation matrix, which is calculated offline from known illuminants.



<https://www.vortexmediastore.com/pages/warmcards-white-balance-system>

<https://photovideocreative.com/en/have-color-neutral-photo-with-smartphone-neither-too-bluish-nor-too-orange/>

<https://photographylife.com/what-is-white-balance-in-photography>

# Illuminant Estimation by AWB

Many heuristics:



# Illuminant Estimation by AWB

Many heuristics:

- Assume that the average color of all pixels in an image is gray (e.g., [128, 128, 128] in sRGB) and scale all pixels accordingly. Fails when the scene is colorful (close-up flower).

# Illuminant Estimation by AWB

Many heuristics:

- Assume that the average color of all pixels in an image is gray (e.g., [128, 128, 128] in sRGB) and scale all pixels accordingly. Fails when the scene is colorful (close-up flower).
- Assume that the brightest pixel is white. Fails quite often (night scene with traffic lights).

# Illuminant Estimation by AWB

Many heuristics:

- Assume that the average color of all pixels in an image is gray (e.g., [128, 128, 128] in sRGB) and scale all pixels accordingly. Fails when the scene is colorful (close-up flower).
- Assume that the brightest pixel is white. Fails quite often (night scene with traffic lights).
- Other techniques try to guess the illumination: bright image is probably outdoor; dim images are probably indoor, etc.

# Illuminant Estimation by AWB

Many heuristics:

- Assume that the average color of all pixels in an image is gray (e.g., [128, 128, 128] in sRGB) and scale all pixels accordingly. Fails when the scene is colorful (close-up flower).
- Assume that the brightest pixel is white. Fails quite often (night scene with traffic lights).
- Other techniques try to guess the illumination: bright image is probably outdoor; dim images are probably indoor, etc.

Train deep learning models to predict capturing illuminant (example).



# Aside: Illuminant Estimation in Augmented Reality





# Aside: Illuminant Estimation in Augmented Reality





# White Balance and Color Correction

In theory, color correction comes first and then white balance, but usually cameras perform WB before CC

- through an equivalent transformation; read [this article](#) by Rowlands for why.

$$\begin{bmatrix} SR \\ SG \\ SB \end{bmatrix} = T_{srgb2xyz}^{-1} \times T_{xyz2lms}^{-1} \times \begin{bmatrix} \frac{L_{w2}}{L_{w1}}, 0, 0 \\ 0, \frac{M_{w2}}{M_{w1}}, 0 \\ 0, 0, \frac{S_{w2}}{S_{w1}} \end{bmatrix} \times T_{xyz2lms} \times T_{cam2xyz} \times \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

$$\begin{bmatrix} SR \\ SG \\ SB \end{bmatrix} = T_{cc} \times T_{wb} \times \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

This "white balance" matrix normalizes the raw RGB of capturing illuminant to [1, 1, 1], which will be mapped to [1, 1, 1] in linear sRGB by  $T_{cc}$ .  $T_{wb}$  technically doesn't perform white balance.

# Red Without Any Red Pixel



The cyan tint tricks your brain to think that the illuminant of the scene is cyan. So your brain subtracts cyan from colors it perceives.

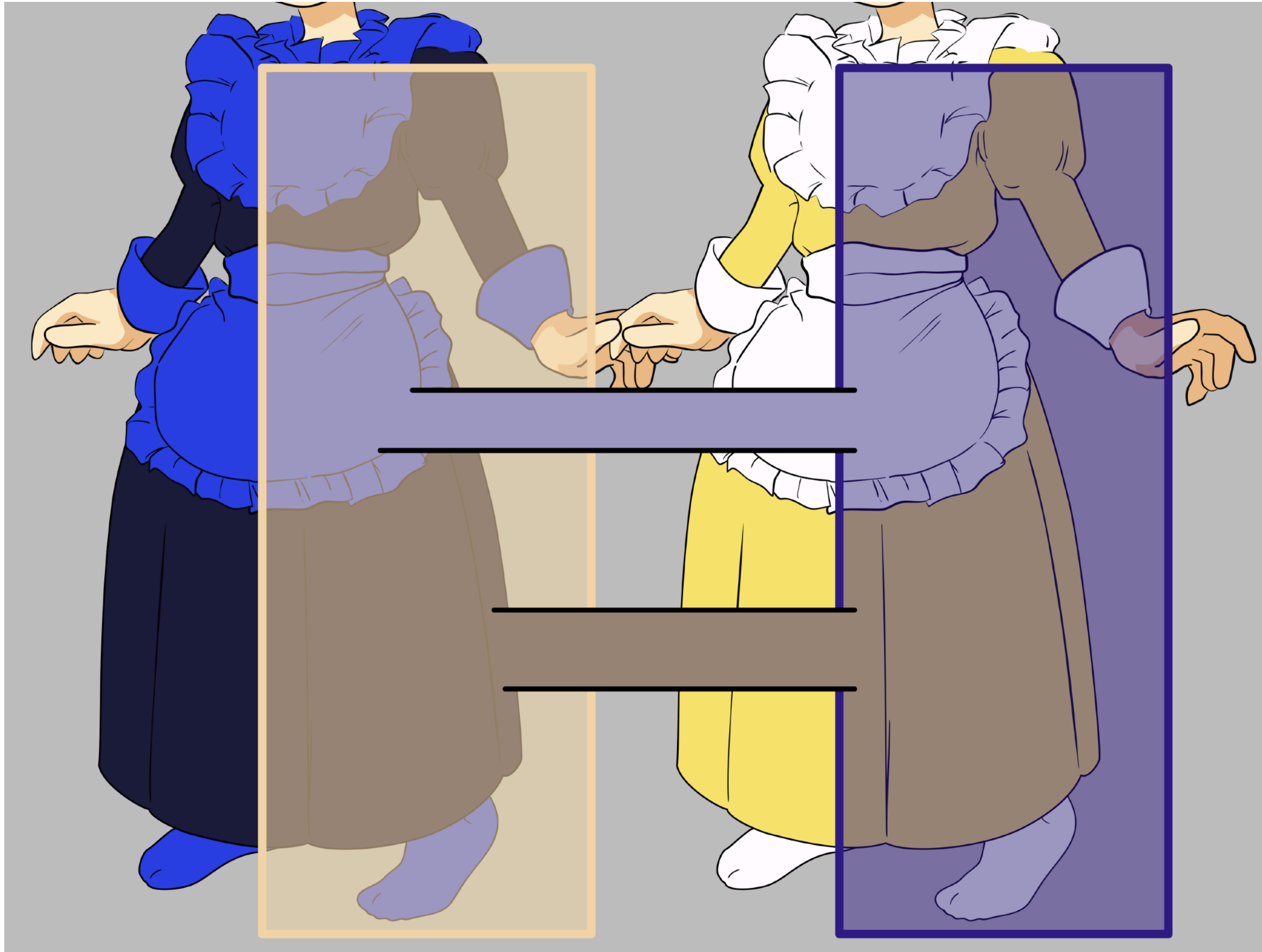
Subtracting cyan is like adding red (recall the RGB color cube), so the gray patches look red now.



# Black and Blue or White and Gold?



# Black and Blue or White and Gold?

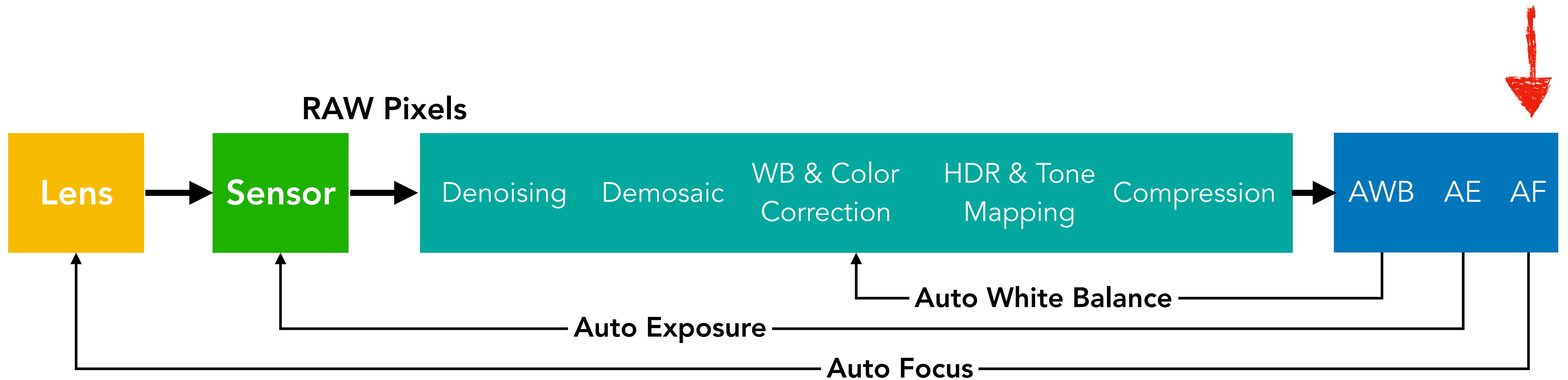


**Left:** if your visual system thinks the illuminant is yellow, it will subtract yellow (add blue) from the colors.

**Right:** if your visual system think the illuminant is blue, it will subtract blue (add yellow) from the colors.

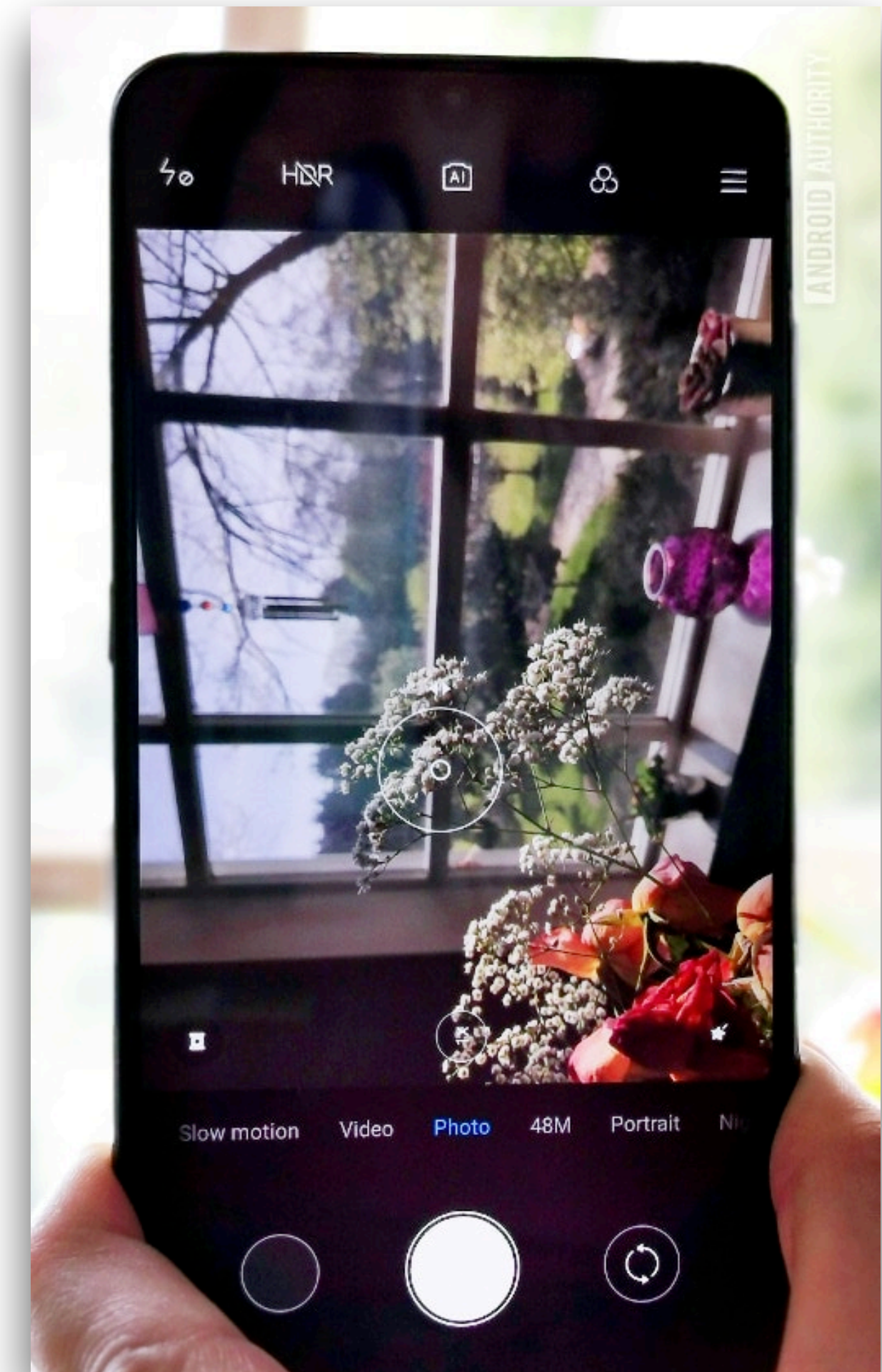
Why different visual systems think differently is anyone's guess.

# Auto Focus





# Auto Focus

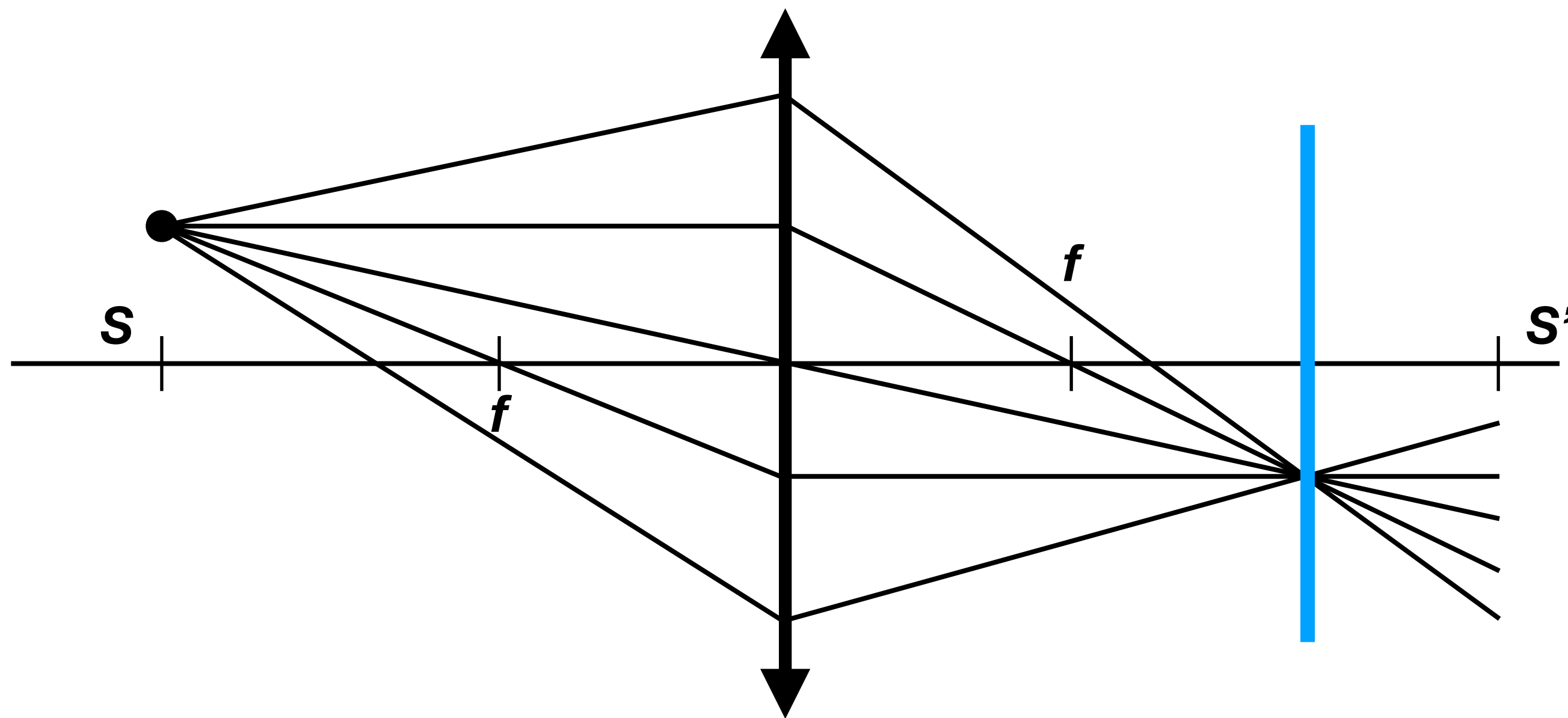




# Recall: Circle of Confusion

If CoC is greater than a threshold, the point appears blur.

- Multiple pixels get some rays from that point
- Blur from defocus; c.f. motion blur

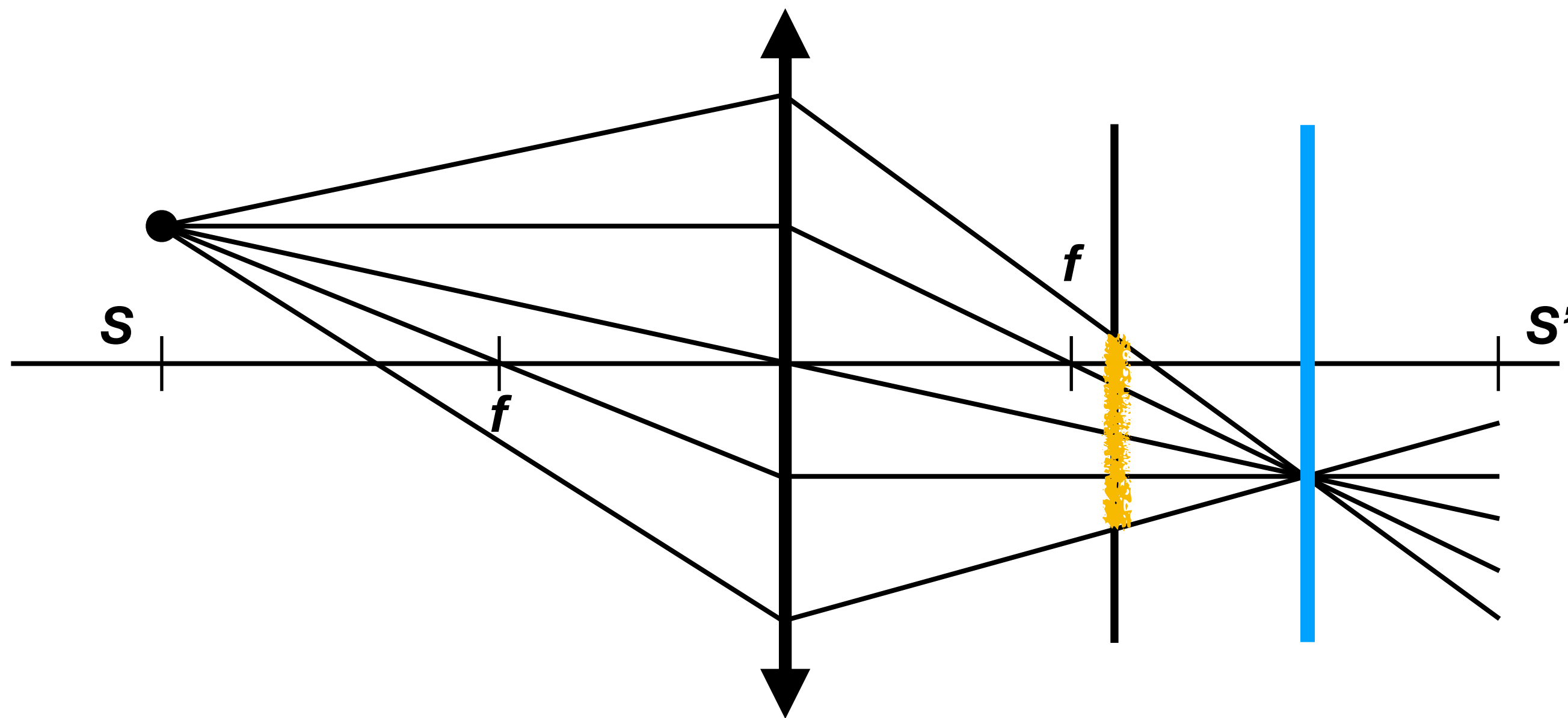




# Recall: Circle of Confusion

If CoC is greater than a threshold, the point appears blur.

- Multiple pixels get some rays from that point
- Blur from defocus; c.f. motion blur

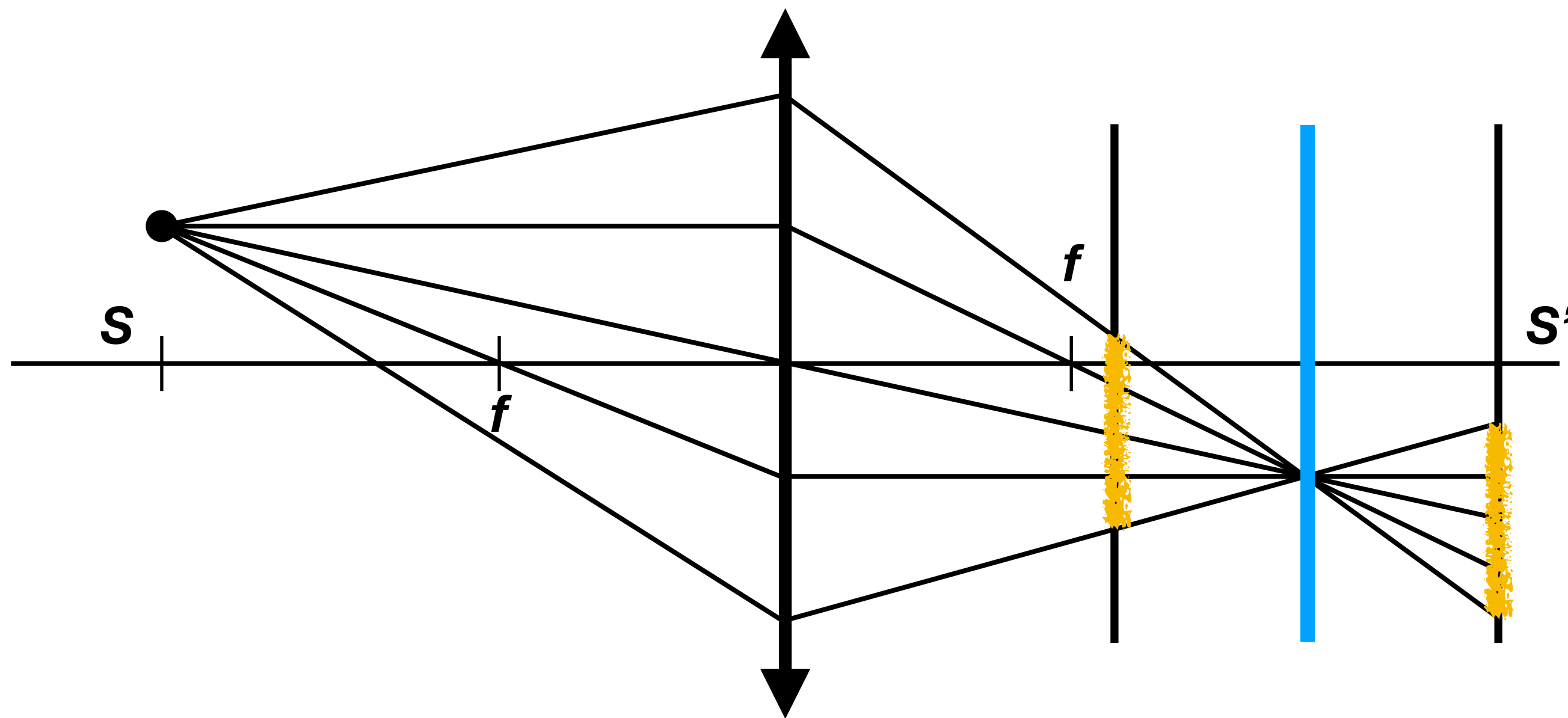




# Recall: Circle of Confusion

If CoC is greater than a threshold, the point appears blur.

- Multiple pixels get some rays from that point
- Blur from defocus; c.f. motion blur

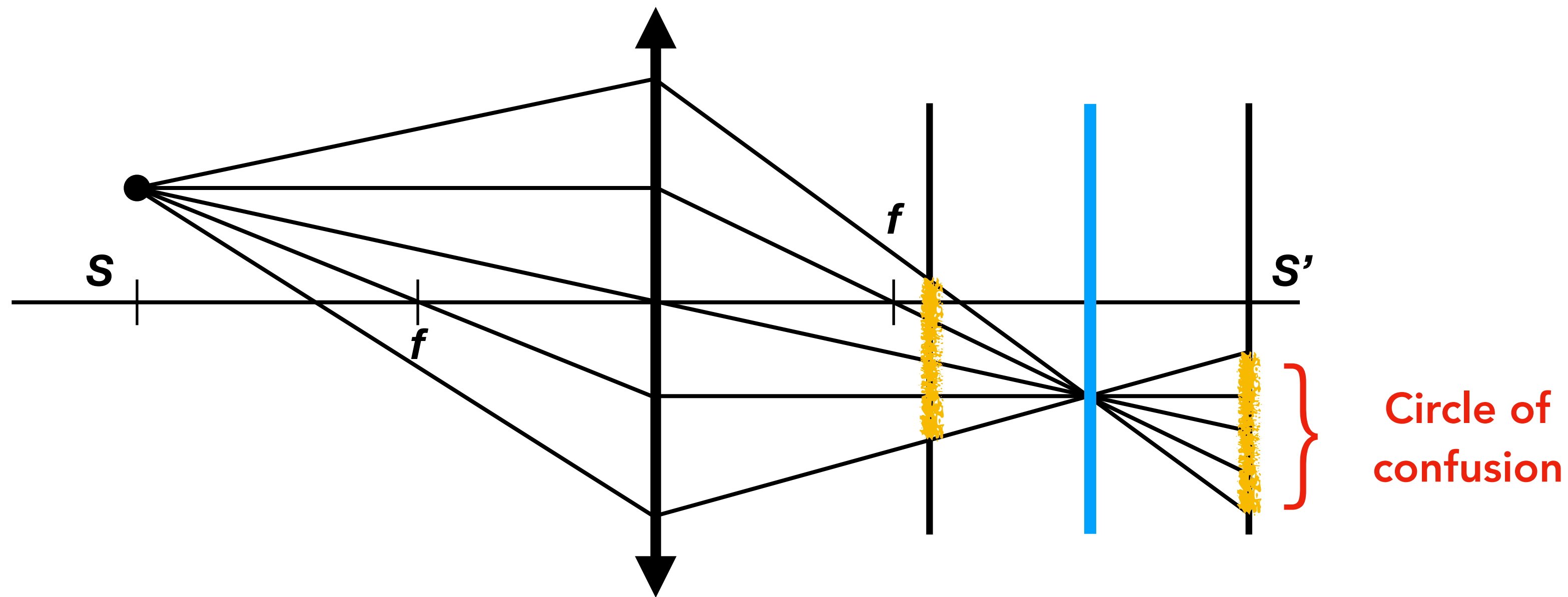




# Recall: Circle of Confusion

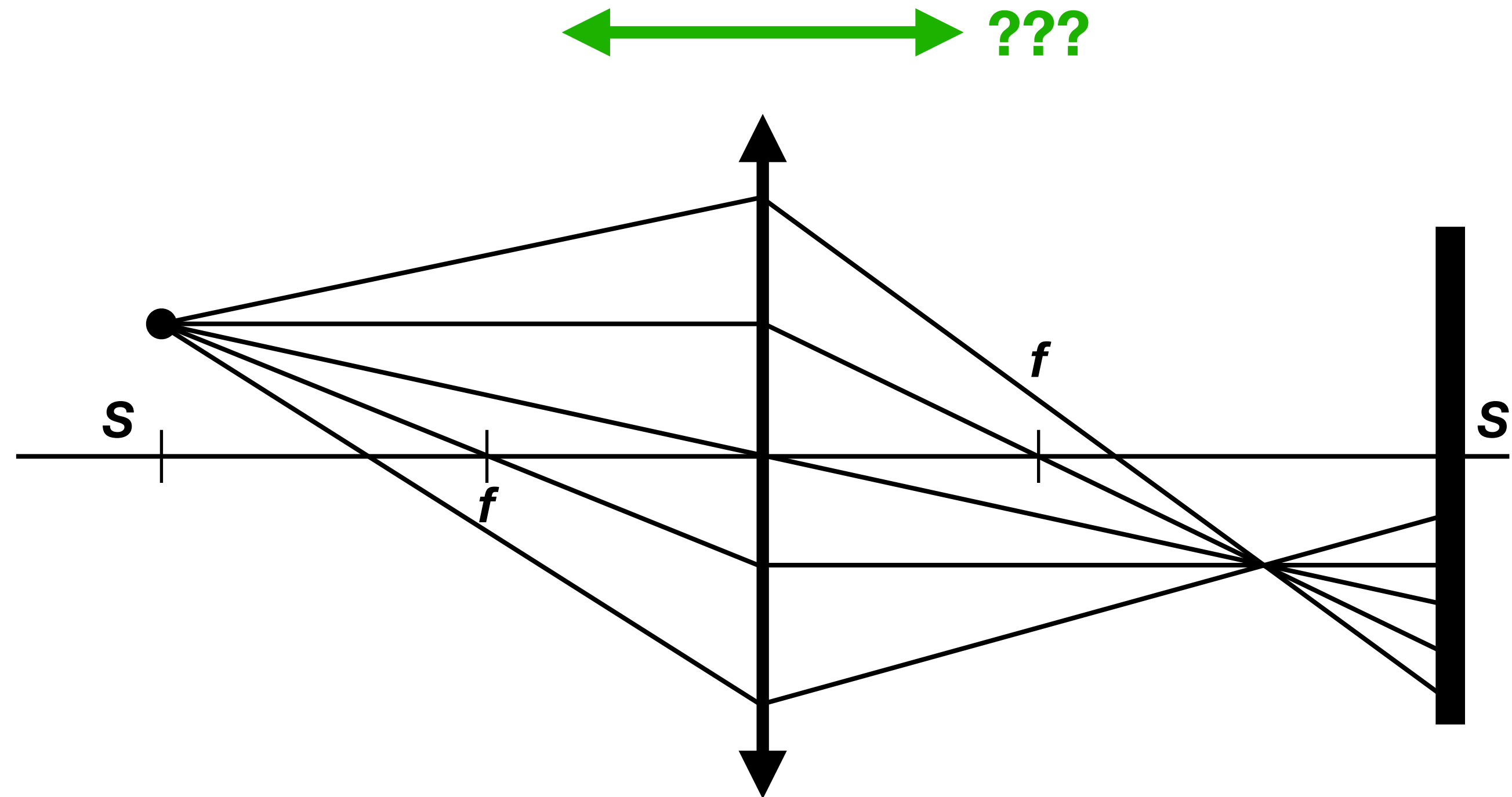
If CoC is greater than a threshold, the point appears blur.

- Multiple pixels get some rays from that point
- Blur from defocus; c.f. motion blur



# Auto Focus

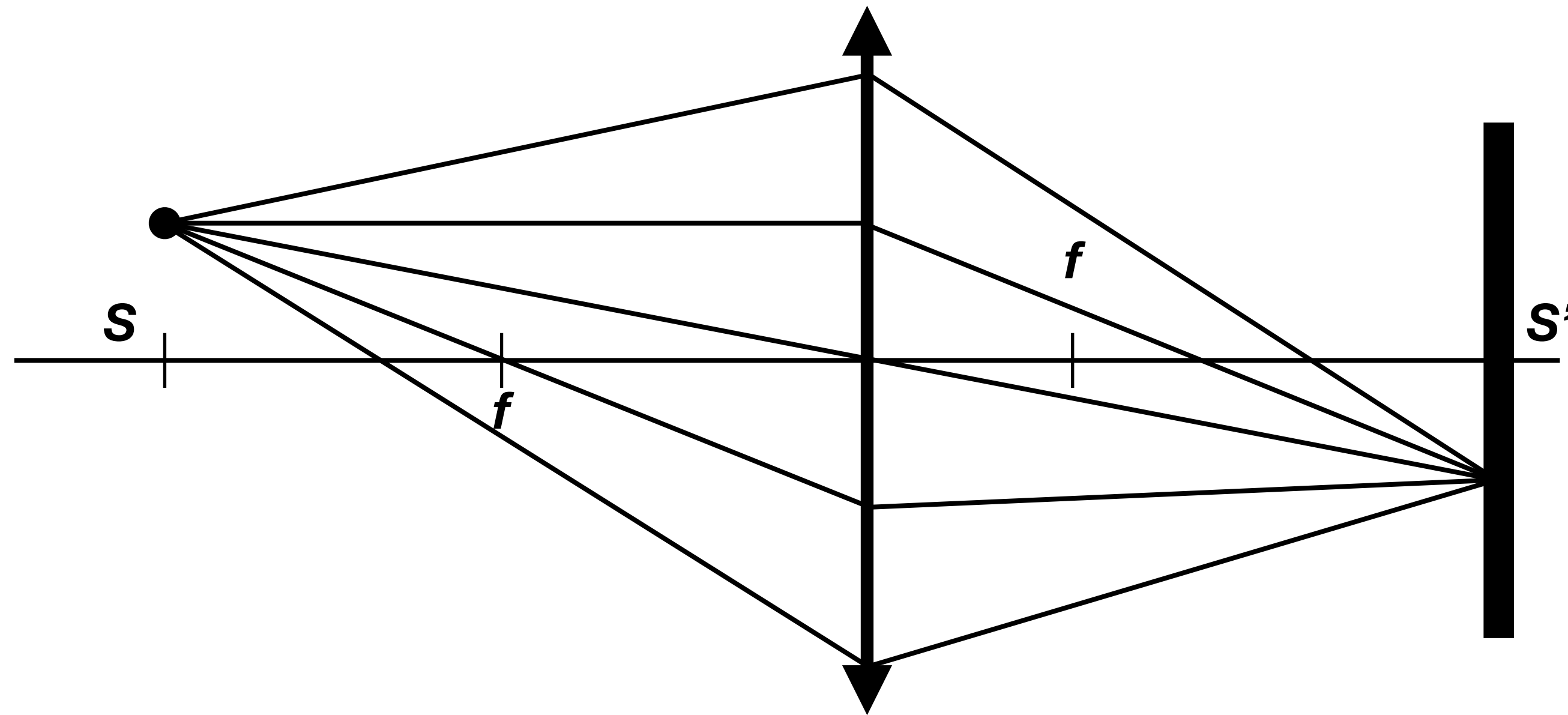
AF works by moving lenses while keeping the sensor fixed.





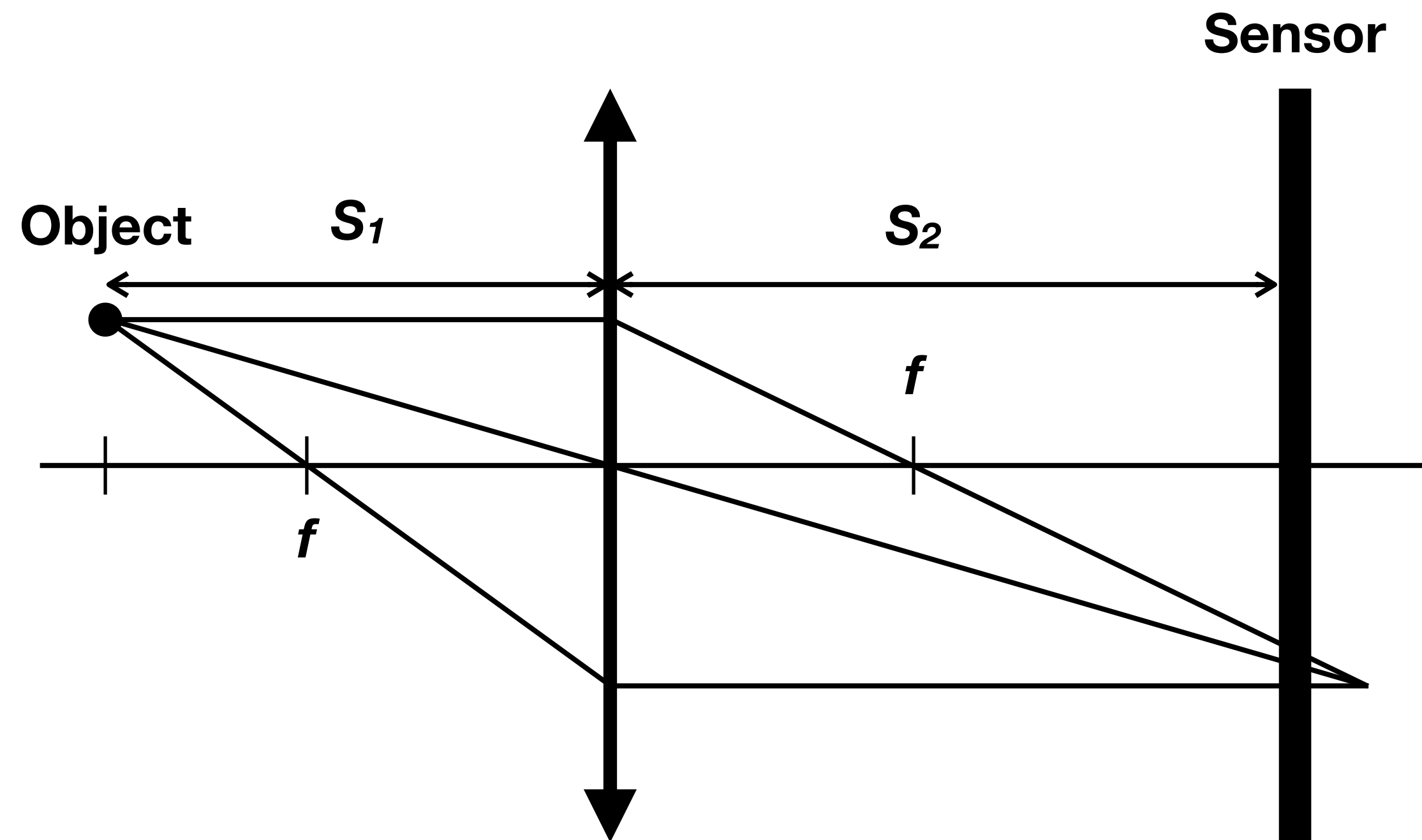
# Auto Focus

AF works by moving lenses while keeping the sensor fixed.



# The General Idea

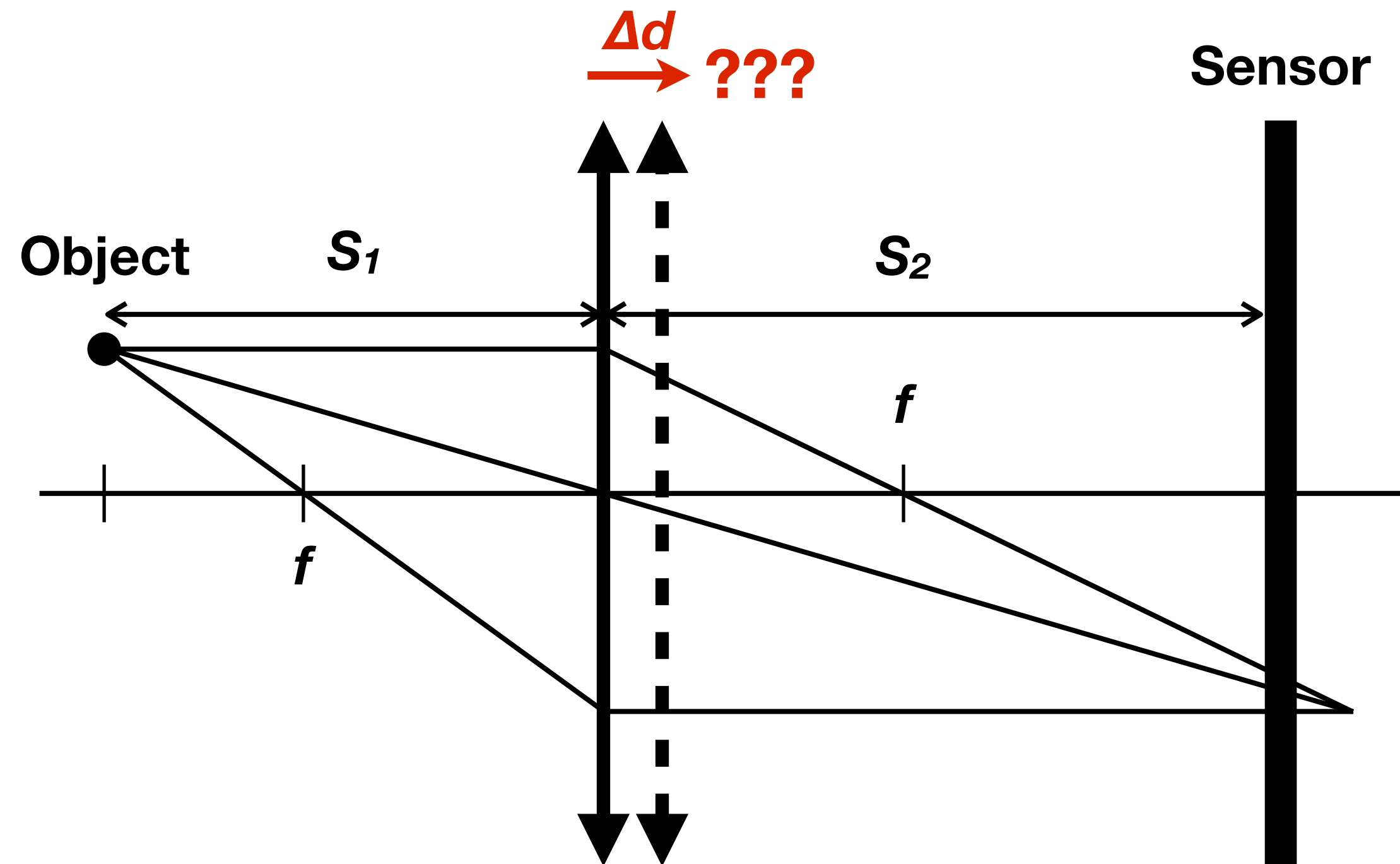
AF is inherently about **depth estimation**. From depth we can use the Gauss lens equation to determine the correct lens position.





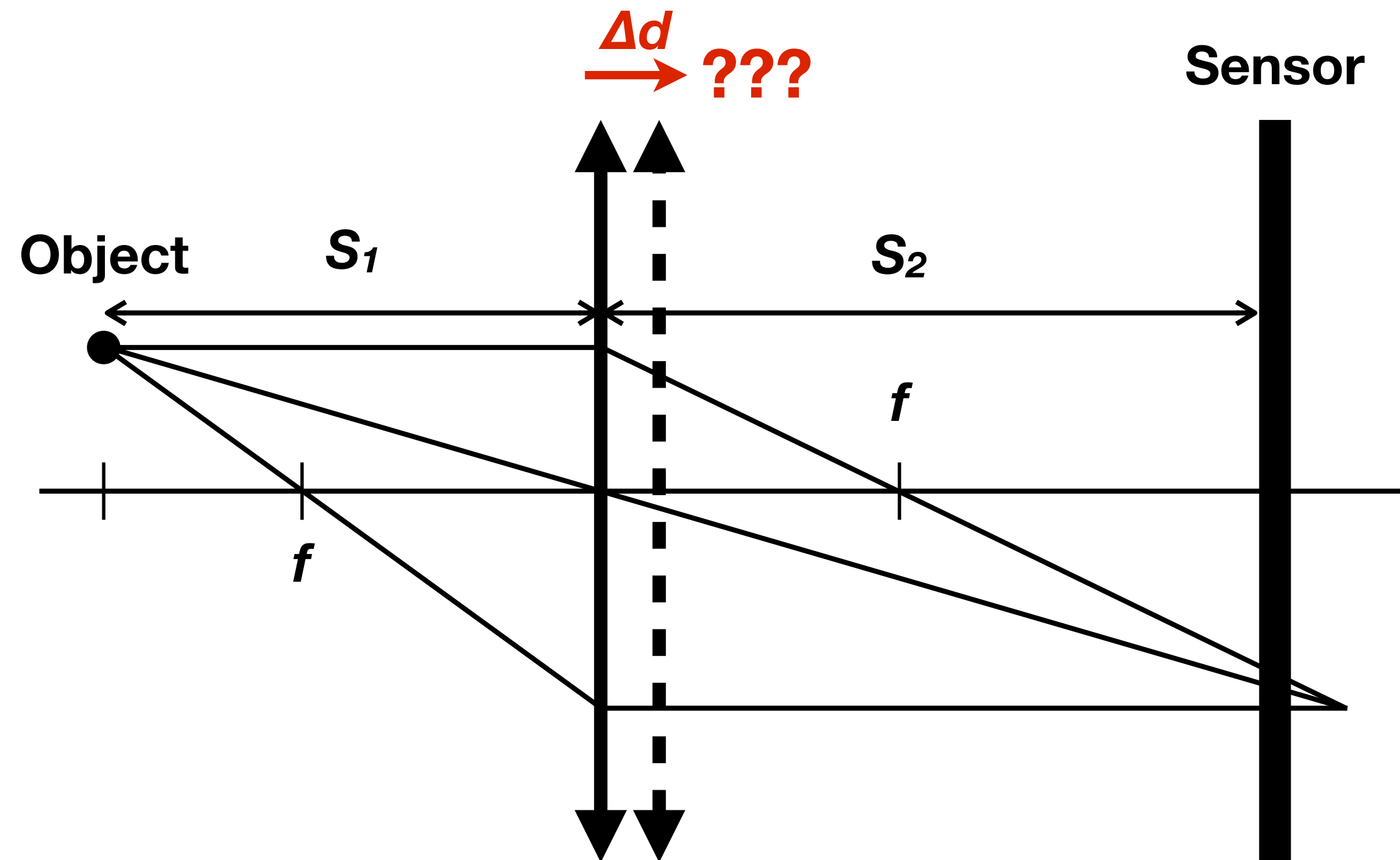
# The General Idea

AF is inherently about **depth estimation**. From depth we can use the Gauss lens equation to determine the correct lens position.



# The General Idea

AF is inherently about **depth estimation**. From depth we can use the Gauss lens equation to determine the correct lens position.

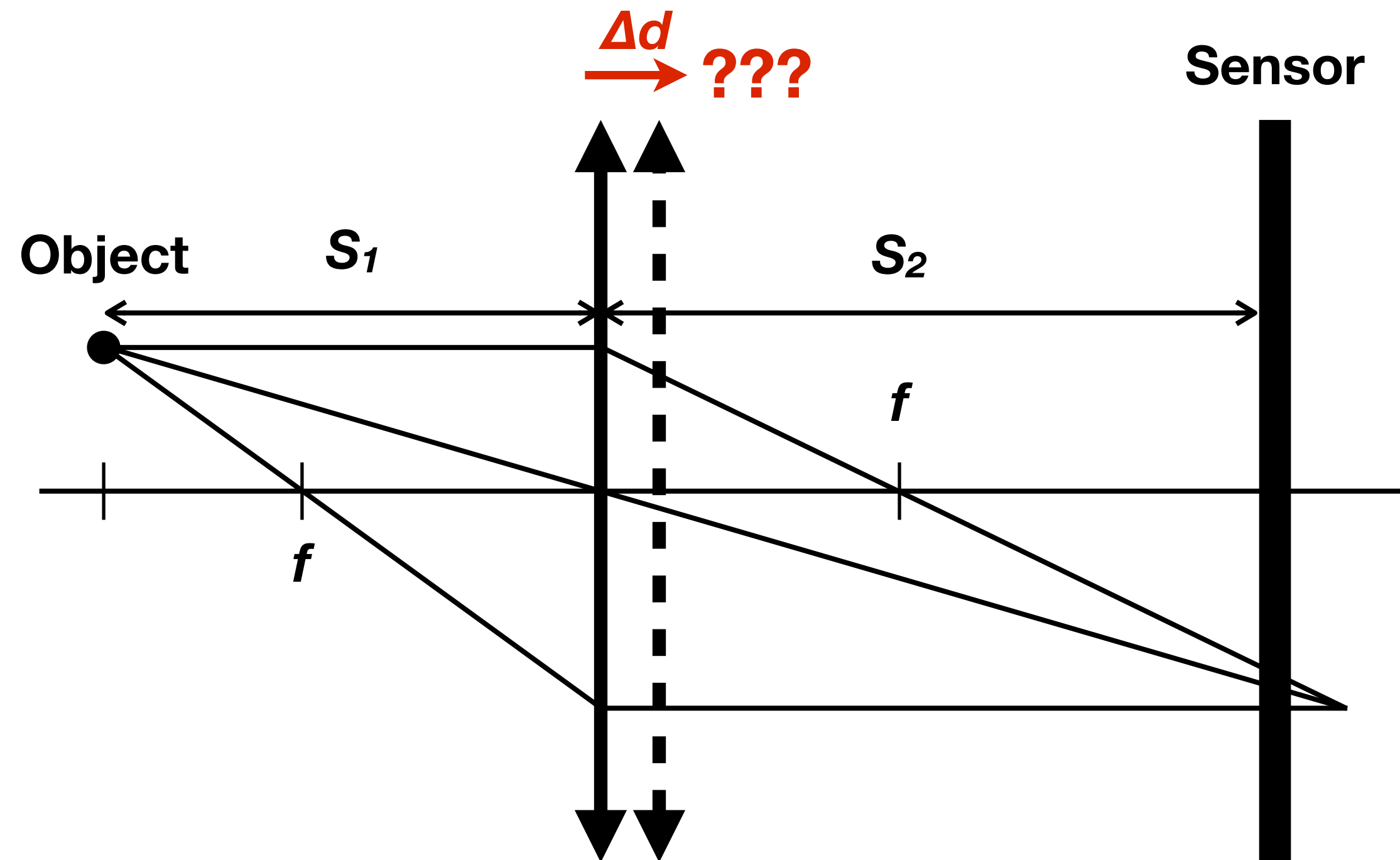


$$\frac{1}{S_1 + \Delta d} + \frac{1}{S_2 - \Delta d} = \frac{1}{f}$$



# The General Idea

AF is inherently about **depth estimation**. From depth we can use the Gauss lens equation to determine the correct lens position.

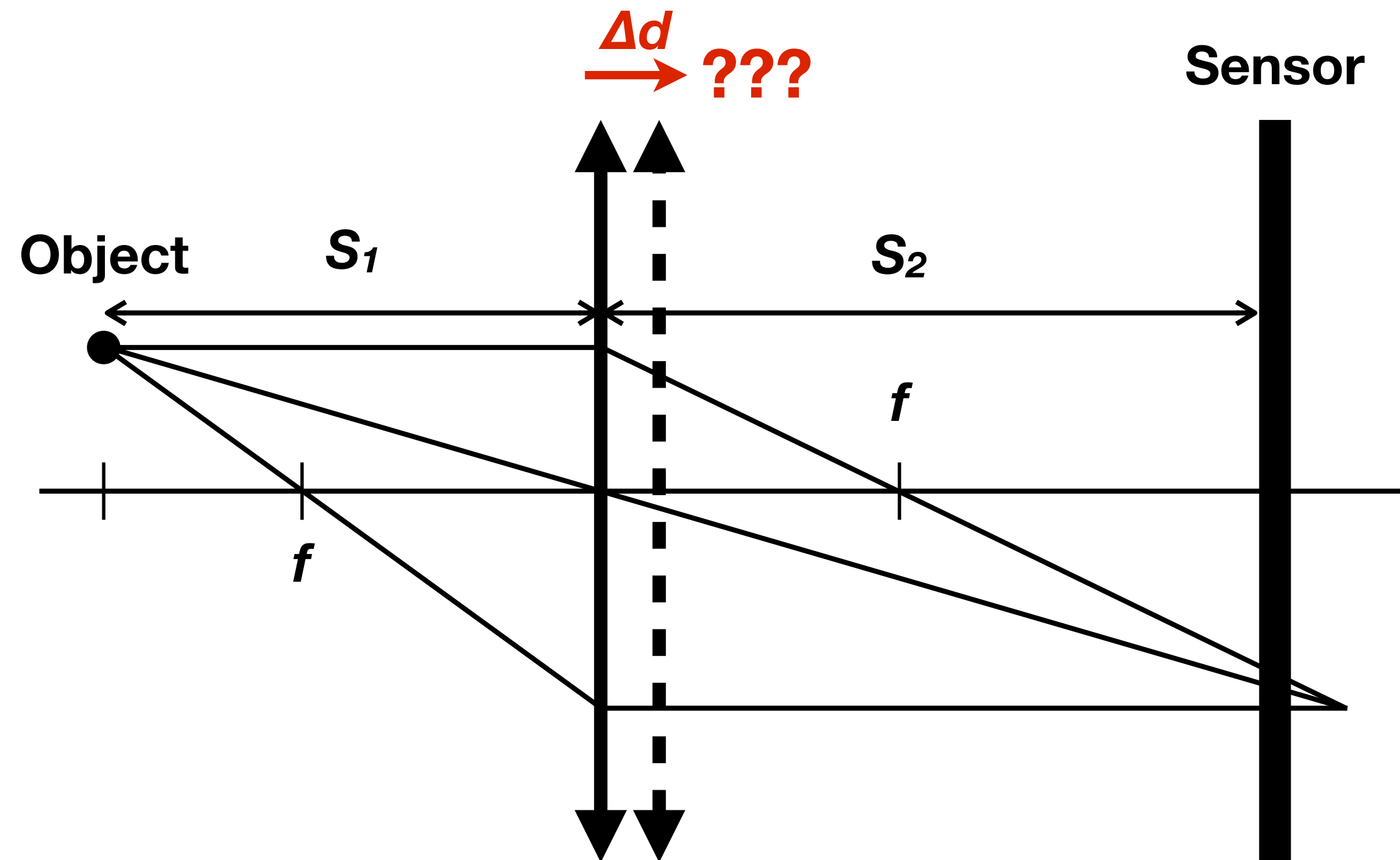


$$\frac{1}{S_1 + \Delta d} + \frac{1}{S_2 - \Delta d} = \frac{1}{f}$$

Known from  
current position

# The General Idea

AF is inherently about **depth estimation**. From depth we can use the Gauss lens equation to determine the correct lens position.



$$\frac{1}{S_1 + \Delta d} + \frac{1}{S_2 - \Delta d} = \frac{1}{f}$$

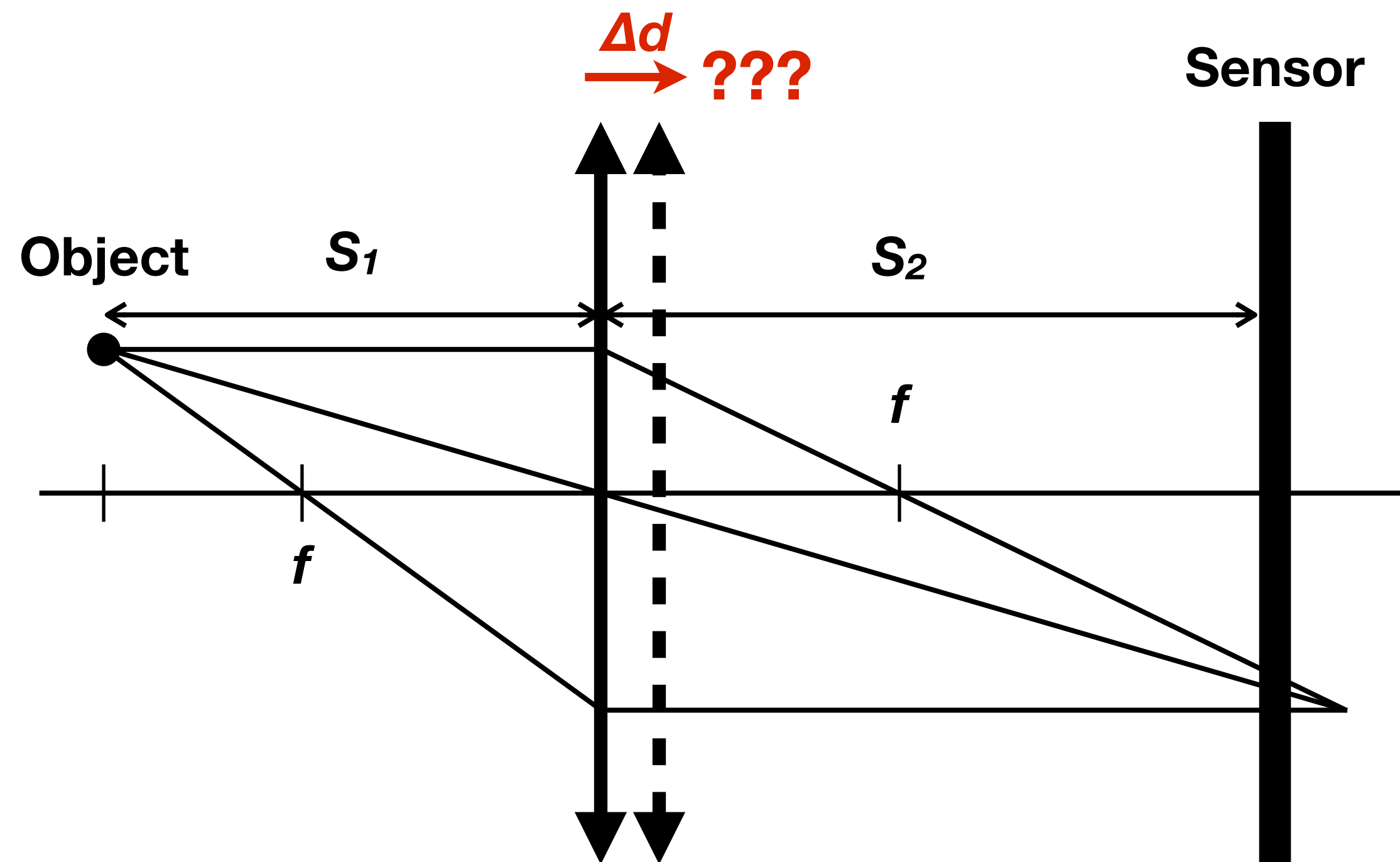
Known from current position

Constant



# The General Idea

AF is inherently about **depth estimation**. From depth we can use the Gauss lens equation to determine the correct lens position.



$$\frac{1}{S_1 + \Delta d} + \frac{1}{S_2 - \Delta d} = \frac{1}{f}$$

Annotations:  $S_1$  is circled in red with a red arrow pointing to it and the text "???",  $S_2$  is circled in green with a green arrow pointing to it and the text "Known from current position", and  $f$  is circled in green with a green arrow pointing to it and the text "Constant".

# Active vs. Passive Auto Focus

**Active** AF: camera emits some sort of radiation, whose reflections off of the object are captured by the camera to calculate depth, from which lens adjustment is calculated.

- IR light (structured light), Sonar, LiDAR

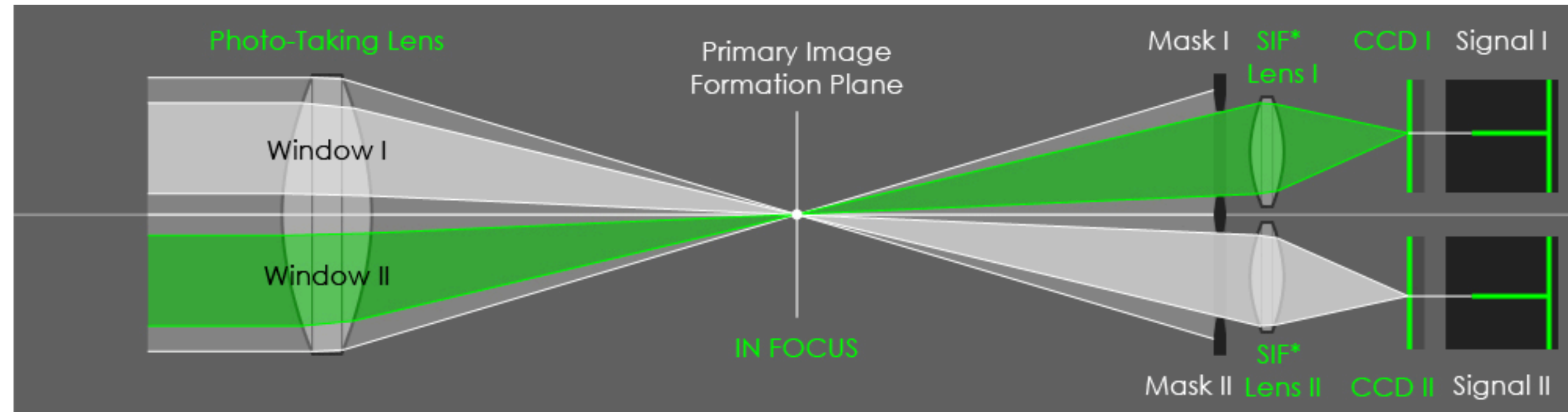
**Passive** AF: nothing is emitted from the camera, which "*passively*" processes the light it receives to estimate depth and move the lens.

- Phase detection AF
- Contrast detection AF



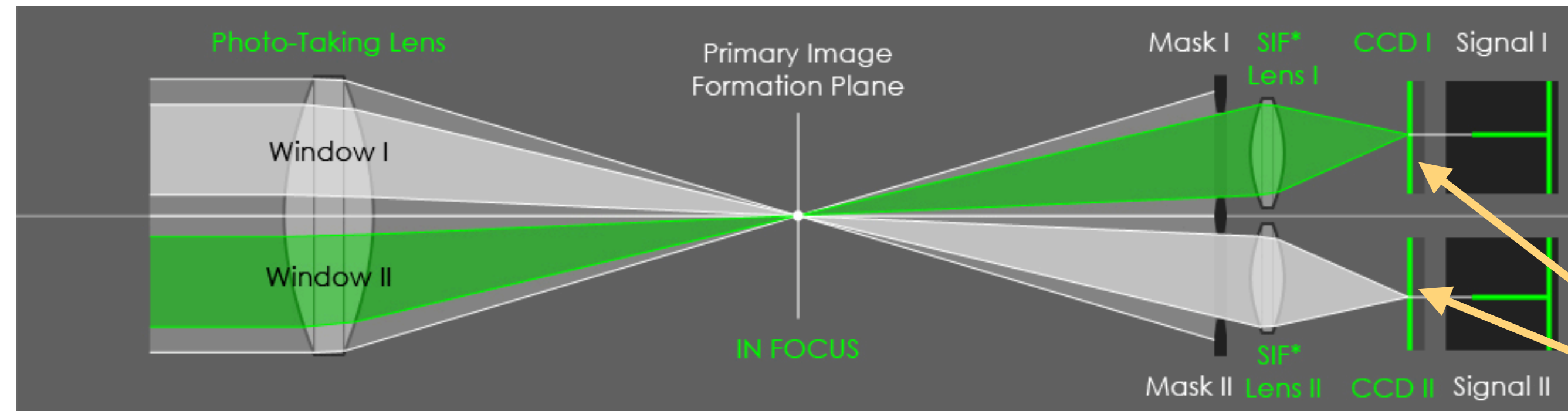
# Phase Detection Auto Focus (PDAF)

Assuming a point source in scene



# Phase Detection Auto Focus (PDAF)

Assuming a point source in scene

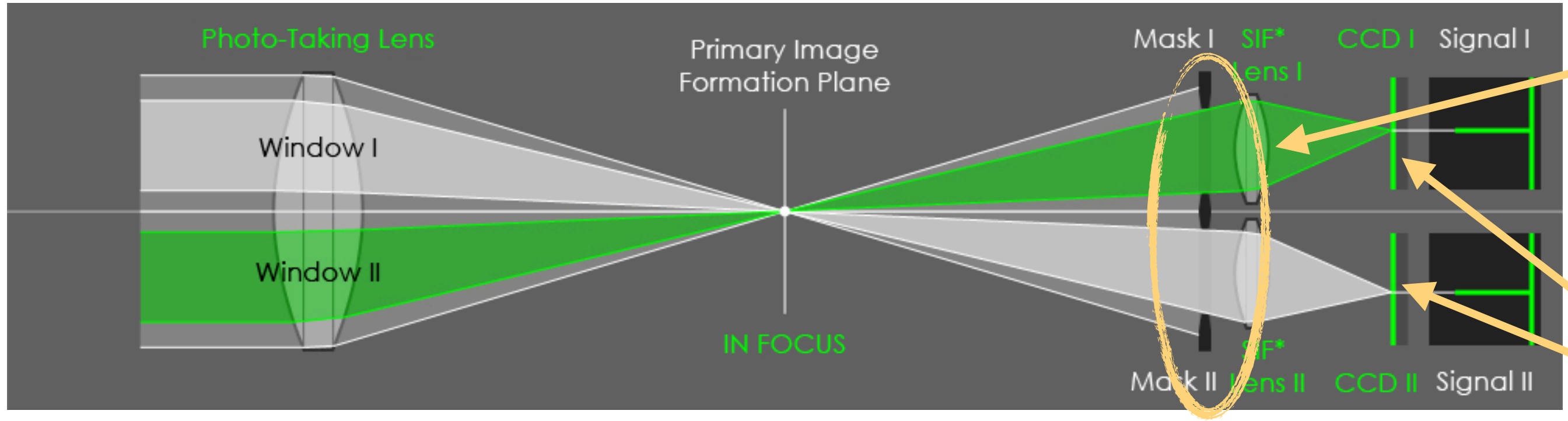


The AF lenses and AF sensors are fixed. They are placed in such a way that if the incident light is in-focus with by main lens, it will also be in-focus on the AF sensors.



# Phase Detection Auto Focus (PDAF)

Assuming a point source in scene

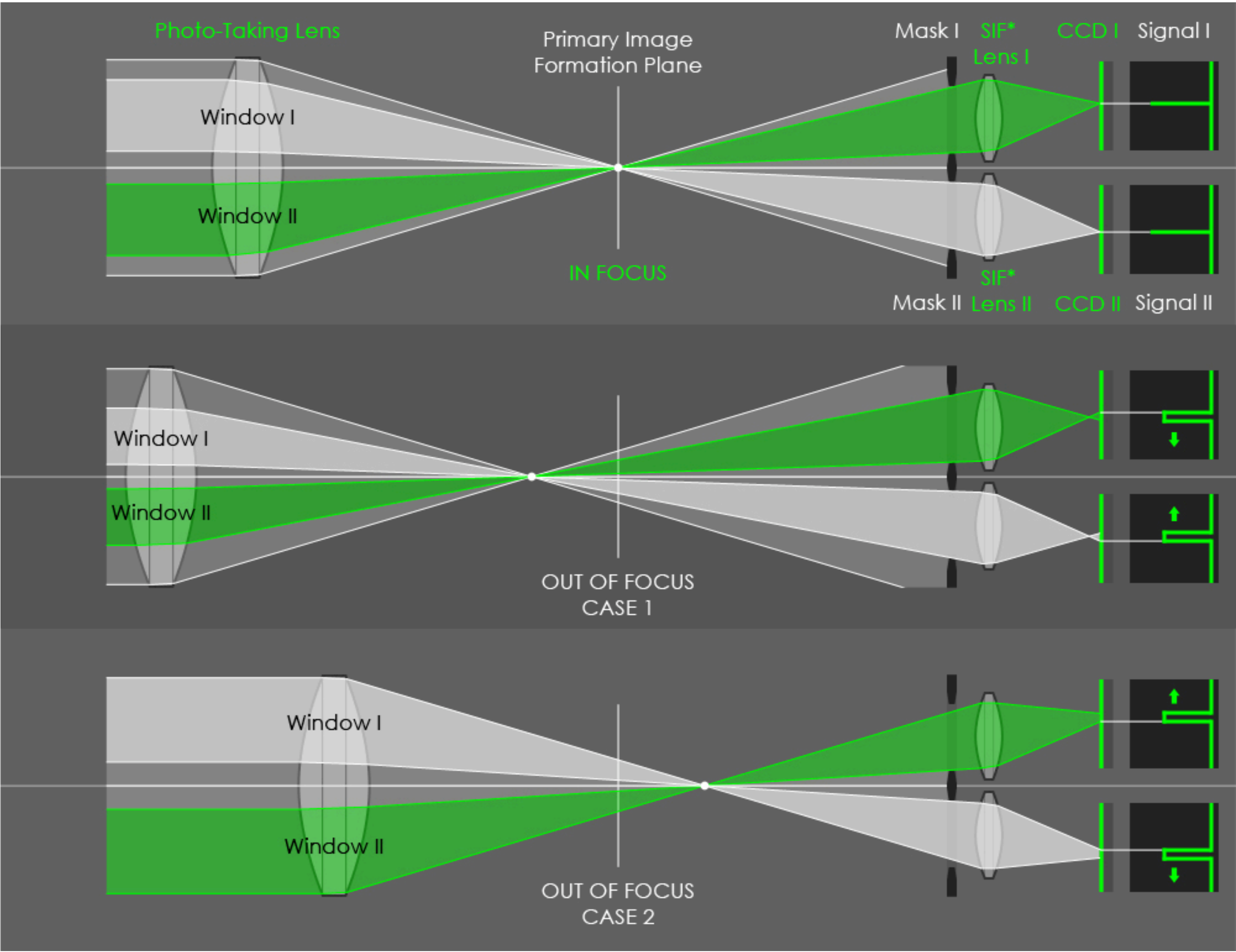


These masks make sure only a small "cone" of lights in each half of the lens is used in PD.

The AF lenses and AF sensors are fixed. They are placed in such a way that if the incident light is in-focus with by main lens, it will also be in-focus on the AF sensors.

# Phase Detection Auto Focus (PDAF)

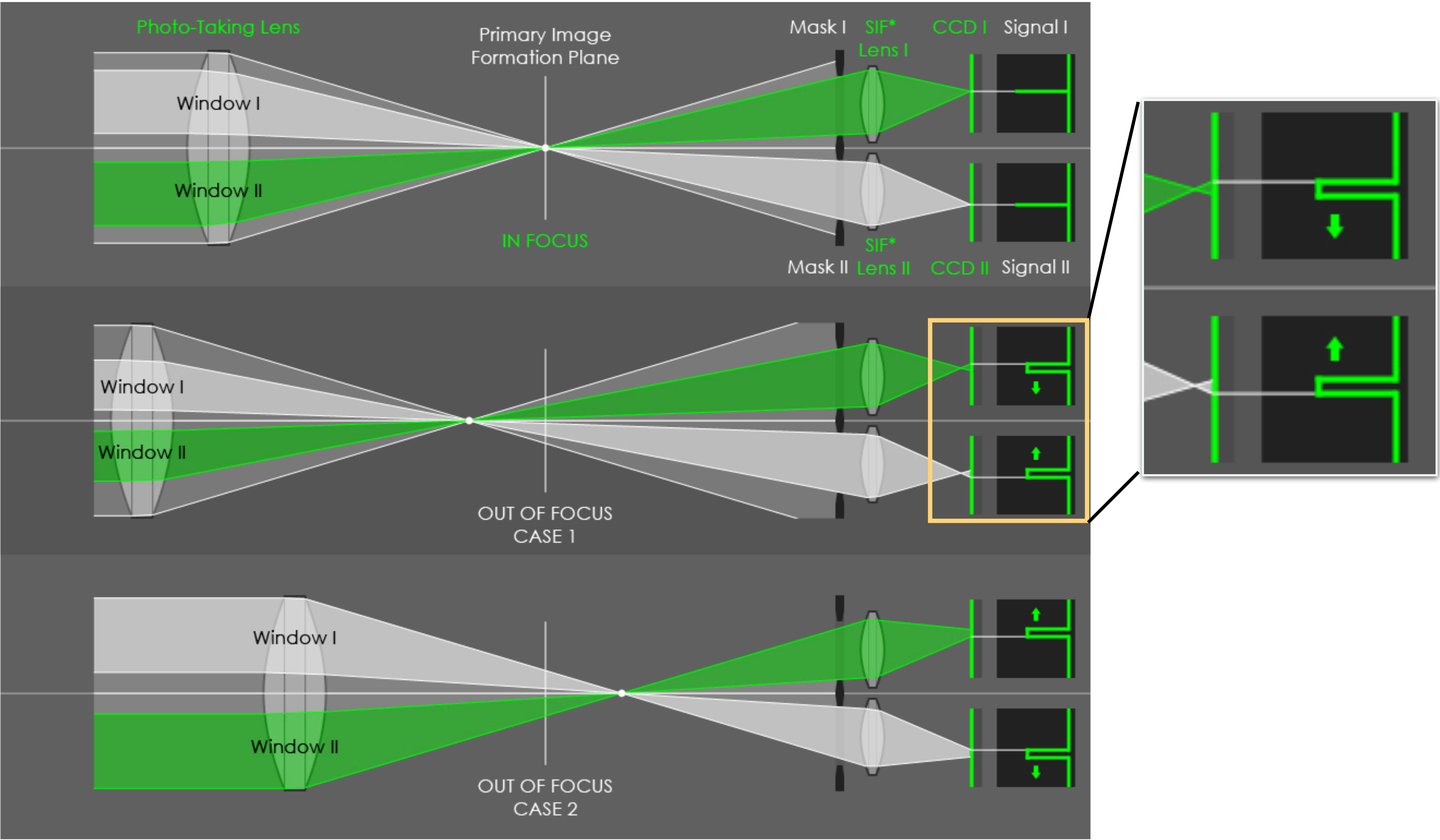
Assuming a point source in scene





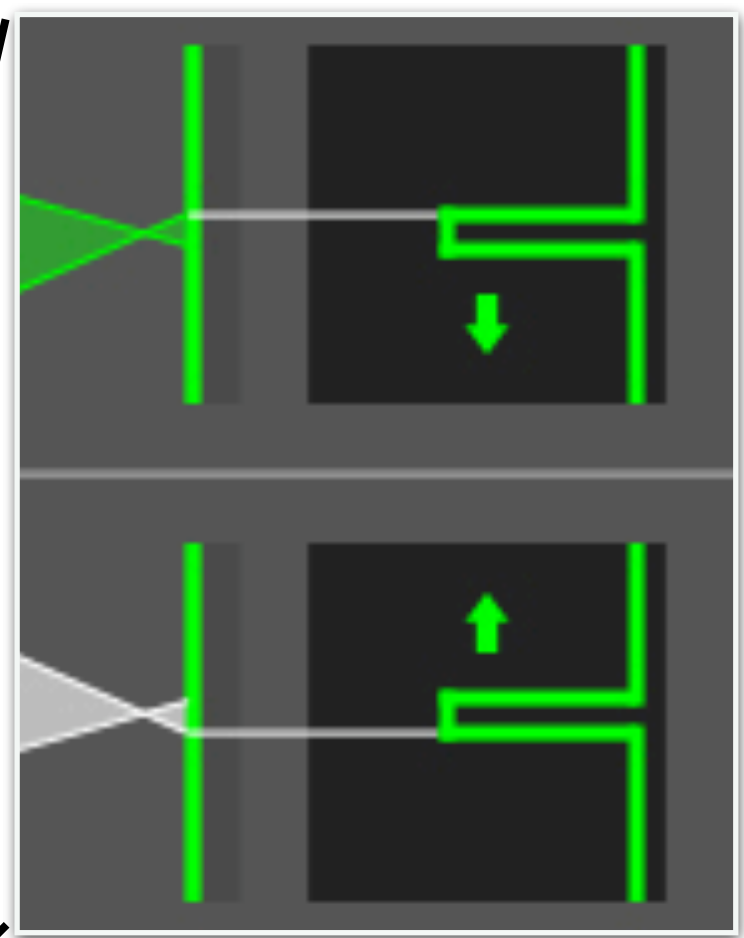
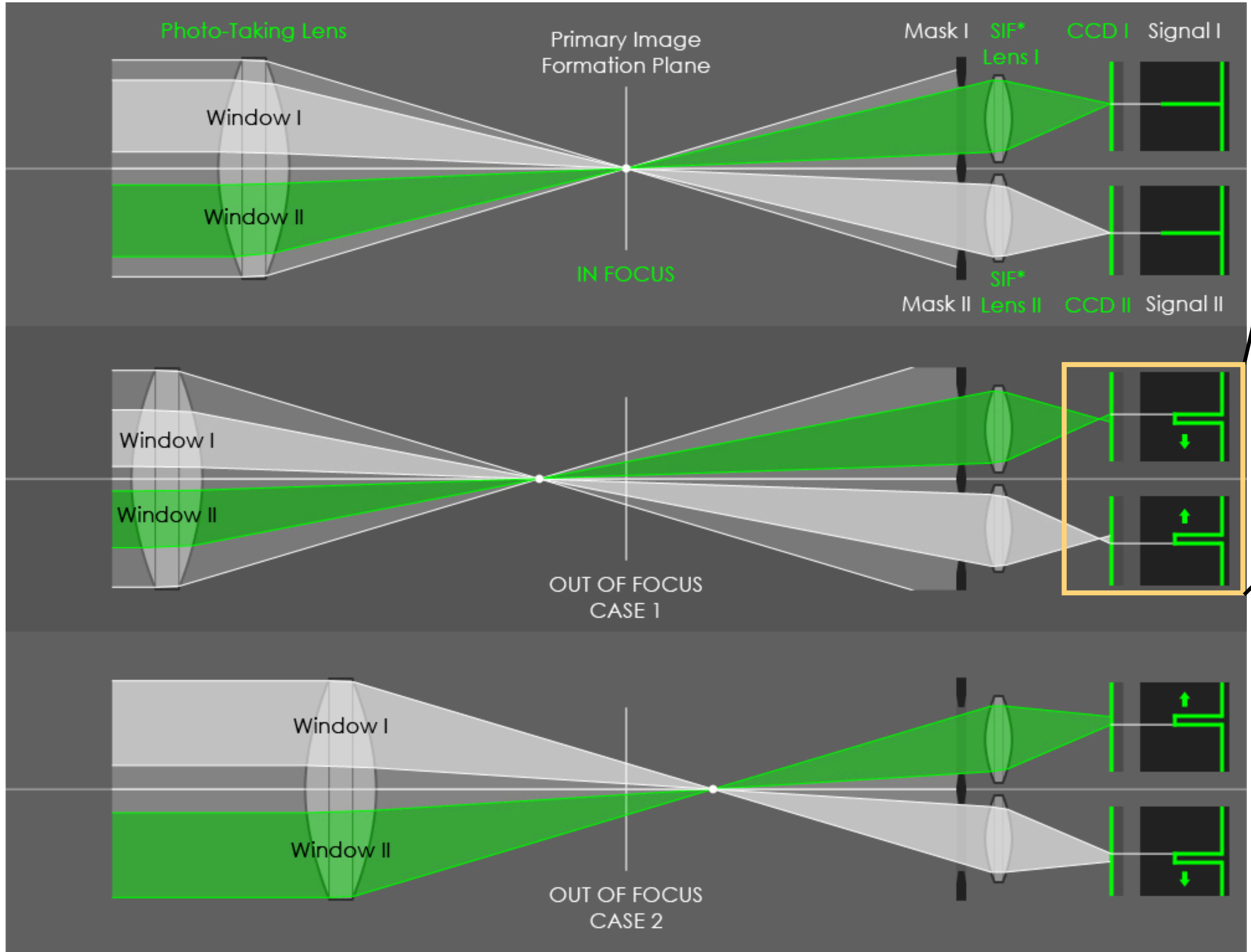
# Phase Detection Auto Focus (PDAF)

Assuming a point source in scene



# Phase Detection Auto Focus (PDAF)

Assuming a point source in scene

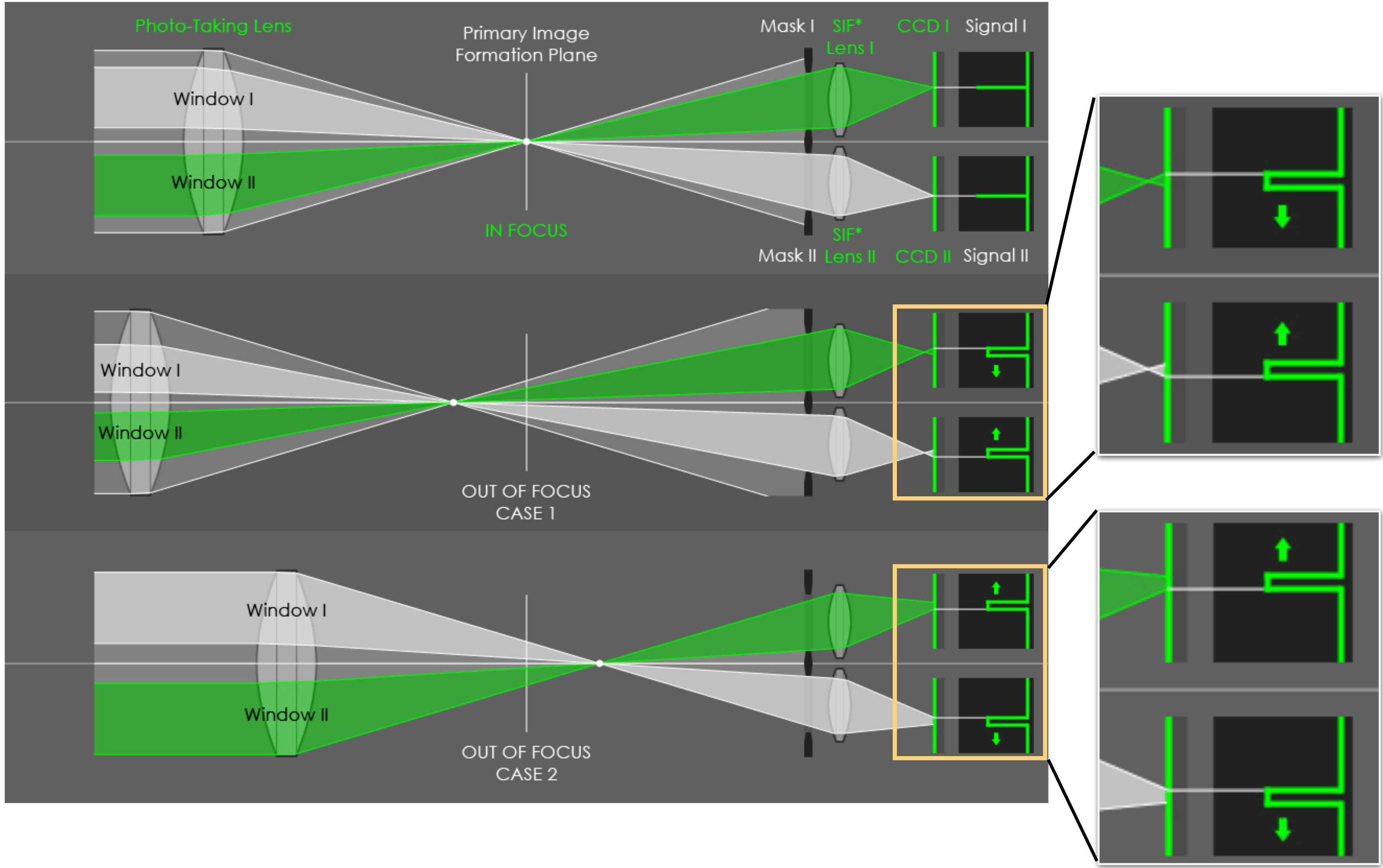


Peaks move **closer**.  
Lens should be moved **toward** sensor.



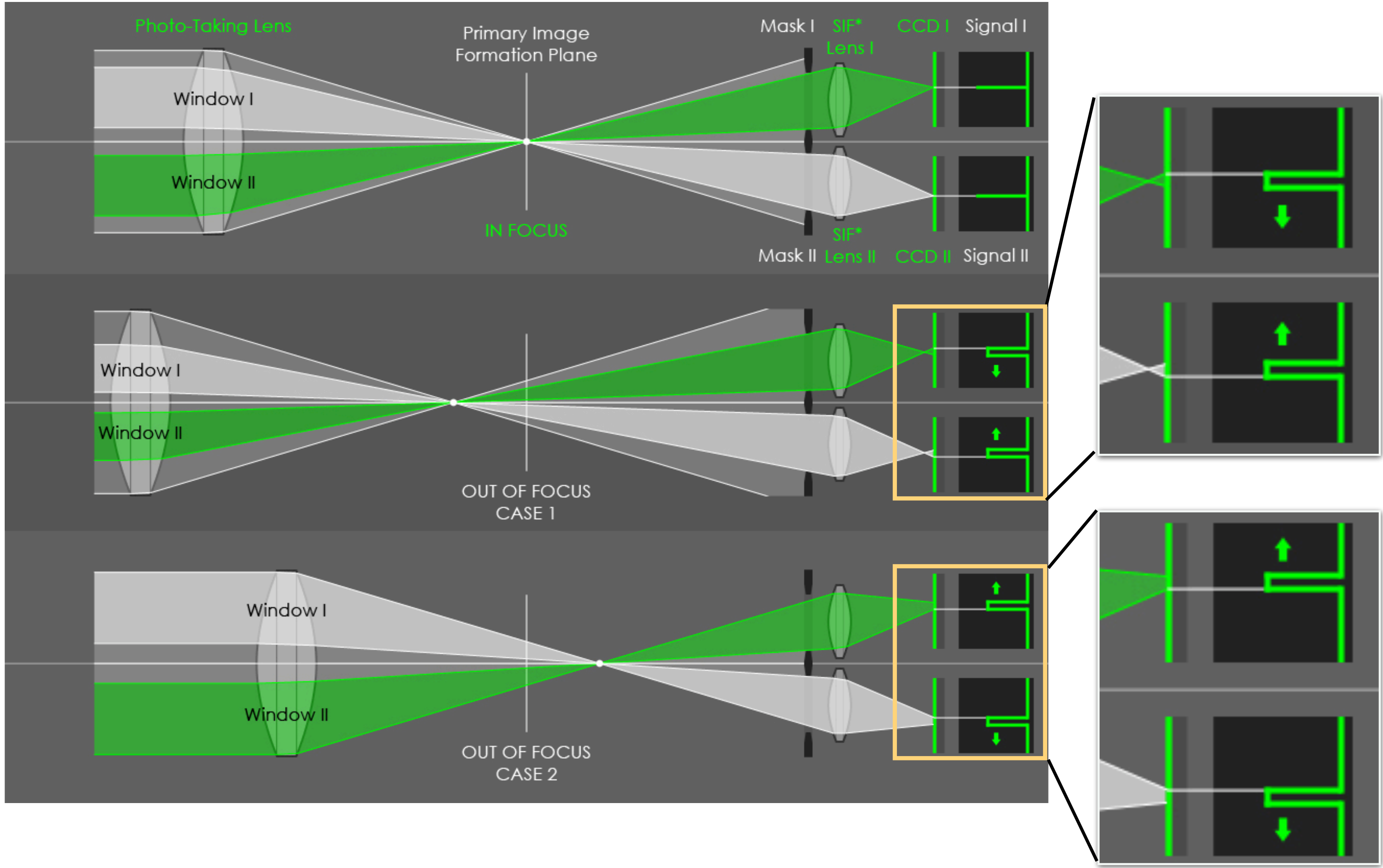
# Phase Detection Auto Focus (PDAF)

Assuming a point source in scene



# Phase Detection Auto Focus (PDAF)

Assuming a point source in scene



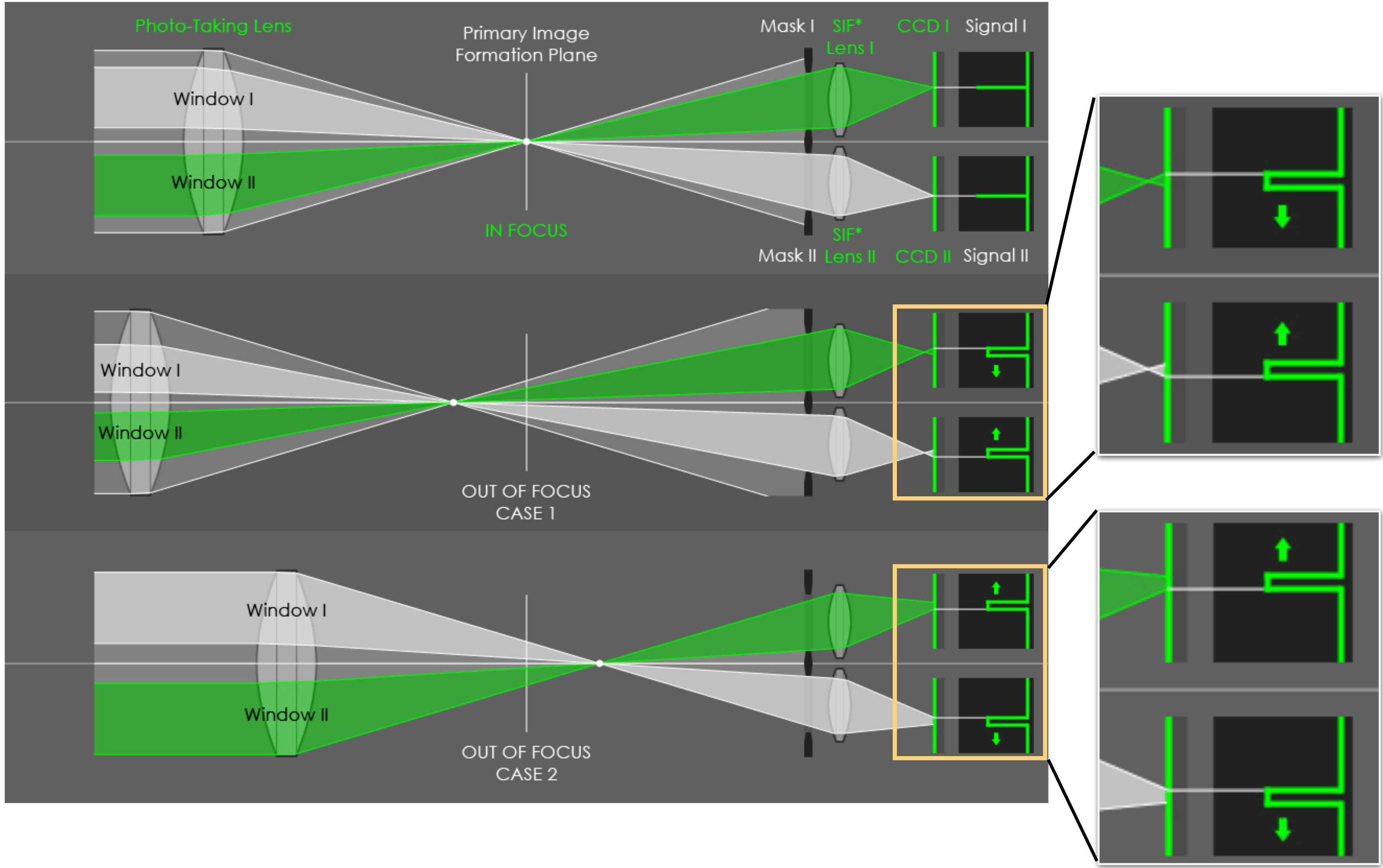
Peaks move **closer**.  
Lens should be moved **toward** sensor.

Peaks move **farther**.  
Lens should be moved **away from** sensor.

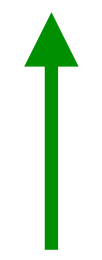


# Phase Detection Auto Focus (PDAF)

Assuming a point source in scene



Peaks move **closer**.  
 Lens should be moved **toward** sensor.

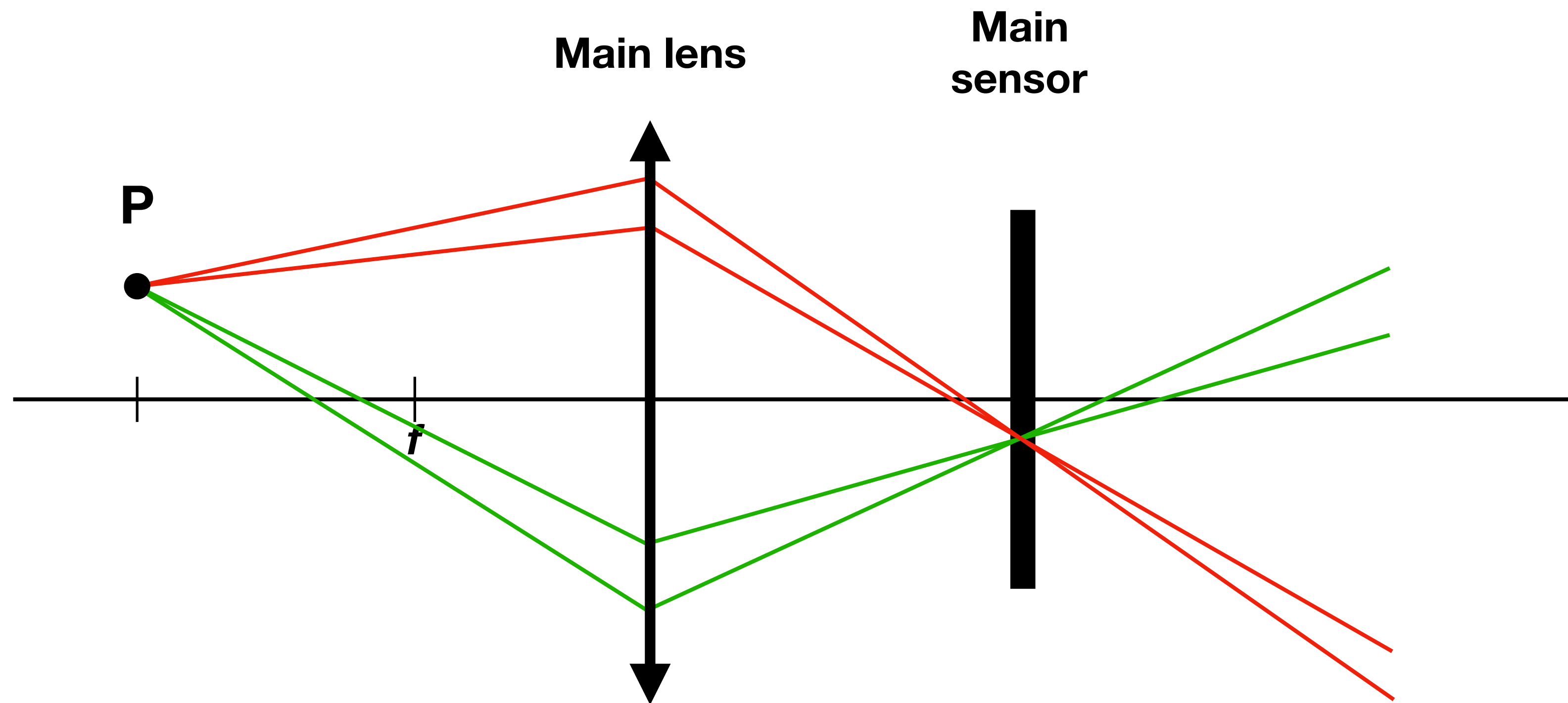


The amount of phase shift dictates how much to move the lens to focus!



Peaks move **farther**.  
 Lens should be moved **away from** sensor.

# Math Behind PDAF



**Assumption: everything is fixed; main lens is the only moving part.**

$d_1$ : distance between main sensor and AF lenses

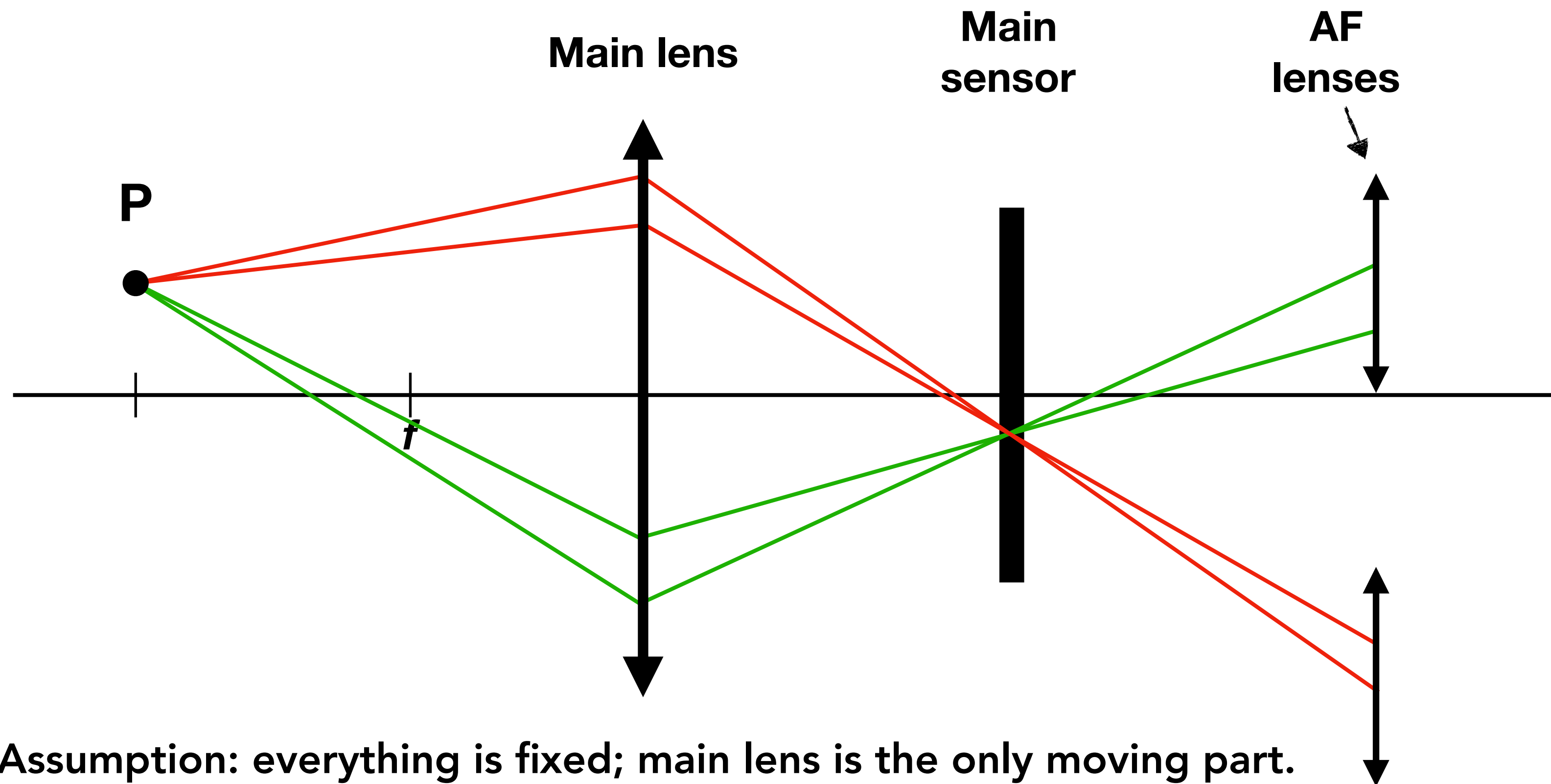
$d_2$ : distance between AF lens and AF sensor

**B**: distance between the centers of the two AF lenses

**O**: distance between the two pixels corresponding to **P**



# Math Behind PDAF



**Assumption: everything is fixed; main lens is the only moving part.**

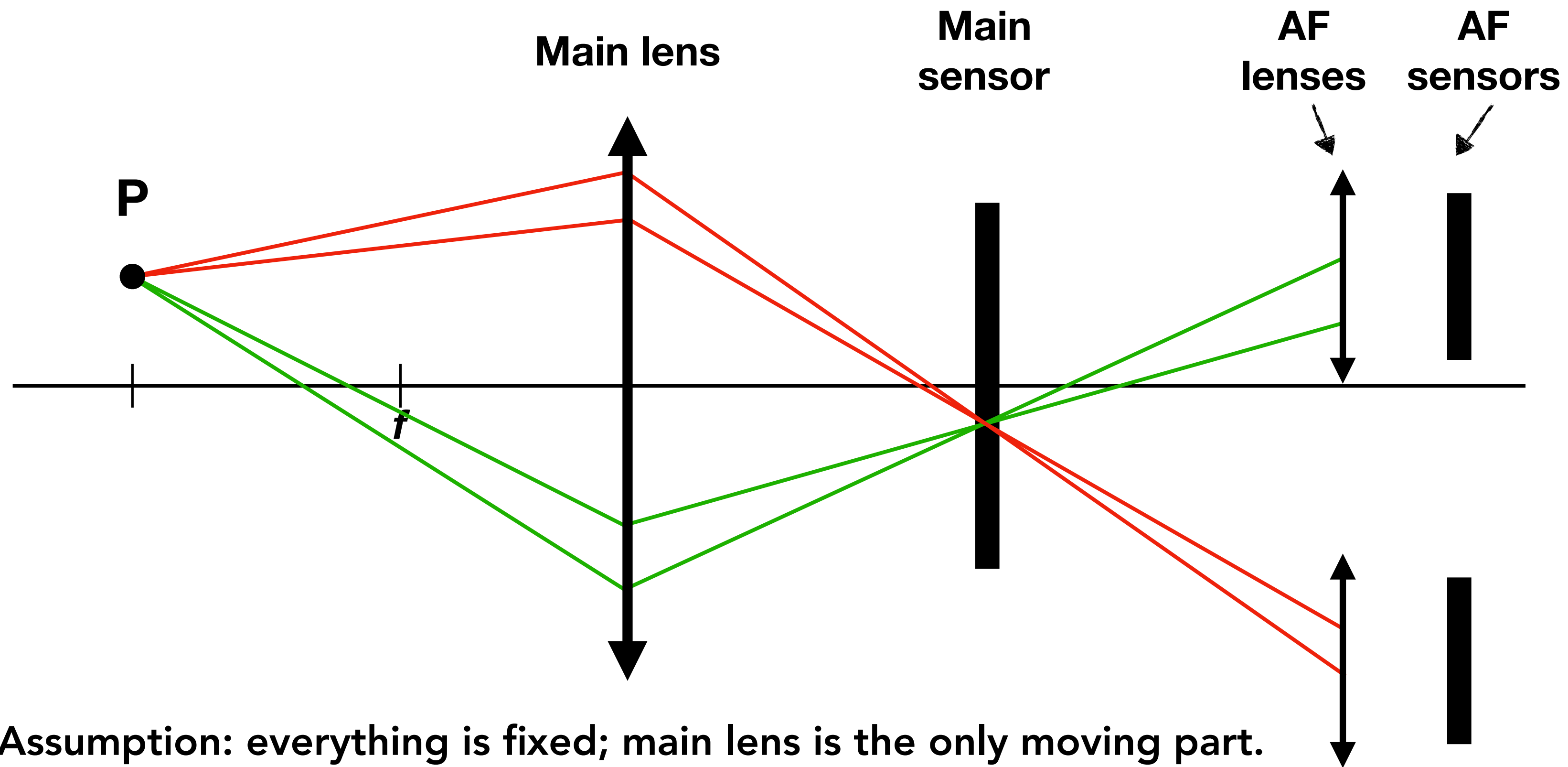
$d_1$ : distance between main sensor and AF lenses

$d_2$ : distance between AF lens and AF sensor

$B$ : distance between the centers of the two AF lenses

$O$ : distance between the two pixels corresponding to  $P$

# Math Behind PDAF



**Assumption: everything is fixed; main lens is the only moving part.**

$d_1$ : distance between main sensor and AF lenses

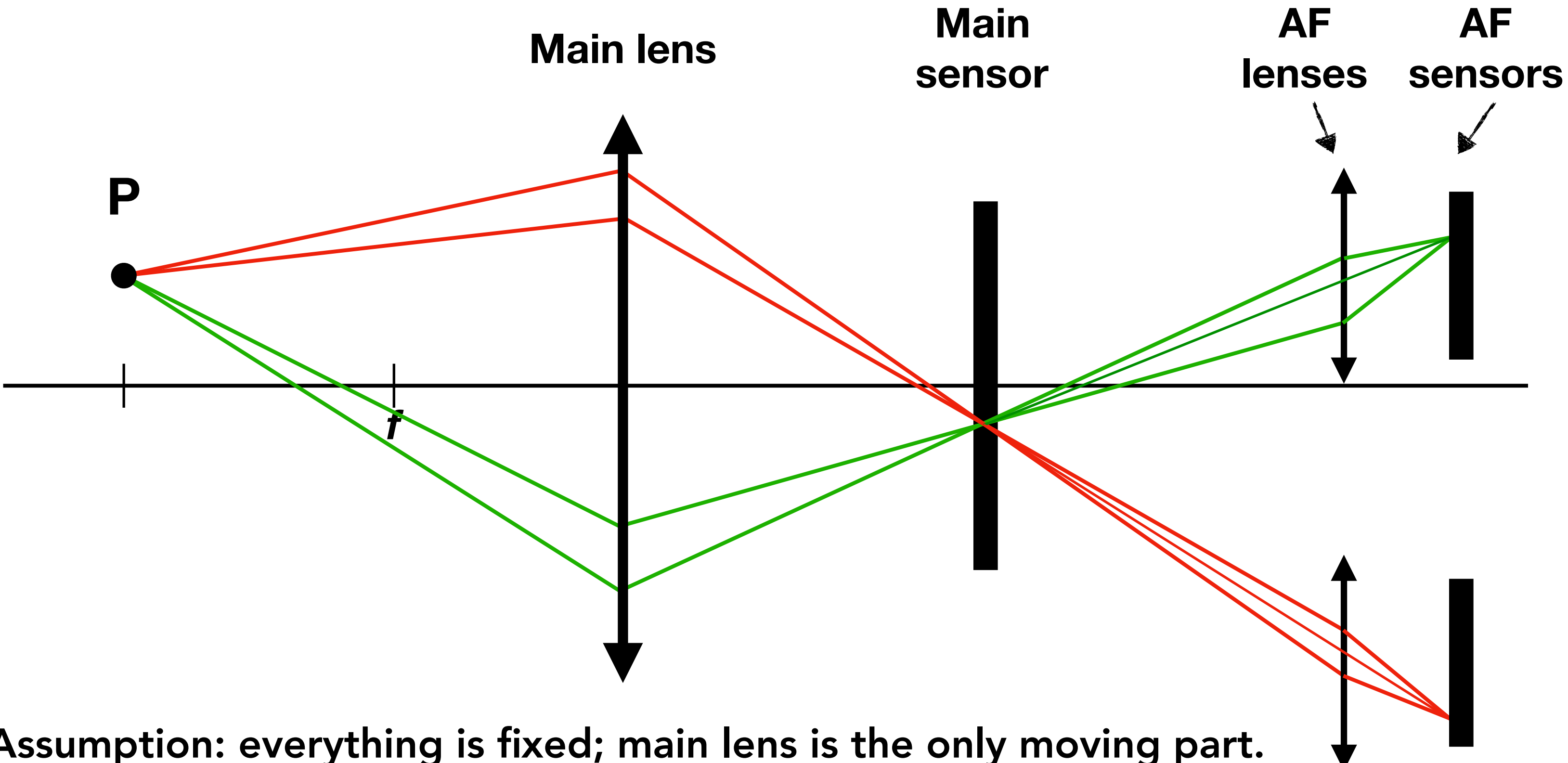
$d_2$ : distance between AF lens and AF sensor

$B$ : distance between the centers of the two AF lenses

$O$ : distance between the two pixels corresponding to  $P$



# Math Behind PDAF



**Assumption: everything is fixed; main lens is the only moving part.**

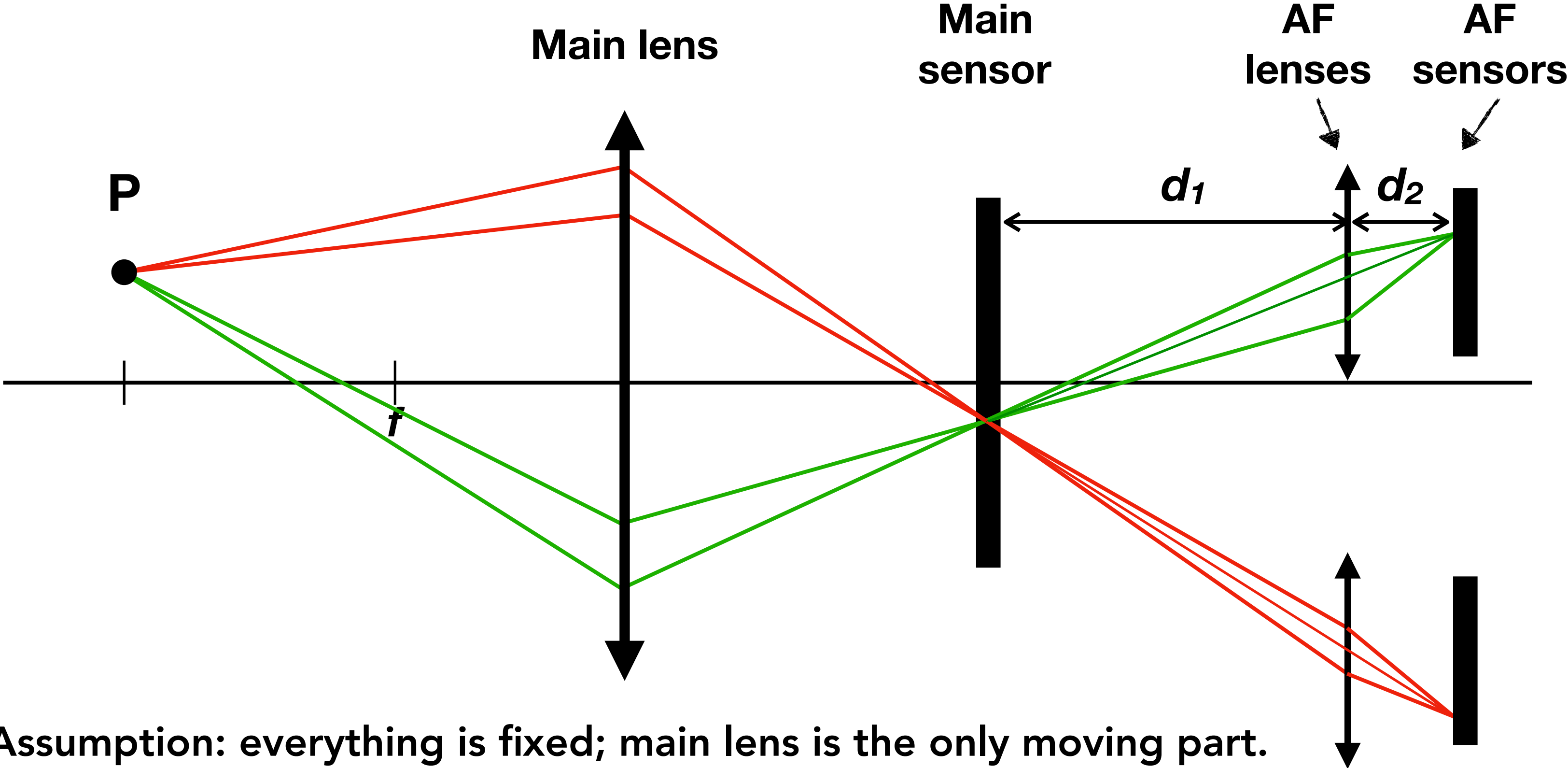
$d_1$ : distance between main sensor and AF lenses

$d_2$ : distance between AF lens and AF sensor

$B$ : distance between the centers of the two AF lenses

$O$ : distance between the two pixels corresponding to  $P$

# Math Behind PDAF



**Assumption: everything is fixed; main lens is the only moving part.**

$d_1$ : distance between main sensor and AF lenses

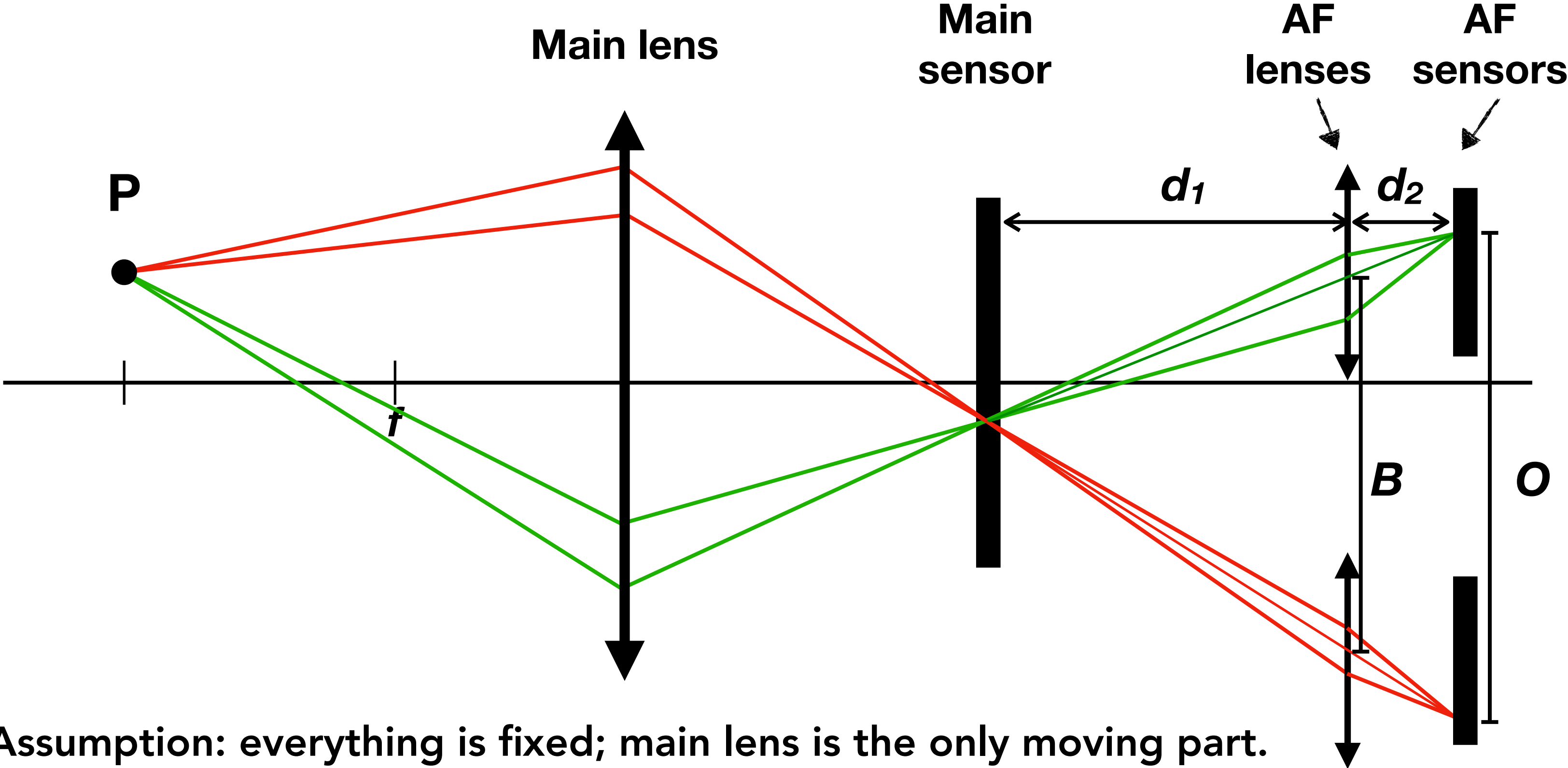
$d_2$ : distance between AF lens and AF sensor

$B$ : distance between the centers of the two AF lenses

$O$ : distance between the two pixels corresponding to  $P$



# Math Behind PDAF



**Assumption: everything is fixed; main lens is the only moving part.**

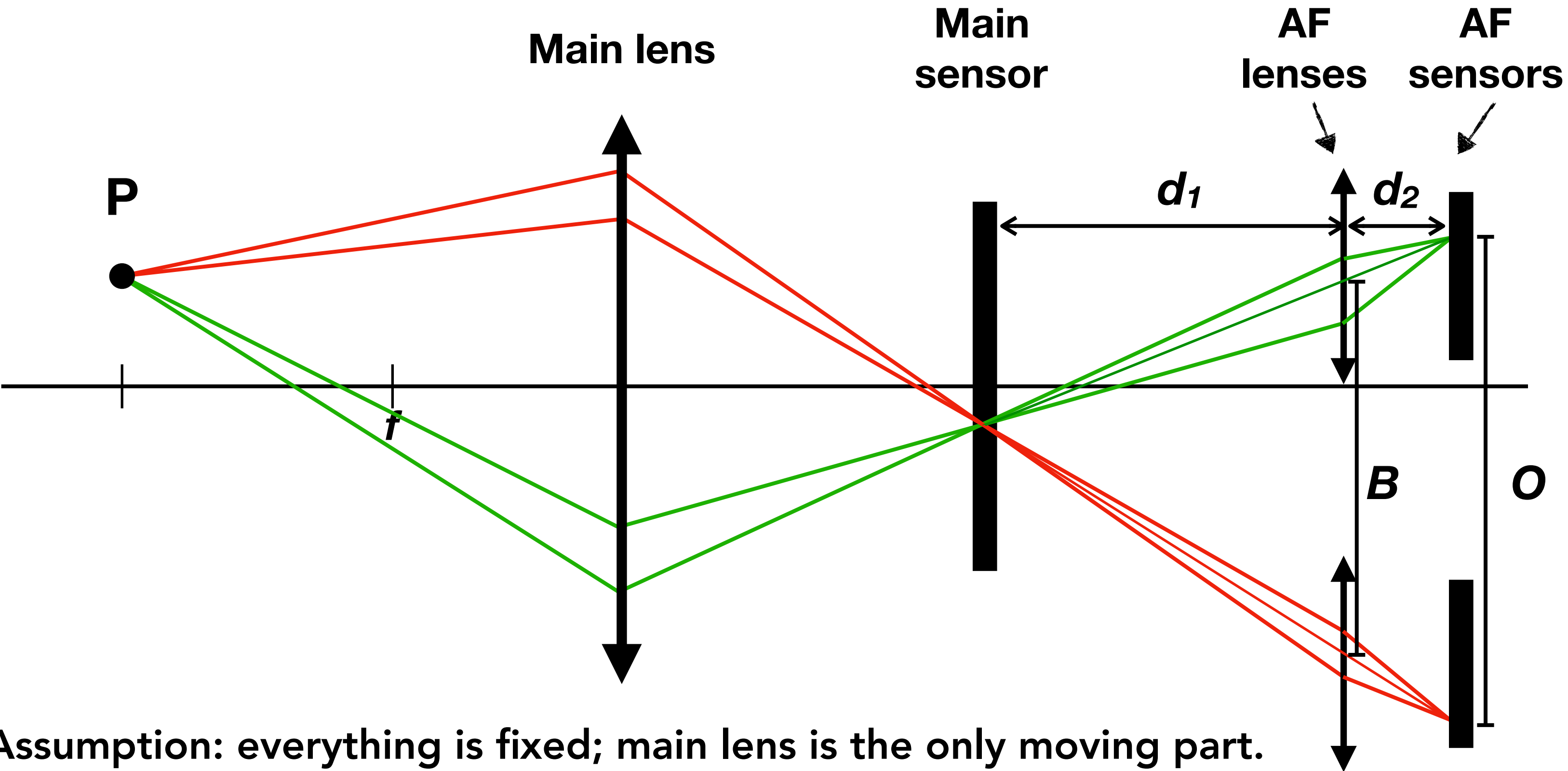
$d_1$ : distance between main sensor and AF lenses

$d_2$ : distance between AF lens and AF sensor

$B$ : distance between the centers of the two AF lenses

$O$ : distance between the two pixels corresponding to  $P$

# Math Behind PDAF



$$\frac{d_1}{d_1 + d_2} = \frac{B}{O}$$

↓

$$O = \frac{(d_1 + d_2)B}{d_1}$$

**Assumption: everything is fixed; main lens is the only moving part.**

**d<sub>1</sub>**: distance between main sensor and AF lenses

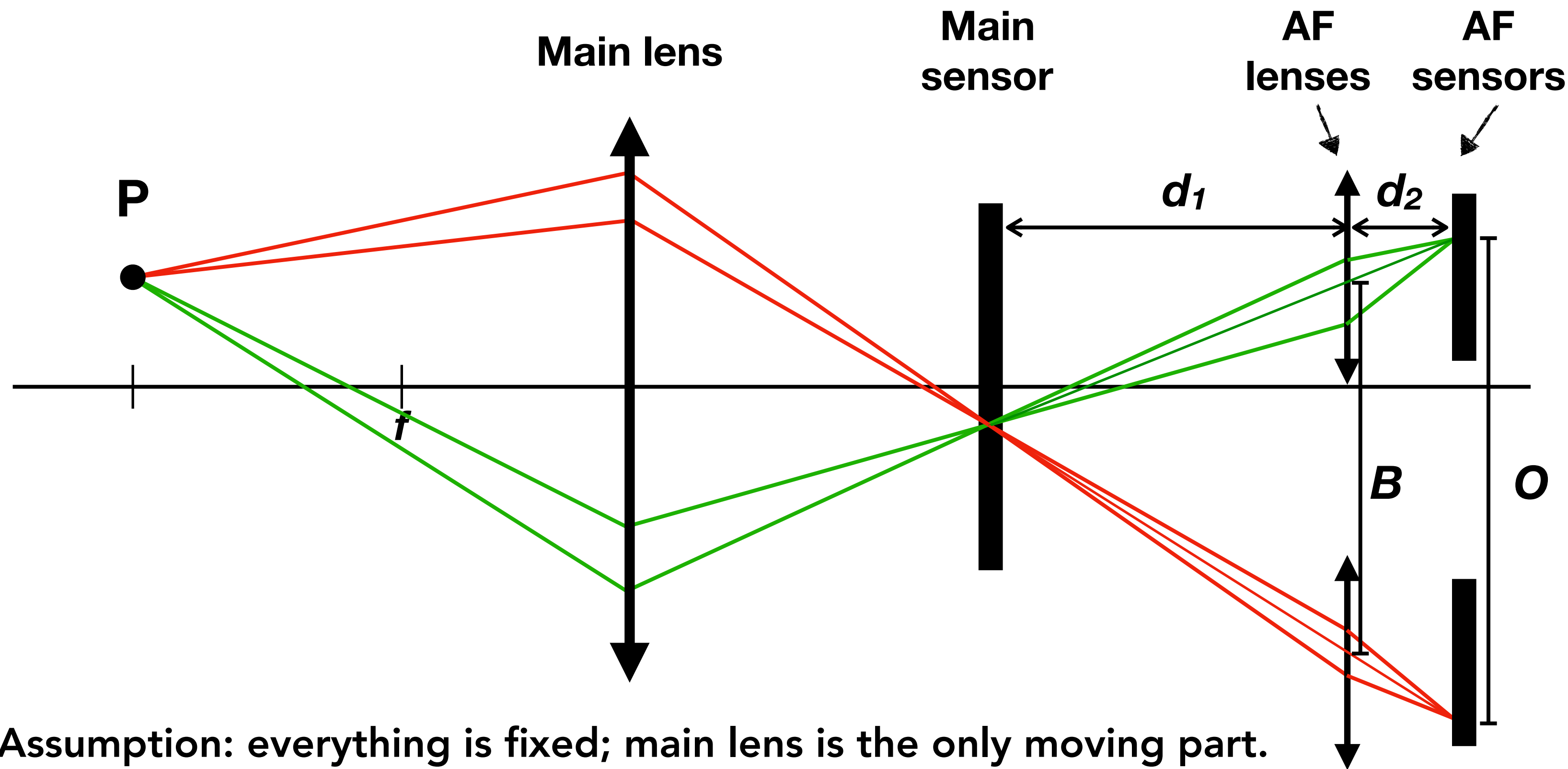
**d<sub>2</sub>**: distance between AF lens and AF sensor

**B**: distance between the centers of the two AF lenses

**O**: distance between the two pixels corresponding to P



# Math Behind PDAF



**Assumption: everything is fixed; main lens is the only moving part.**

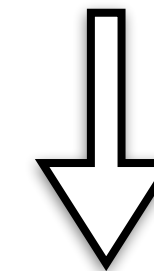
$d_1$ : distance between main sensor and AF lenses

$d_2$ : distance between AF lens and AF sensor

$B$ : distance between the centers of the two AF lenses

$O$ : distance between the two pixels corresponding to  $P$

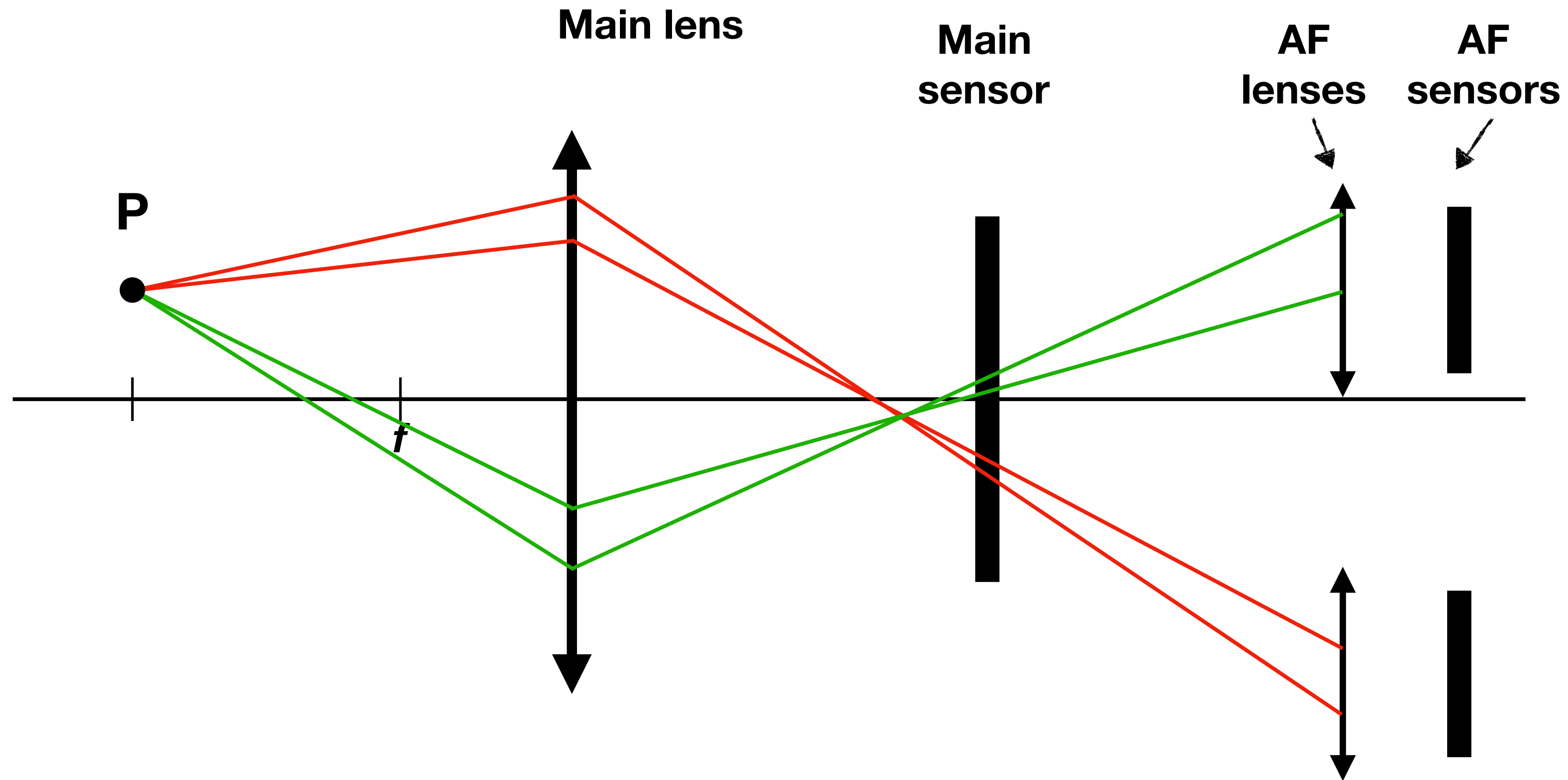
$$\frac{d_1}{d_1 + d_2} = \frac{B}{O}$$



$$O = \frac{(d_1 + d_2)B}{d_1}$$

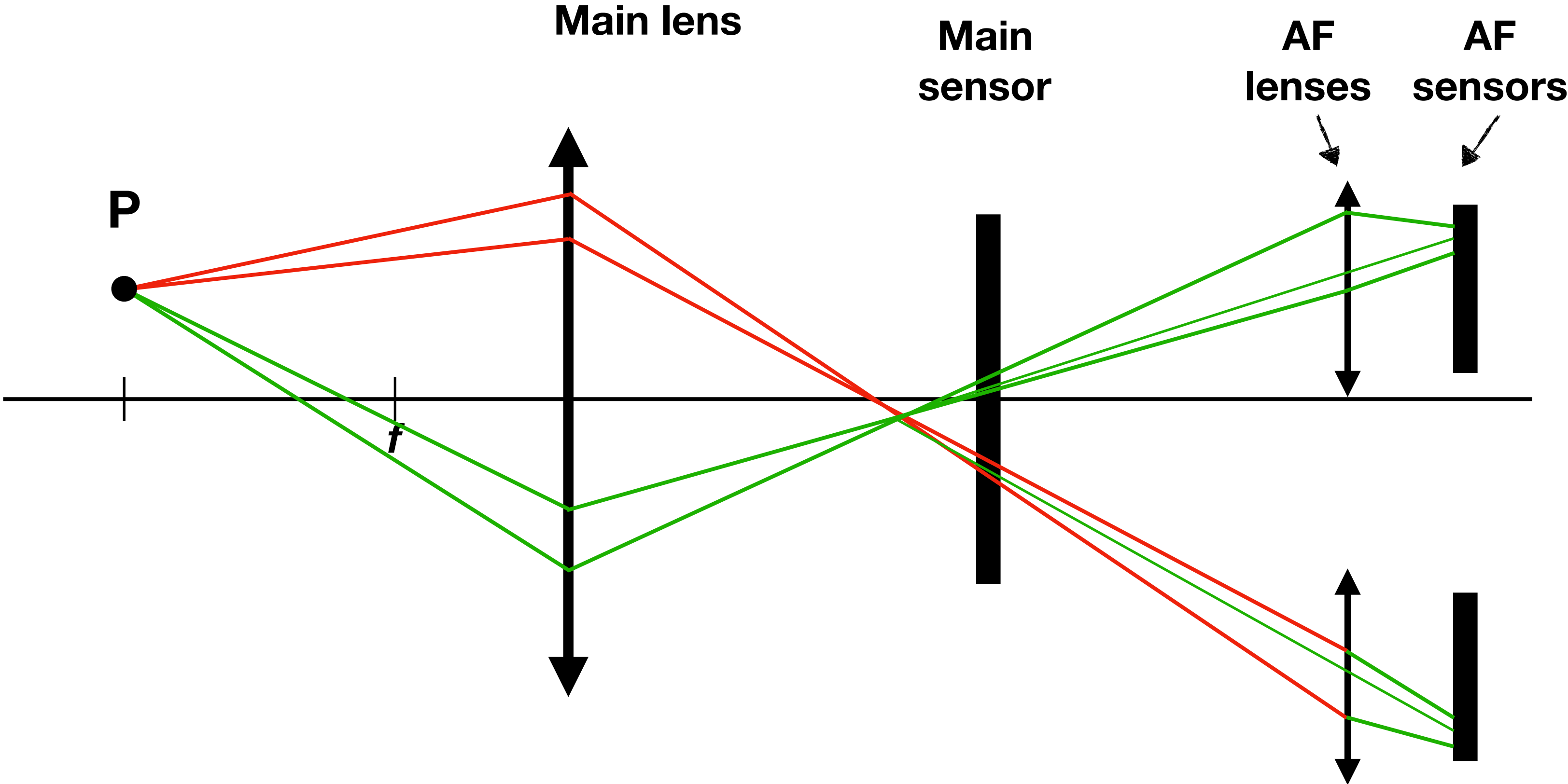
As long as the scene point  $P$  is in-focus, the offset between the corresponding pixels,  $O$ , is **fixed** and is known **offline**.

# Math Behind PDAF

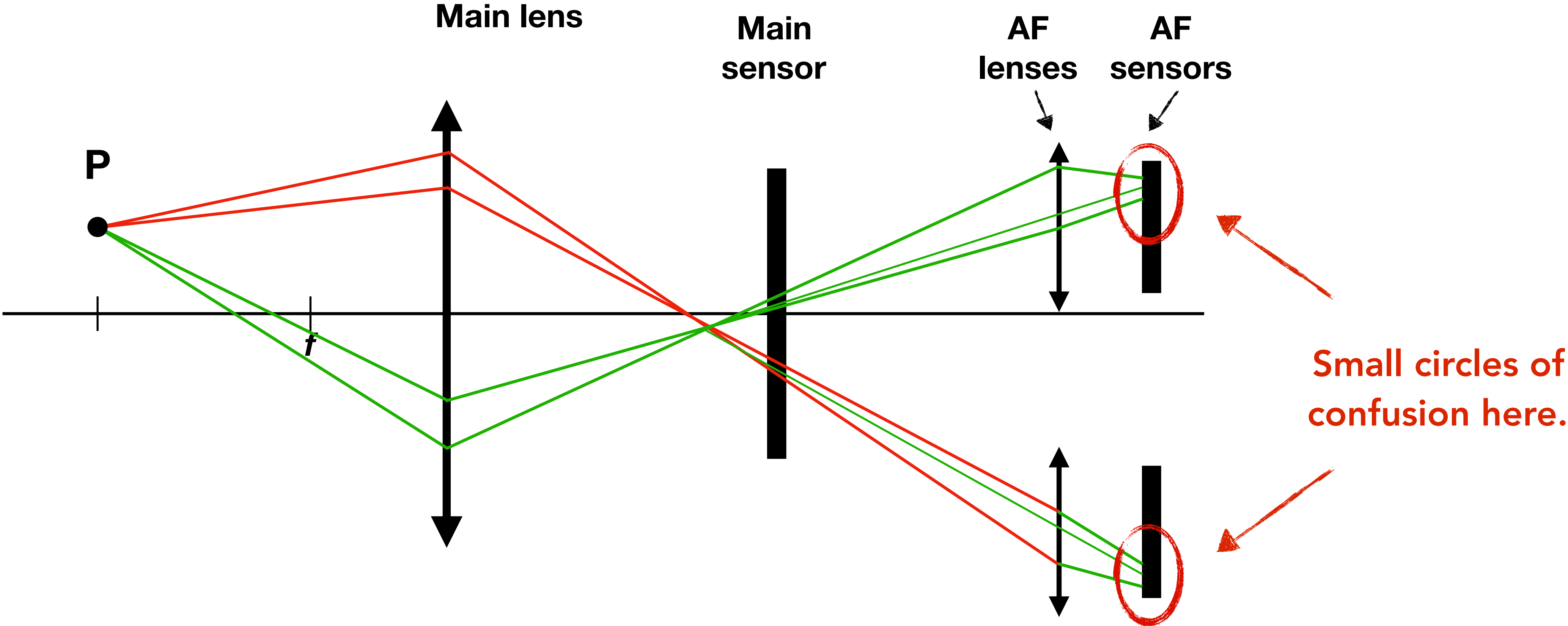




# Math Behind PDAF



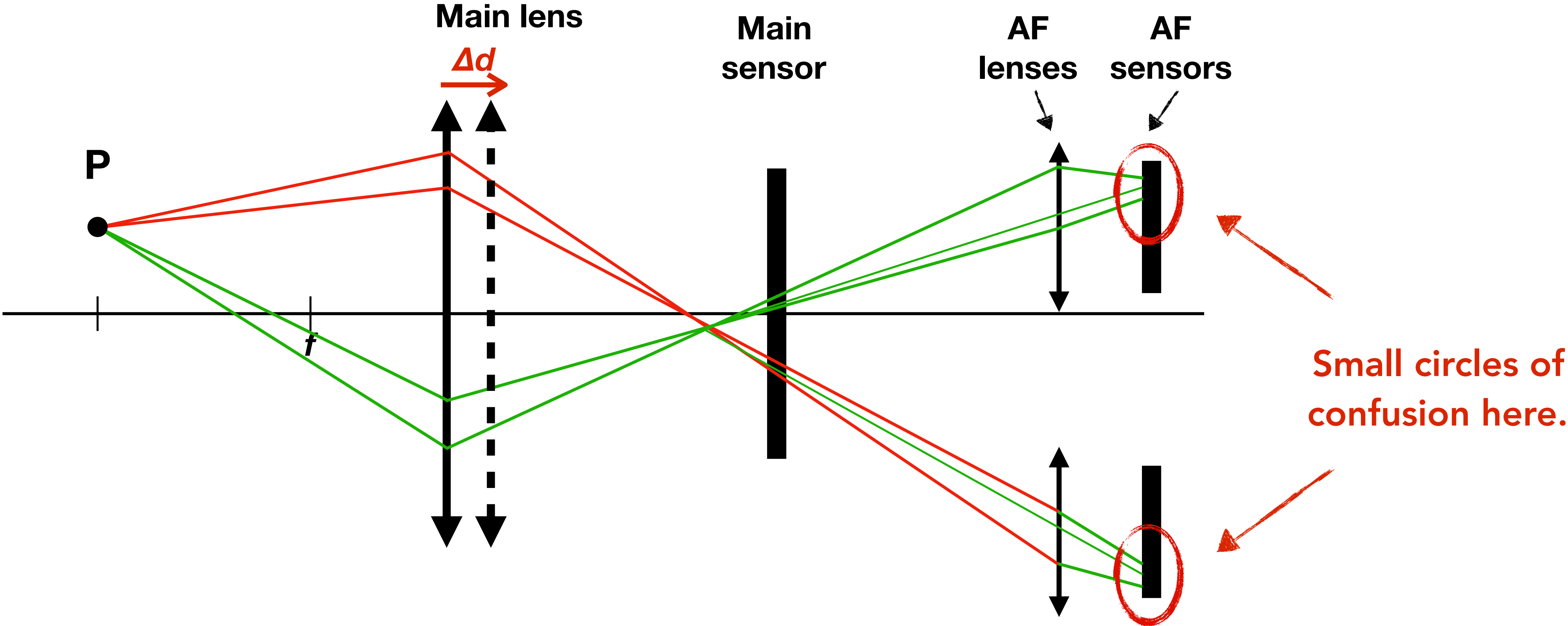
# Math Behind PDAF



\* The circles of confusion (CoC) on the AF sensors are usually very small because the apertures of the AF lenses are small (recall the DOF equation). So we will use pinhole camera models to analyze the AF lenses/sensors.

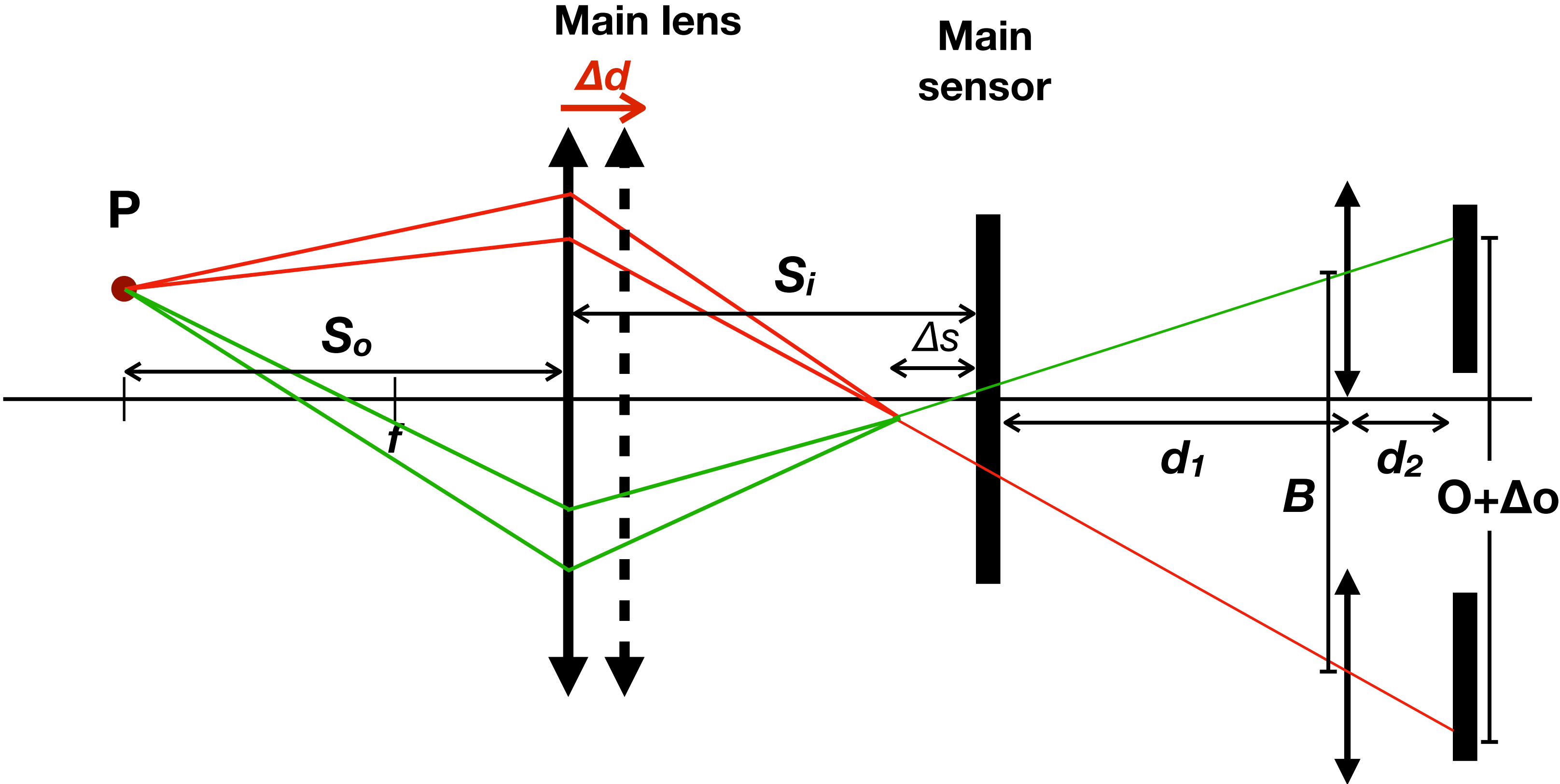


# Math Behind PDAF



\* The circles of confusion (CoC) on the AF sensors are usually very small because the apertures of the AF lenses are small (recall the DOF equation). So we will use pinhole camera models to analyze the AF lenses/sensors.

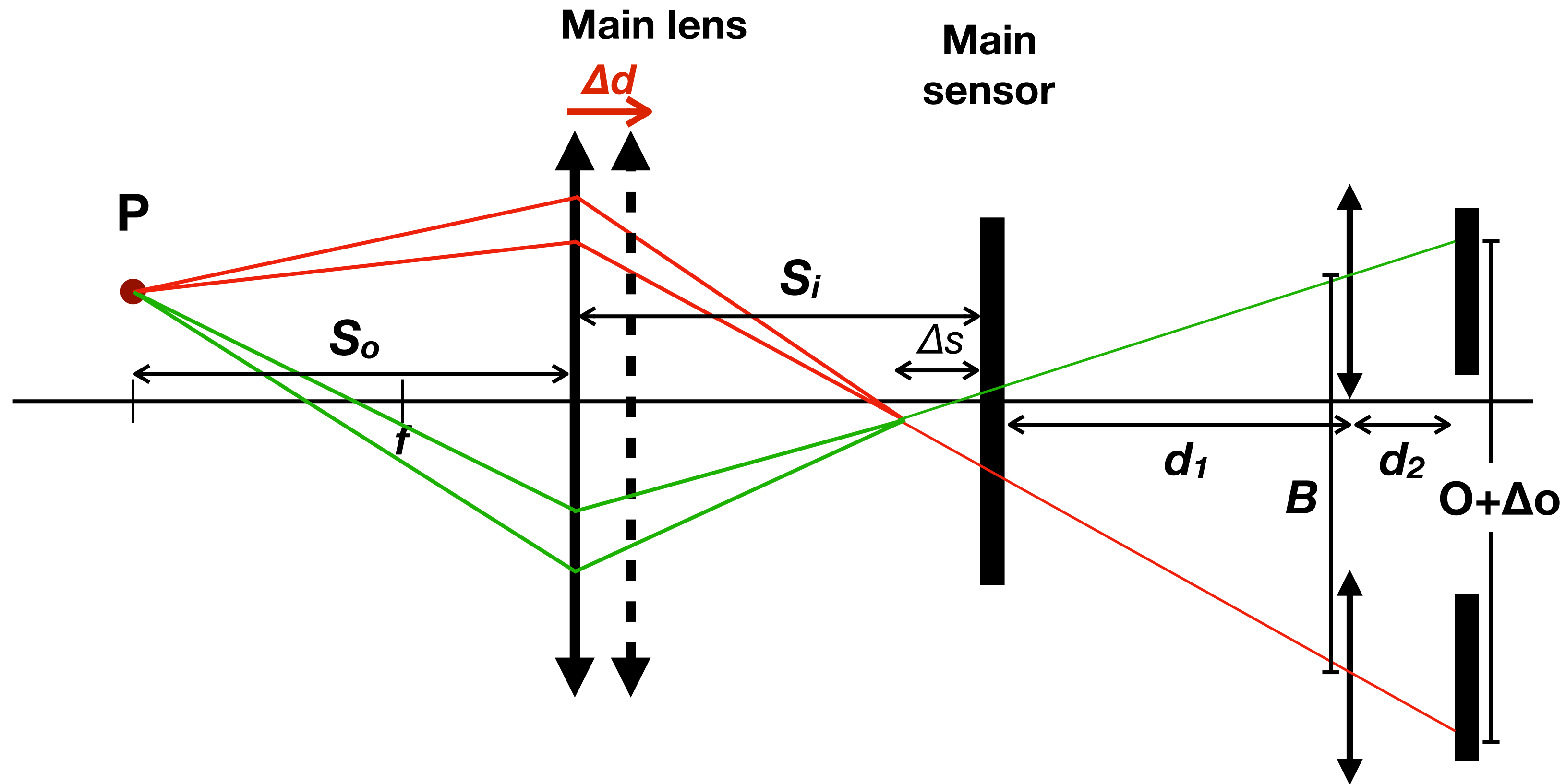
# Math Behind PDAF



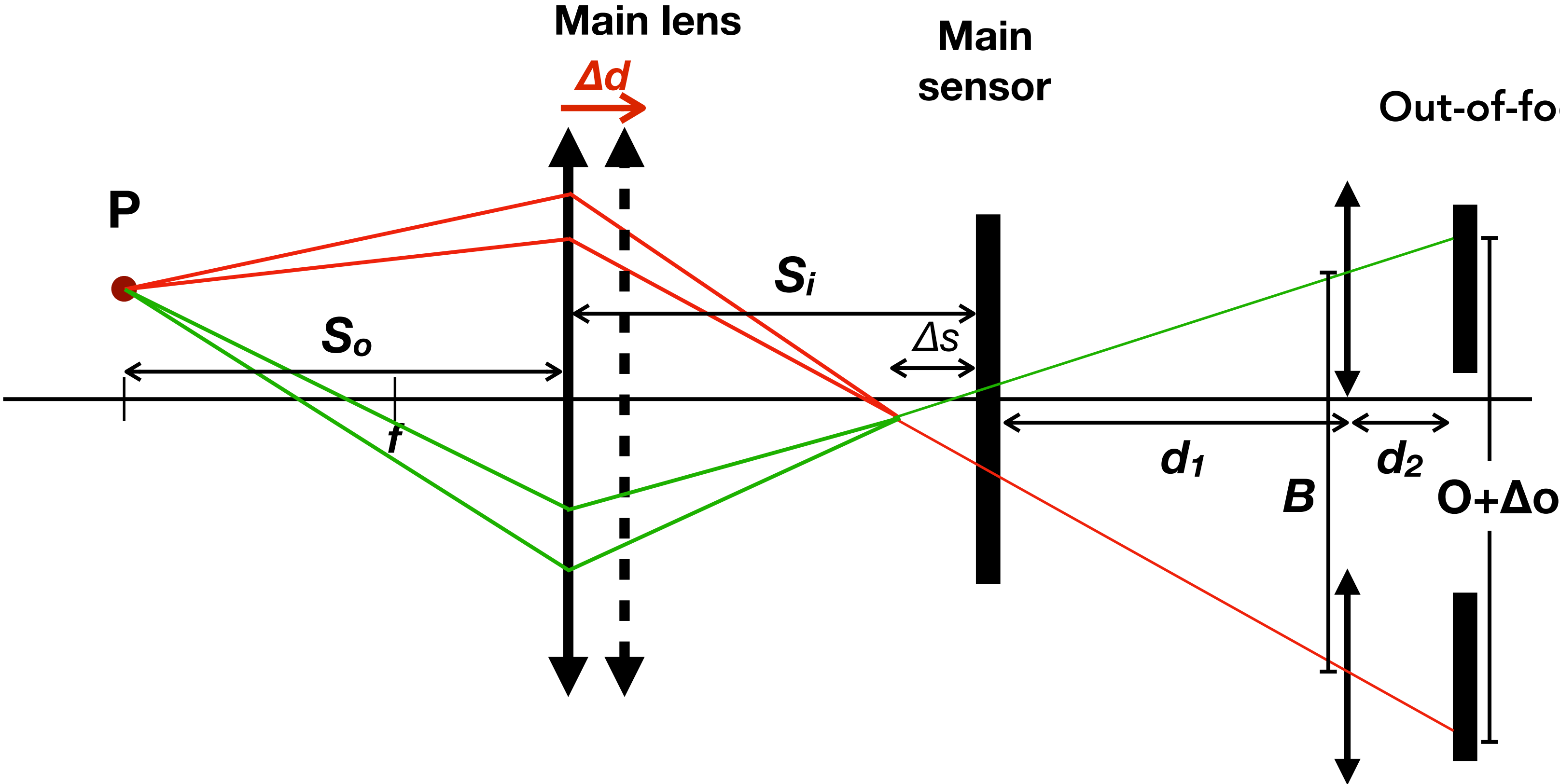


# Math Behind PDAF

In-focus:  $\frac{d_1}{d_1 + d_2} = \frac{B}{O}$



# Math Behind PDAF

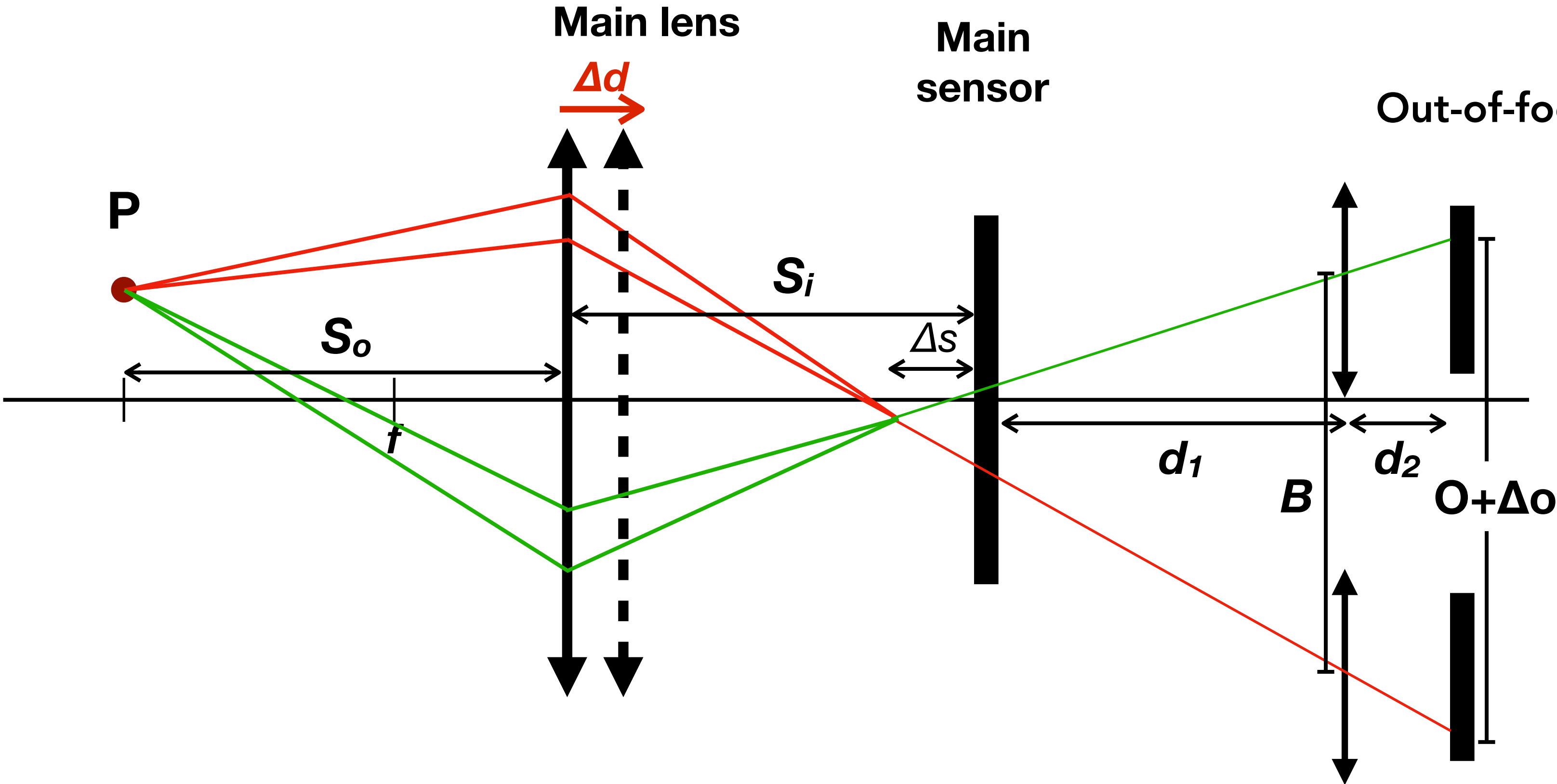


In-focus: 
$$\frac{d_1}{d_1 + d_2} = \frac{B}{O}$$

Out-of-focus: 
$$\frac{d_1 + \Delta s}{d_1 + \Delta s + d_2} = \frac{B}{O + \Delta o}$$



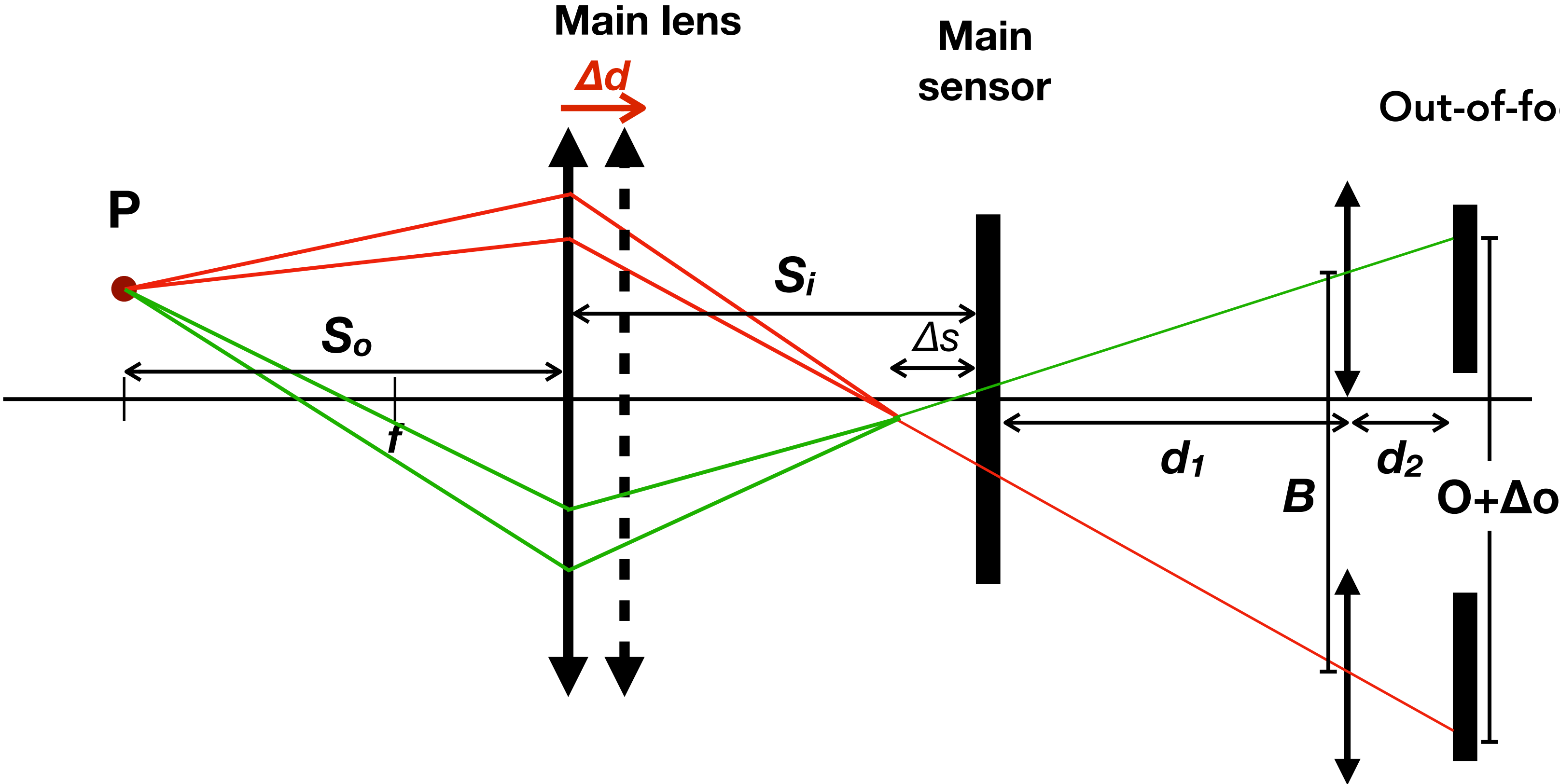
# Math Behind PDAF



In-focus:  $\frac{d_1}{d_1 + d_2} = \frac{B}{O}$  Phase shift

Out-of-focus:  $\frac{d_1 + \Delta s}{d_1 + \Delta s + d_2} = \frac{B}{O + \Delta o}$

# Math Behind PDAF



In-focus:  $\frac{d_1}{d_1 + d_2} = \frac{B}{O}$  Phase shift

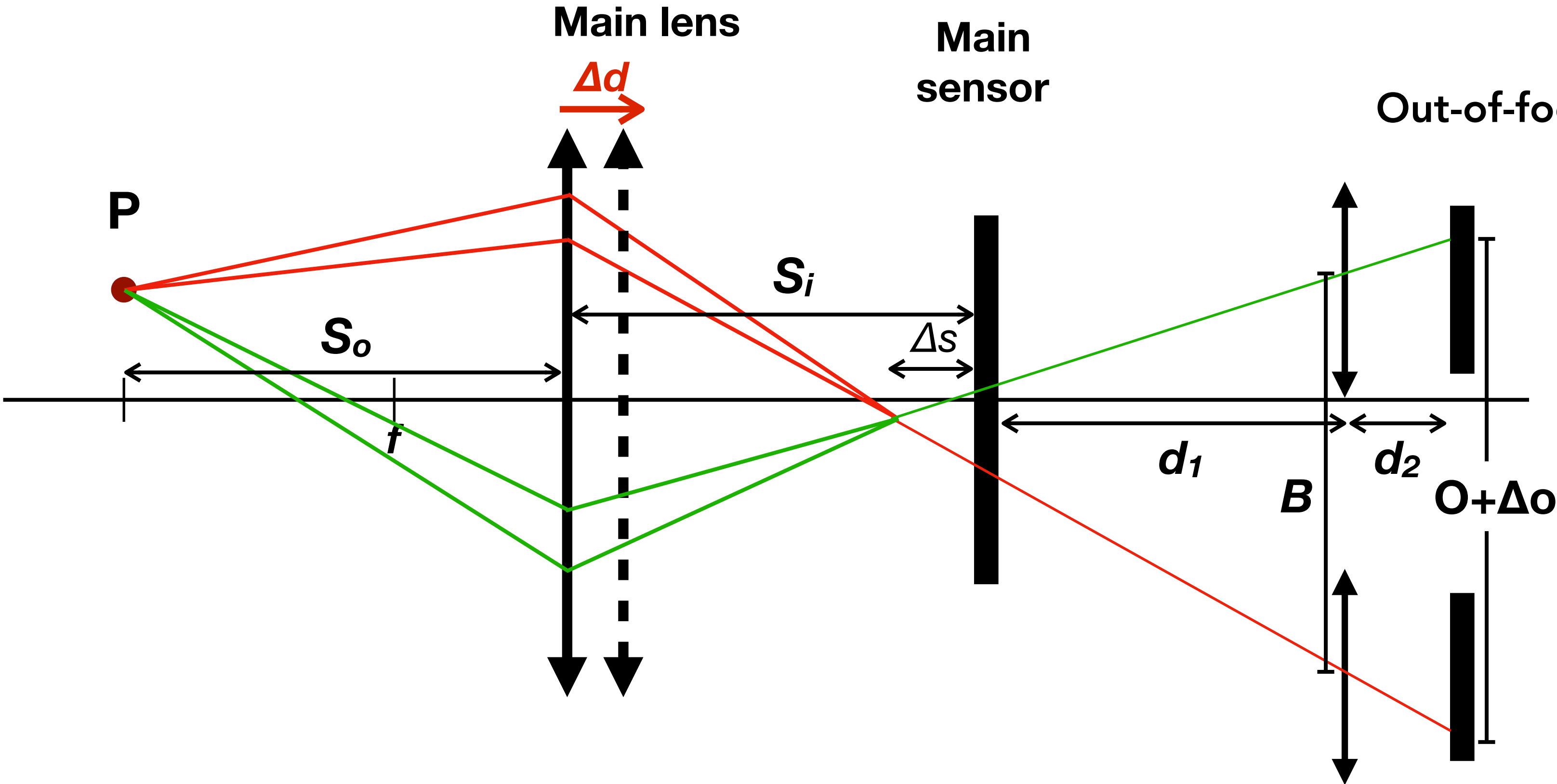
Out-of-focus:  $\frac{d_1 + \Delta s}{d_1 + \Delta s + d_2} = \frac{B}{O + \Delta o}$

$$\frac{1}{S_o} + \frac{1}{S_i - \Delta s} = \frac{1}{f}$$

$$\frac{1}{S_o + \Delta d} + \frac{1}{S_i - \Delta d} = \frac{1}{f}$$



# Math Behind PDAF



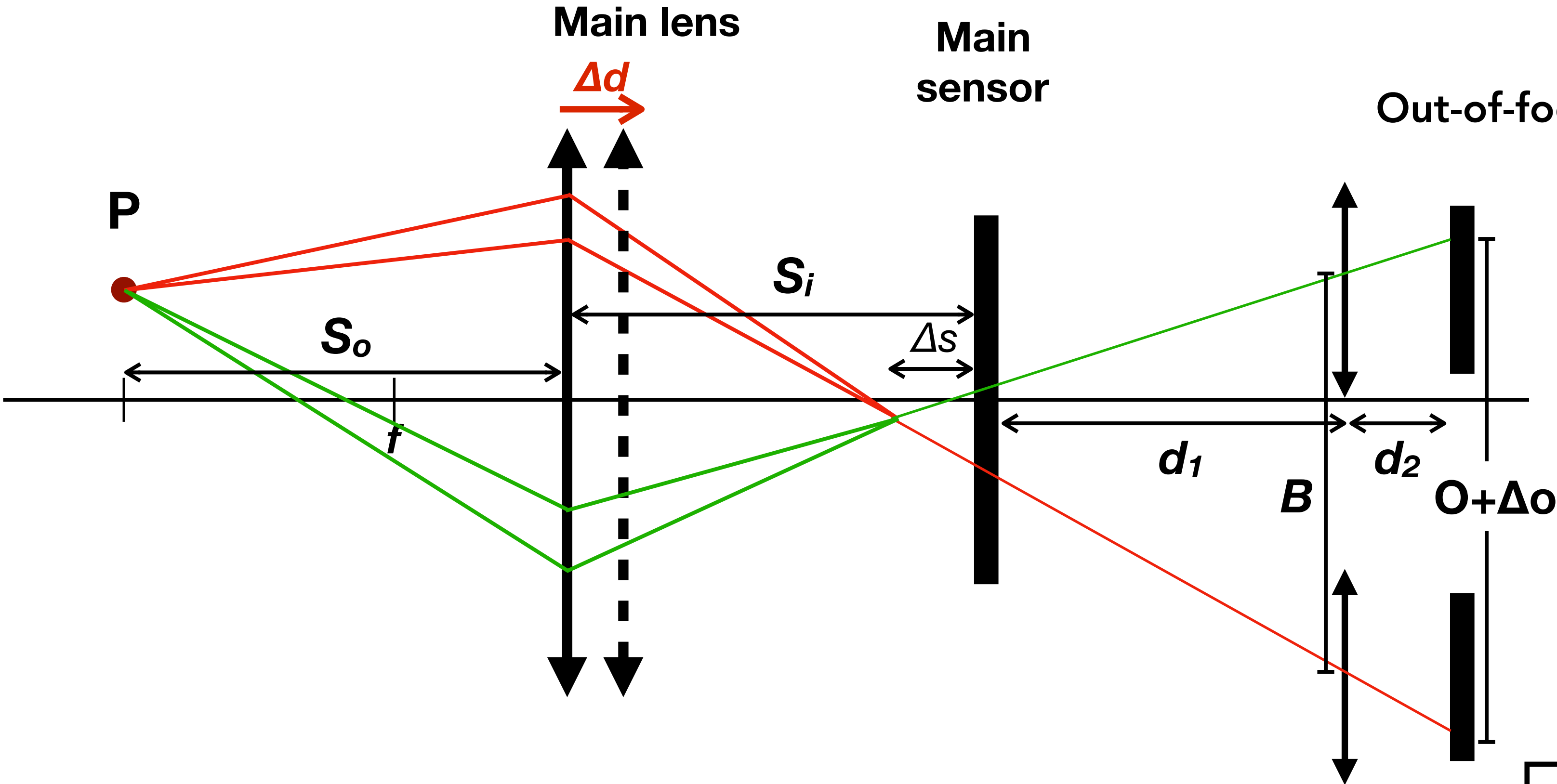
In-focus:  $\frac{d_1}{d_1 + d_2} = \frac{B}{O}$  Phase shift

Out-of-focus:  $\frac{d_1 + \Delta s}{d_1 + \Delta s + d_2} = \frac{B}{O + \Delta o}$

$$\frac{1}{S_o} + \frac{1}{S_i - \Delta s} = \frac{1}{f}$$

$$\frac{1}{S_o + \Delta d} + \frac{1}{S_i - \Delta d} = \frac{1}{f}$$

# Math Behind PDAF

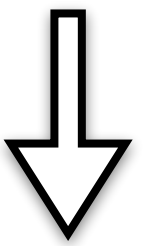


In-focus:  $\frac{d_1}{d_1 + d_2} = \frac{B}{O}$  Phase shift

Out-of-focus:  $\frac{d_1 + \Delta s}{d_1 + \Delta s + d_2} = \frac{B}{O + \Delta o}$

$$\frac{1}{S_o} + \frac{1}{S_i - \Delta s} = \frac{1}{f}$$

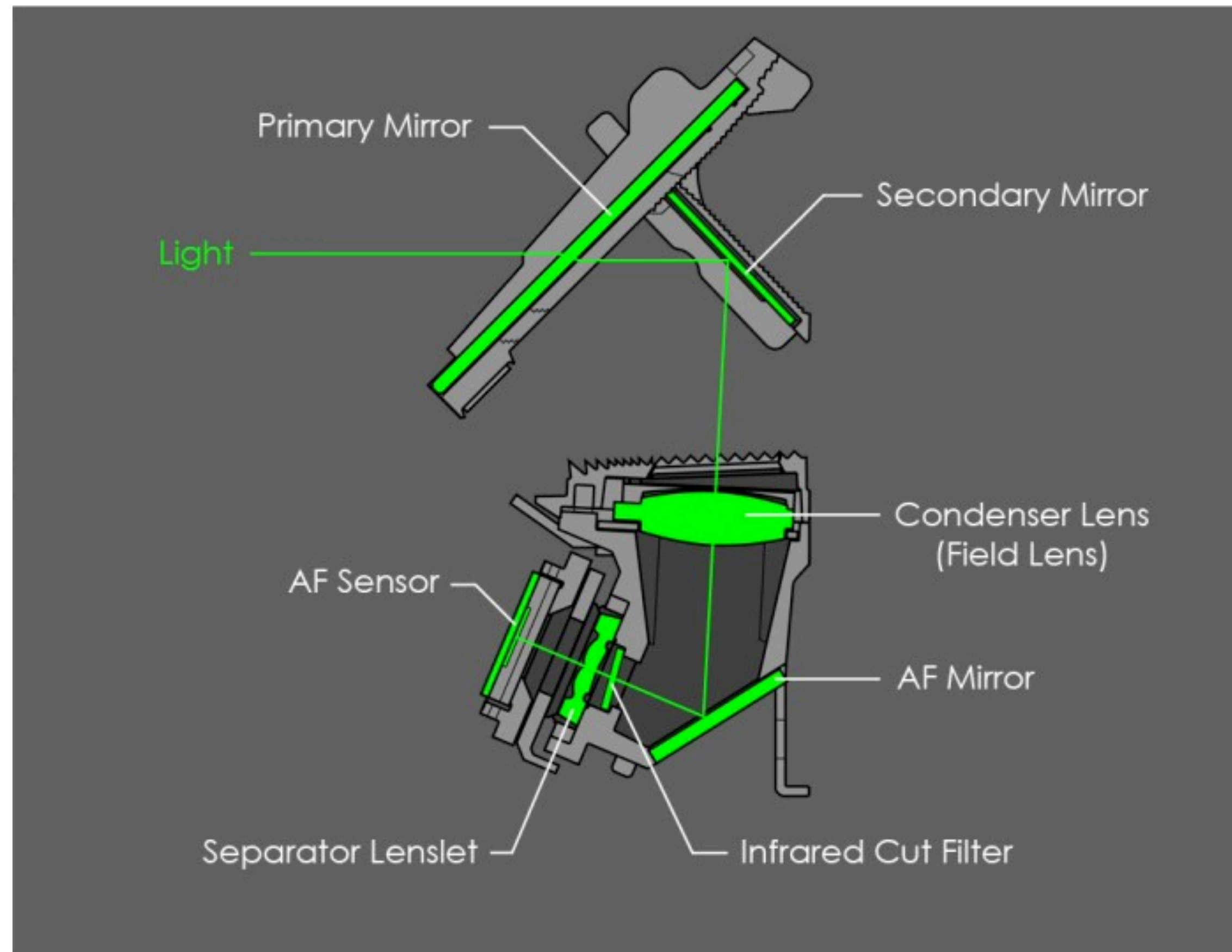
$$\frac{1}{S_o + \Delta d} + \frac{1}{S_i - \Delta d} = \frac{1}{f}$$



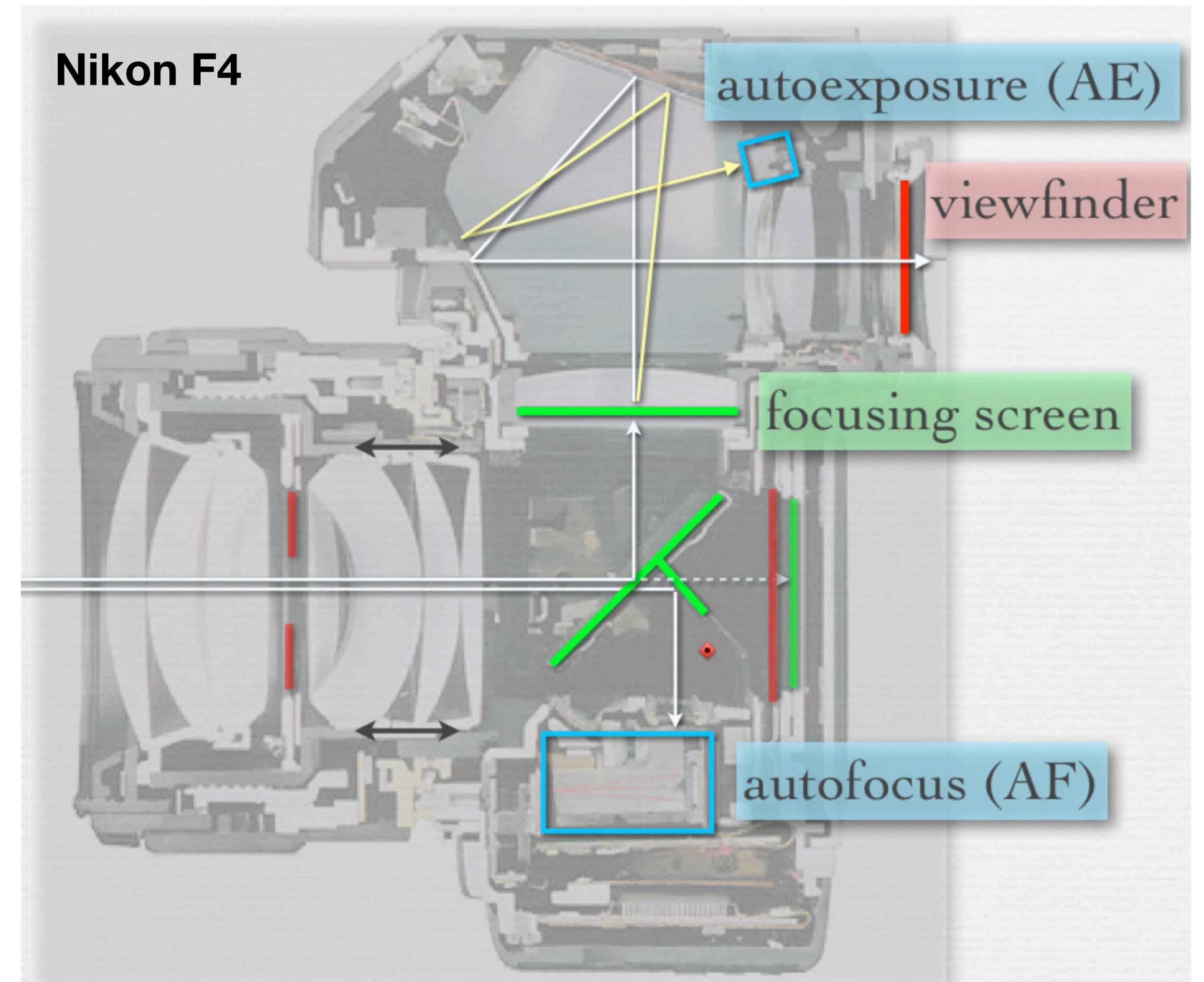
Knowing  $\Delta o$  we can calculate  $\Delta d$ , which is how much the lens needs to be moved. The **sign** of  $\Delta d$  dictates whether to move the lens away or toward the sensor.



# Actual Implementations

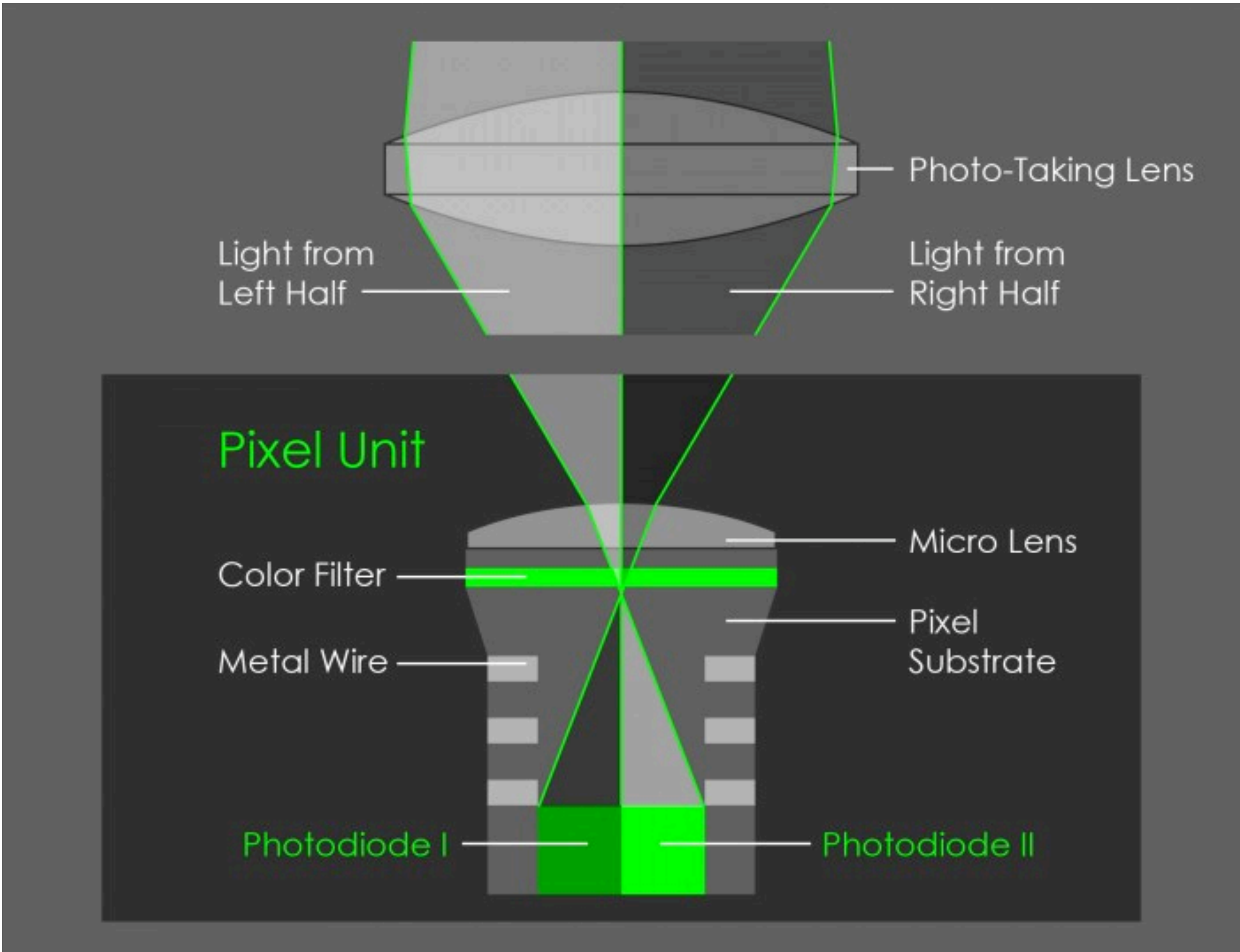
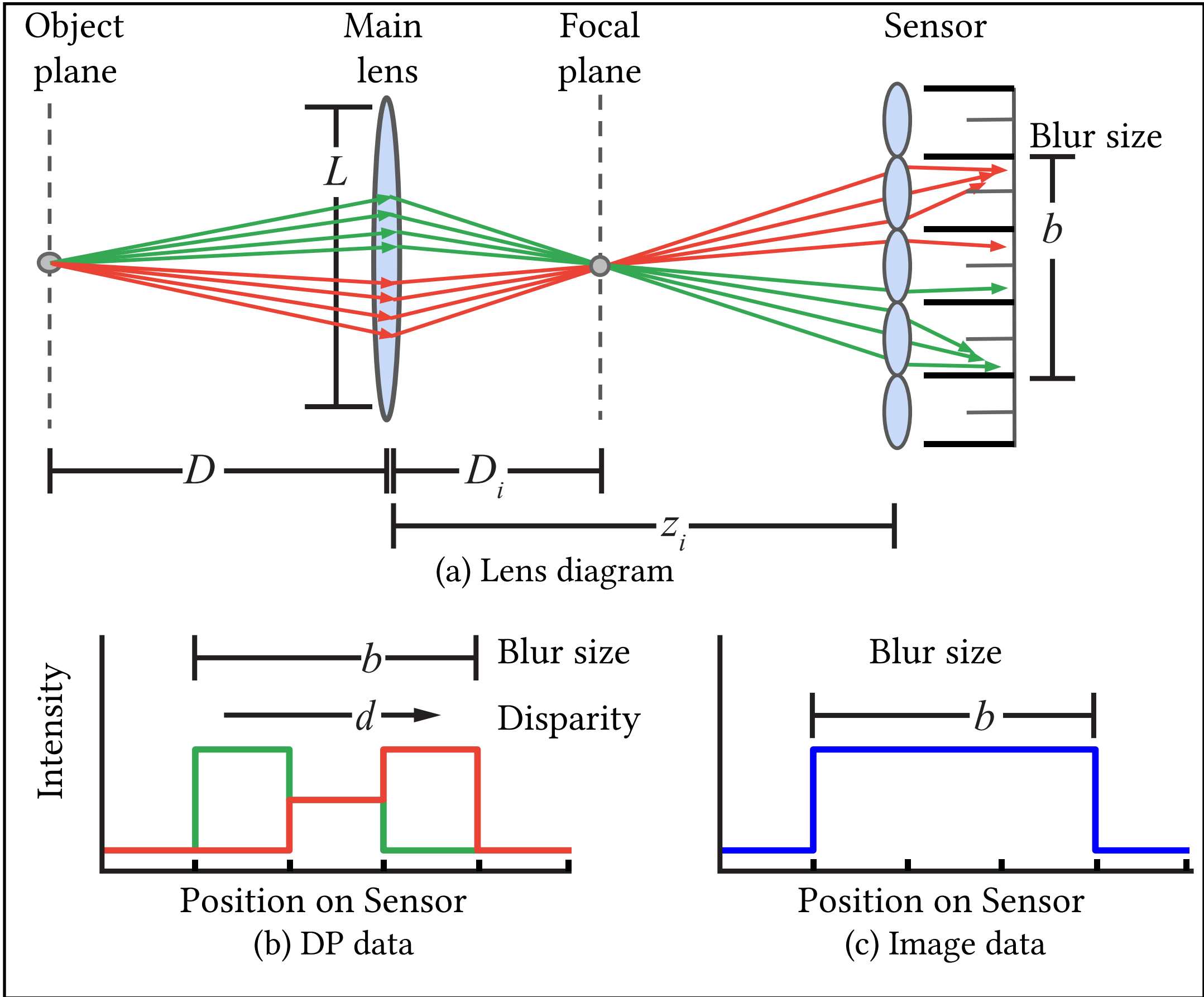


The secondary mirror deflects light from the photographic lens to the phase detection sensor array.



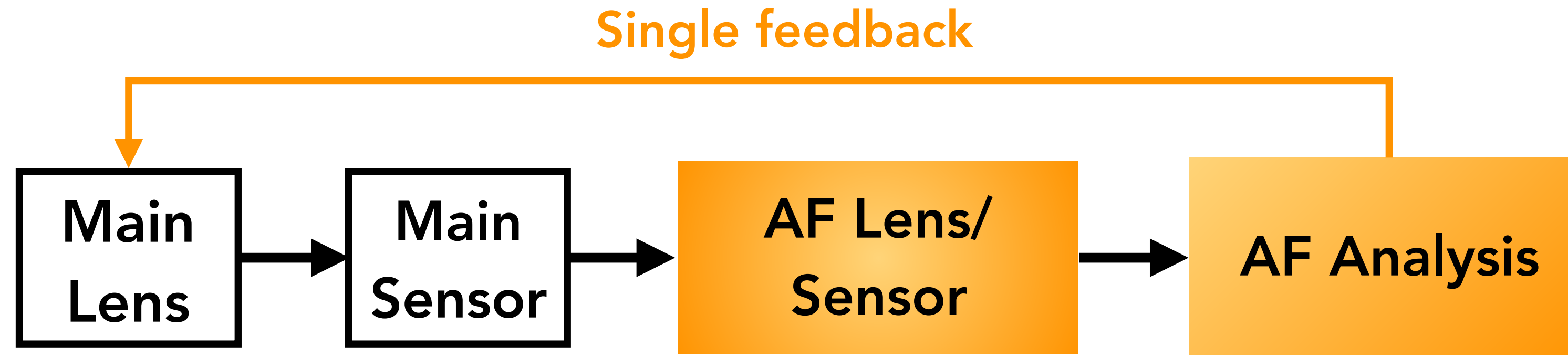


# Dual Pixel PDAF





# PDAF Recap



## Pros:

- Fast. Jump to in-focus position. No hunting for the correct lens adjustment. From one single phase shift, we can calculate exactly how to adjust the lens.

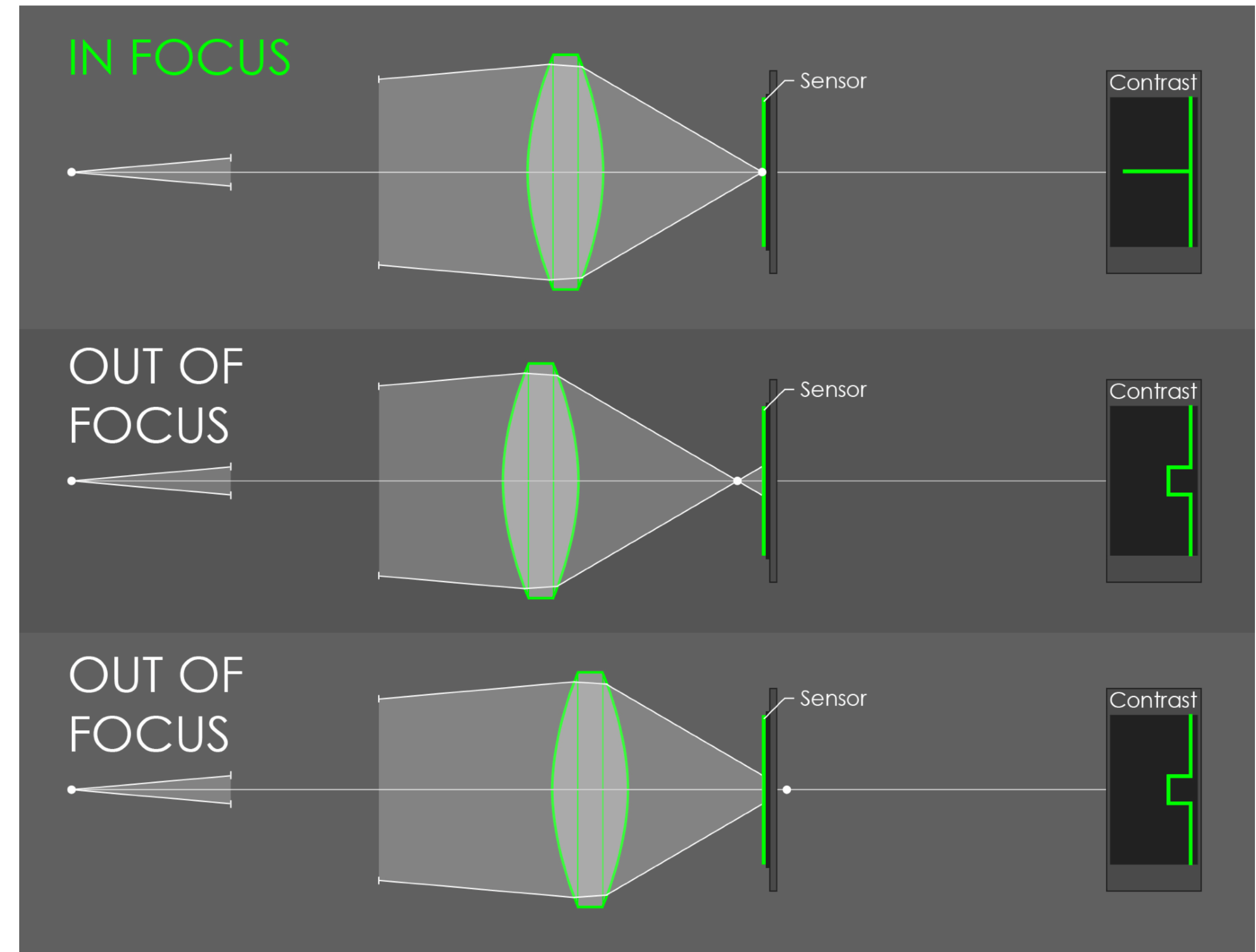
## Cons:

- Requires physical calibration (AF sensor baseline, etc.)
- AF doesn't work when in live preview mode and when actually taking the picture, since the reflex mirror is up, and so AF sensors receive no light.

# Contrast Detection AF

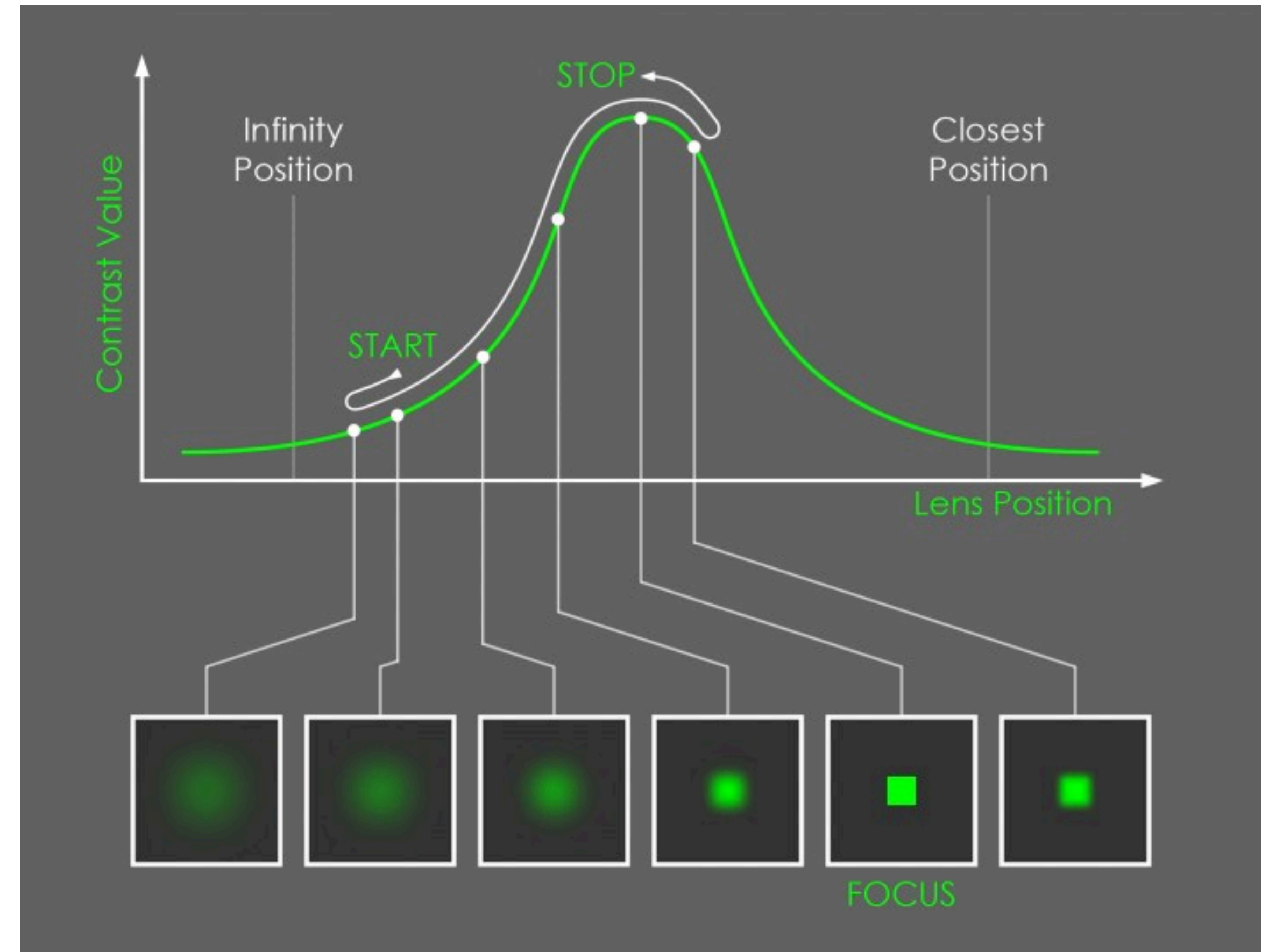
Idea: in-focus objects are sharp, so adjust the lens until the object is sharp.

Very simple design (no AF lens/sensors/mirrors). Used by point-and-shot and smartphone cameras.





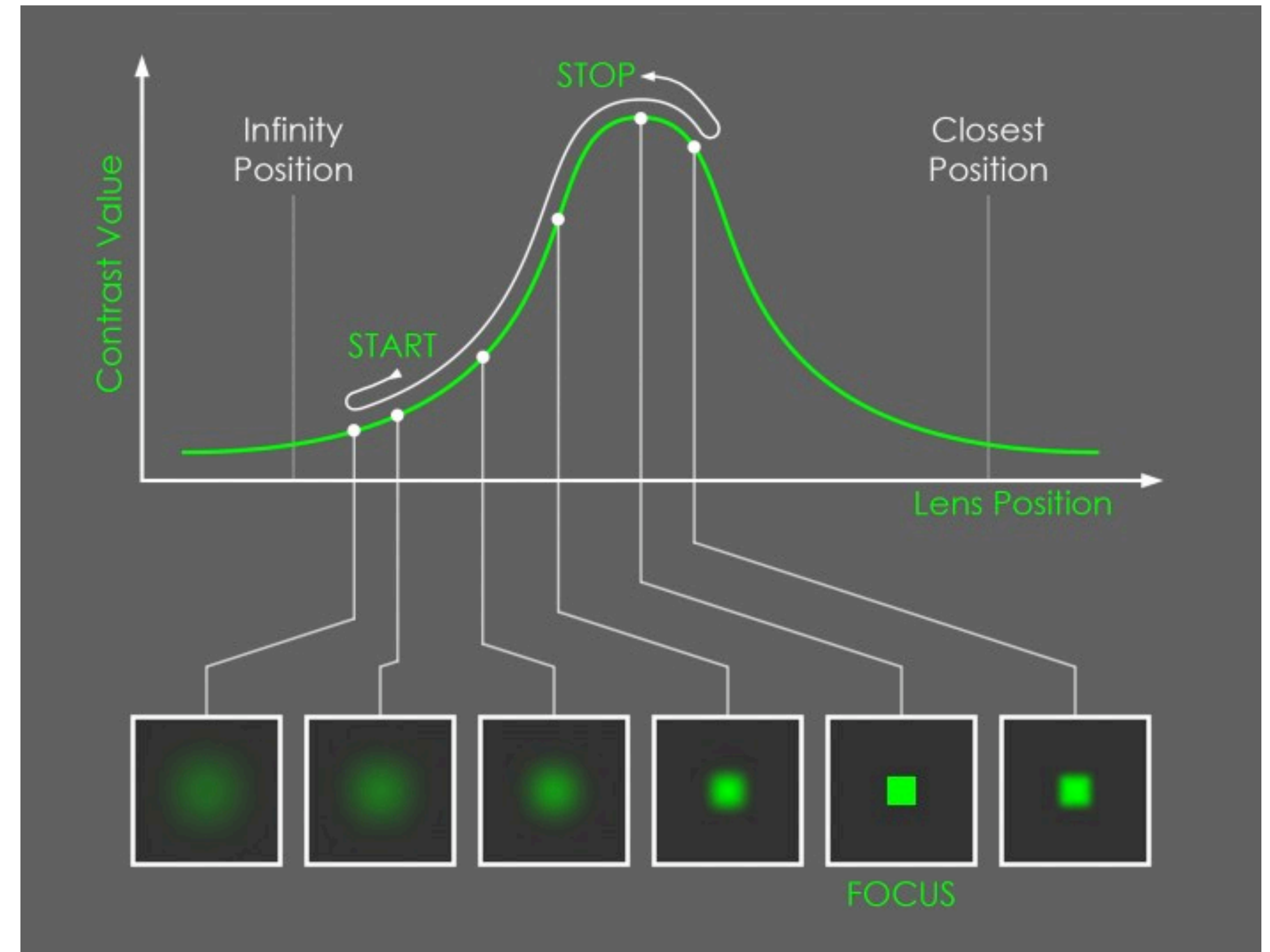
# Detecting Contrast is Tricky



# Detecting Contrast is Tricky

Problem 1: can we simply detect contrast using the height of the peak?

- No. The object to be focused could be dark; then the valley should be used.





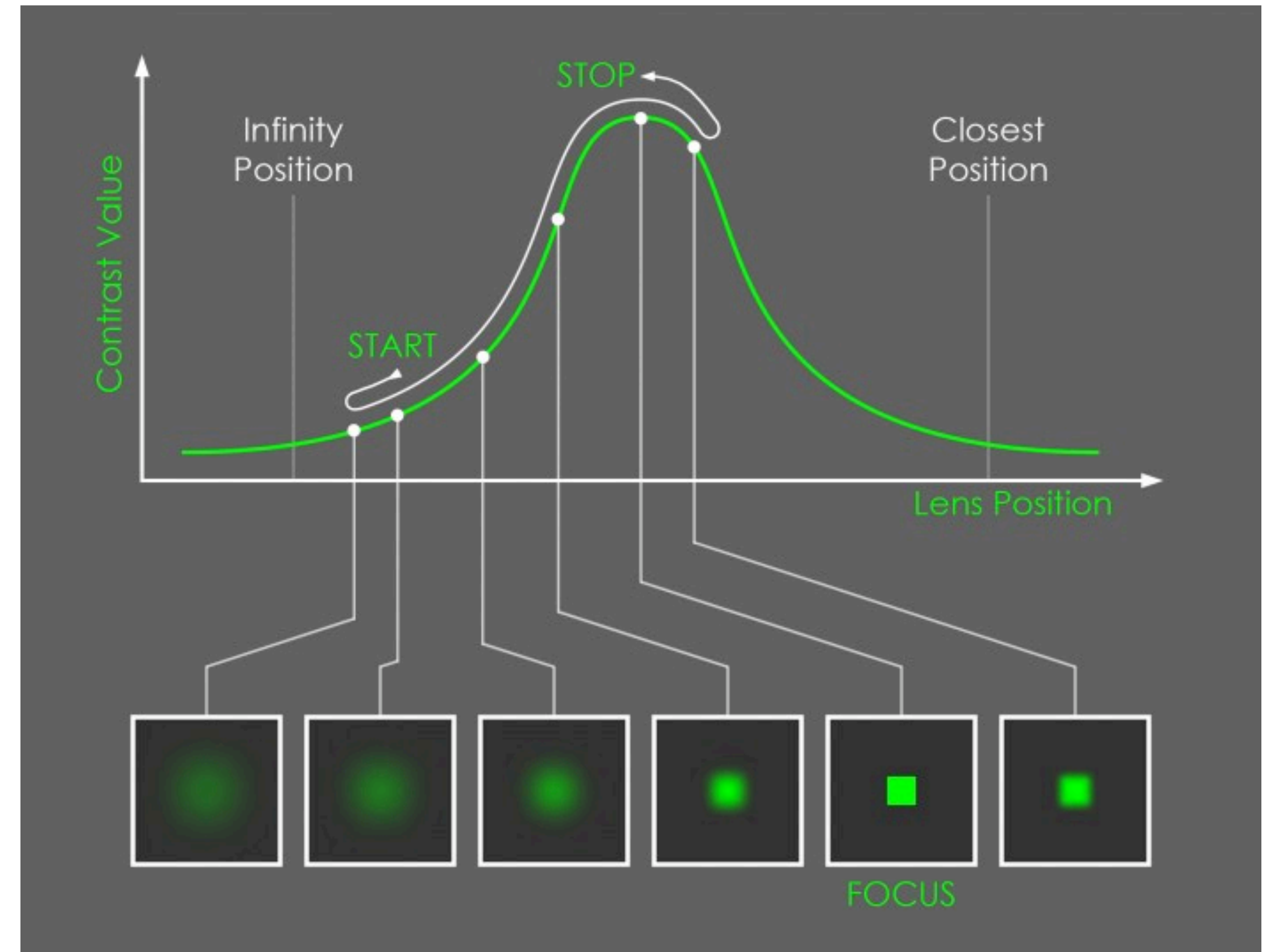
# Detecting Contrast is Tricky

Problem 1: can we simply detect contrast using the height of the peak?

- No. The object to be focused could be dark; then the valley should be used.

Solution: the *gradient* is what we ultimately should care about.

- Calculate the gradient of a pixel by using the pixel values from a small block of neighboring pixels.

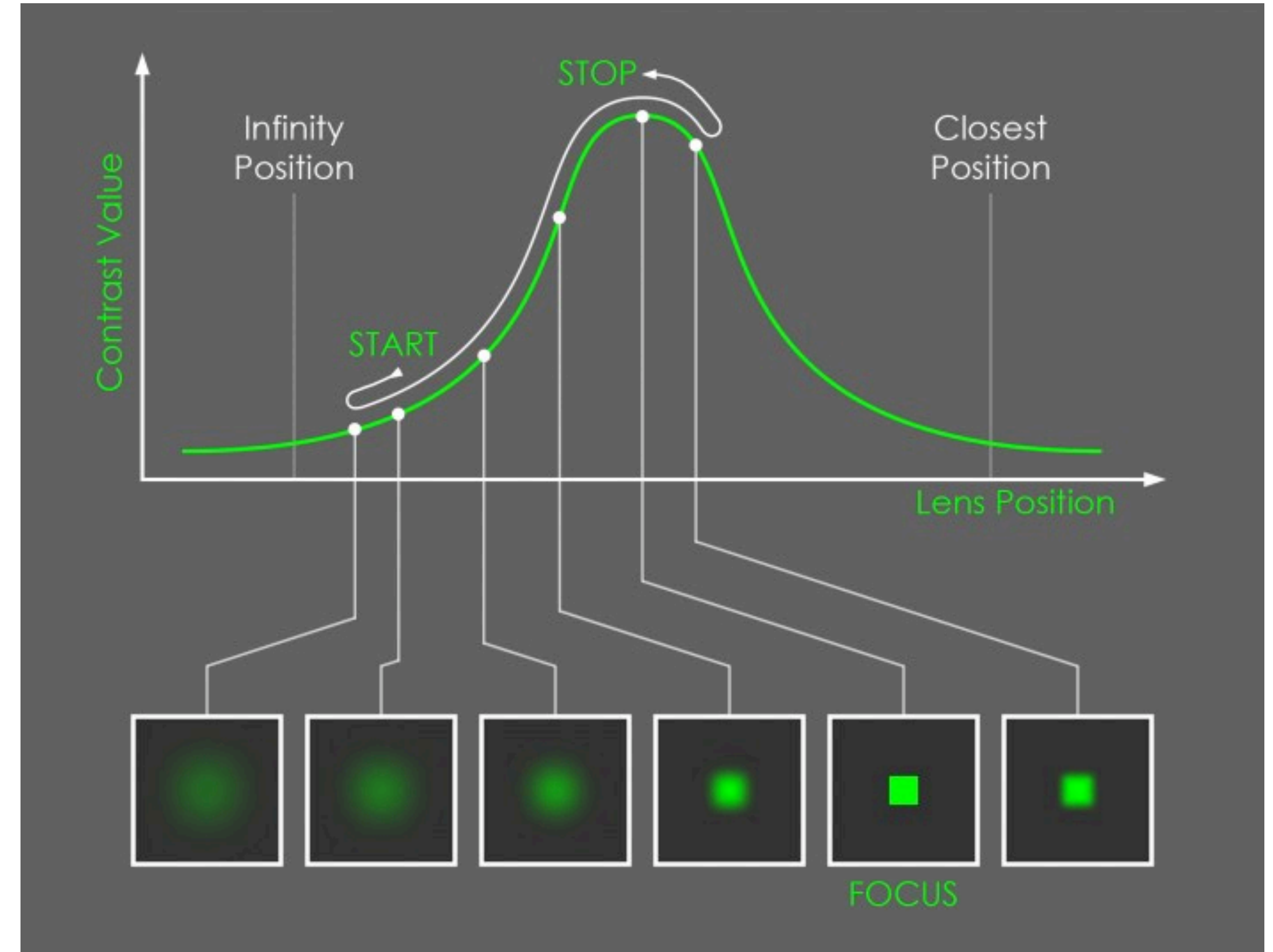
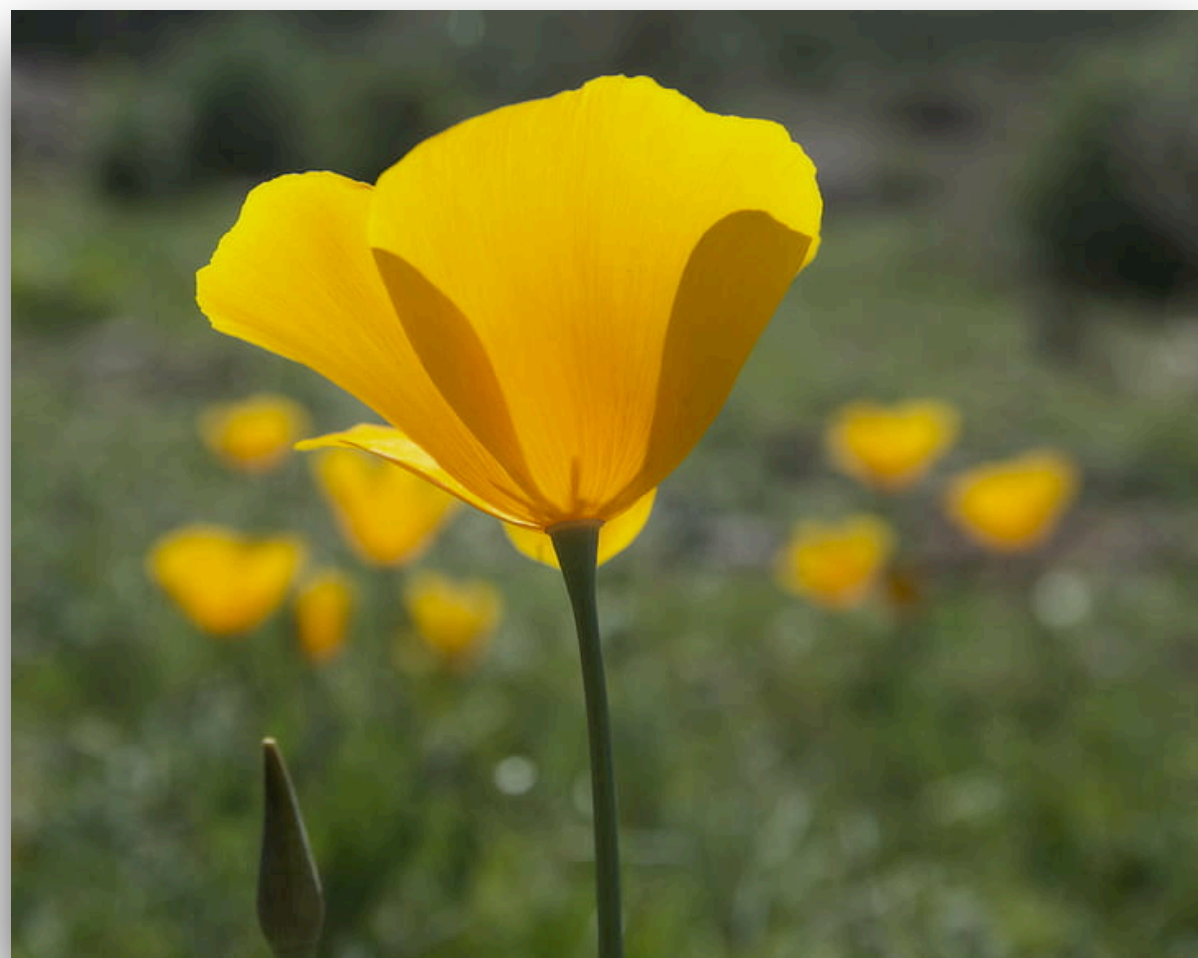


# Detecting Contrast is Tricky

Problem 2: Can we simply use an absolute gradient threshold?

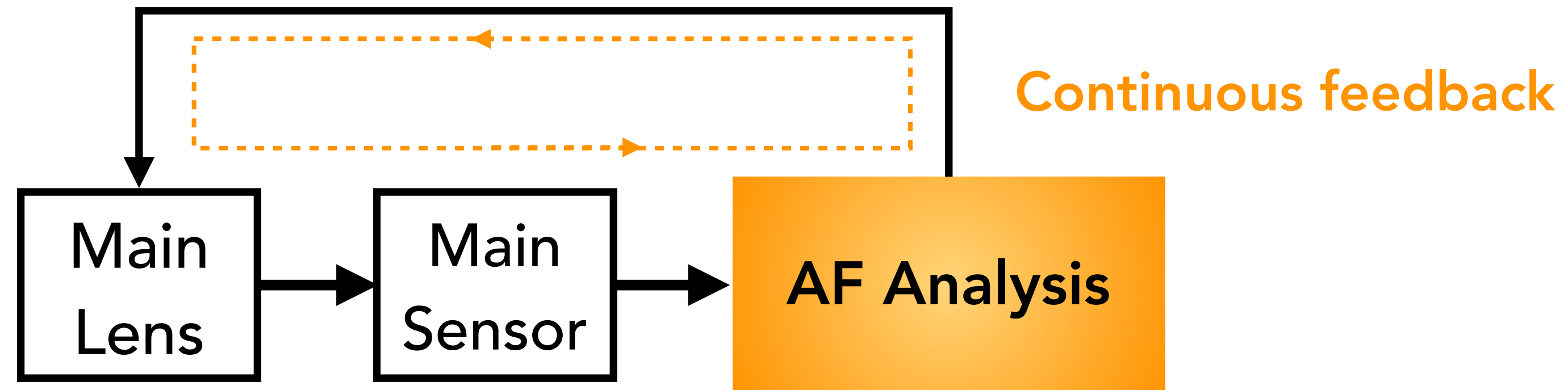
- No. If the object has smooth surface, it's in-focus gradient will be modest.

Solution: trial and error. Slow!





# CDAF Recap



Usually used by point-and-shot cameras.

## Pros:

- Simple and lean hardware design.
- Accurate as it directly operates on the image captured by the main sensor.

## Cons:

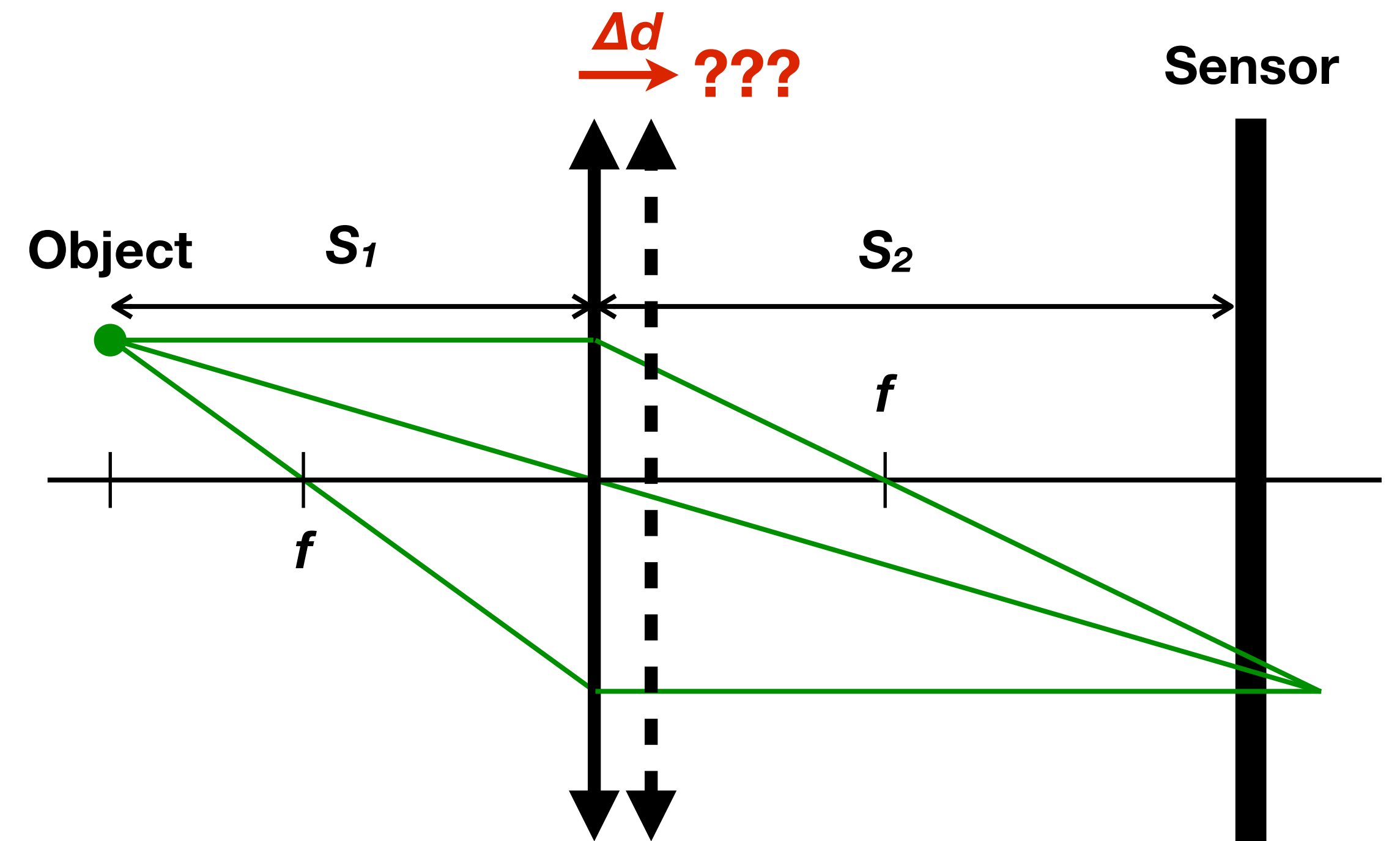
- Slow.

# Active Auto Focus

Active AF directly measures depth. Particularly useful when the scene is textureless, where neither phase nor contrast can be easily detected.

Two main **principles**:

- Time-of-flight
- Depth from stereo



$$\frac{1}{S_1 + \Delta d} + \frac{1}{S_2 - \Delta d} = \frac{1}{f}$$

???

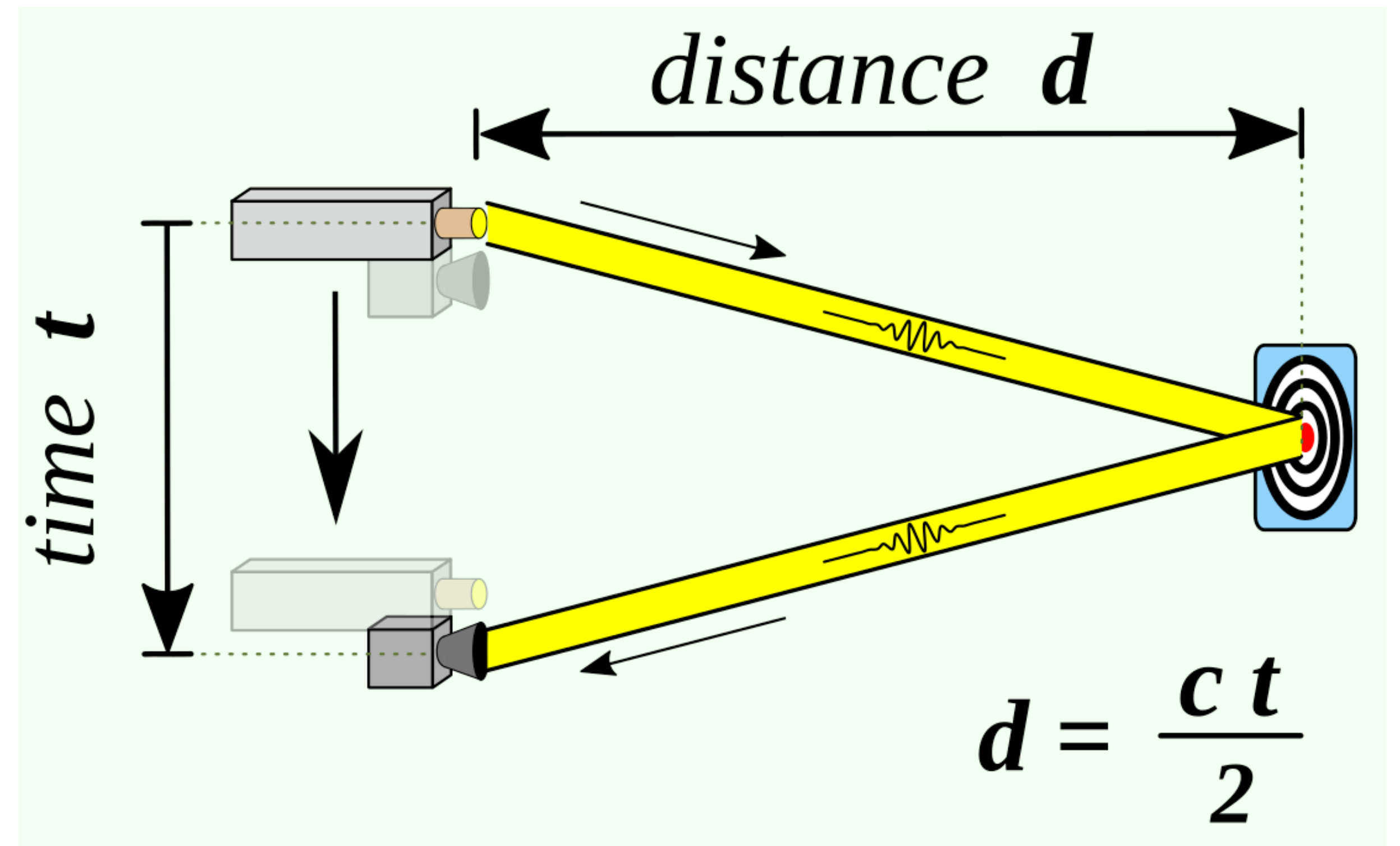
Known from current position

Constant



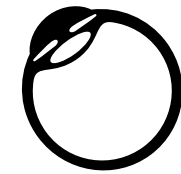
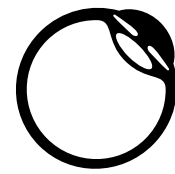
# Time-of-flight Principle

LiDAR: Light Detection and Ranging (Sonar, Infrared light, etc.)



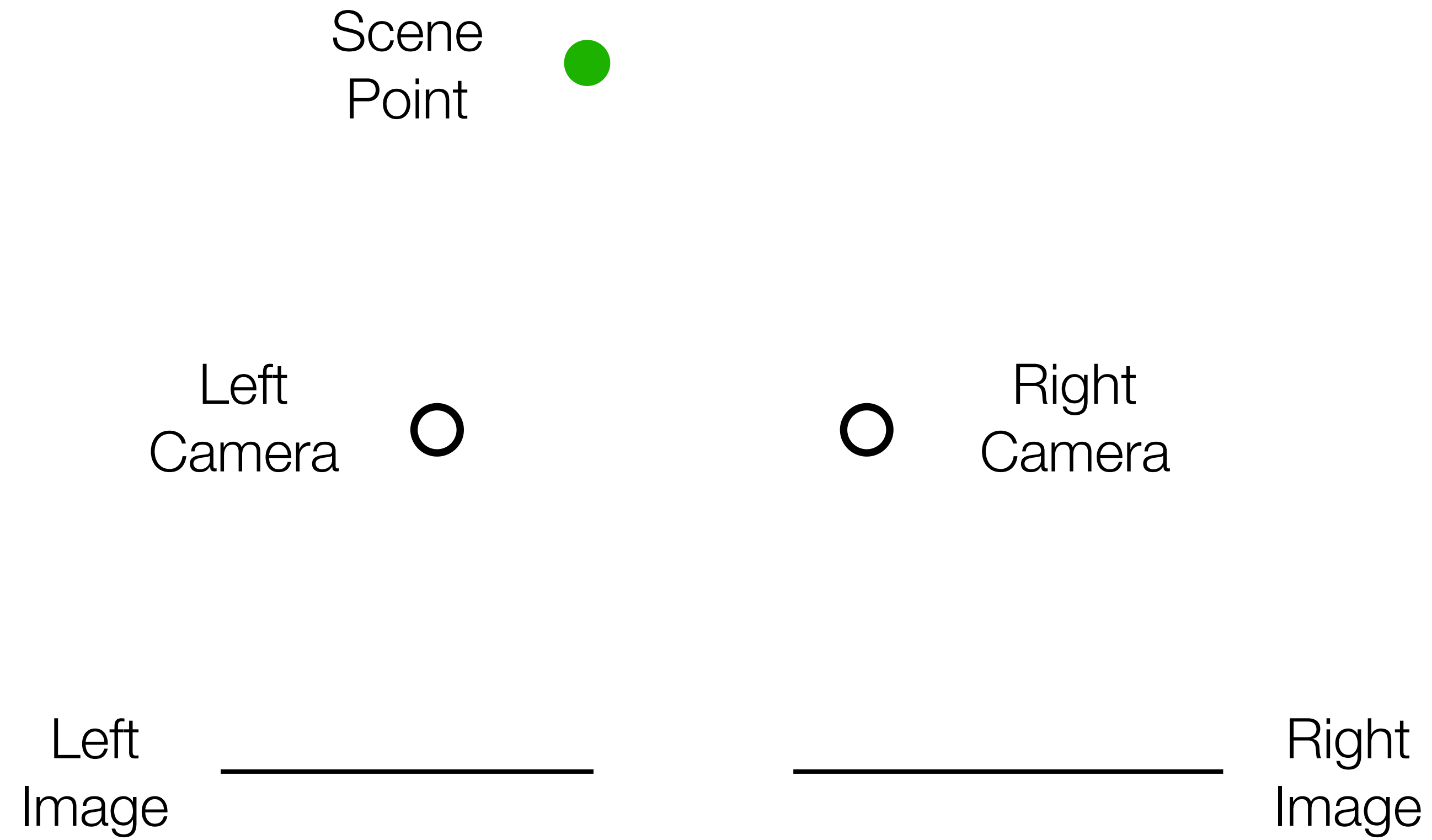
# Depth from Stereo (Triangulation)

Scene  
Point

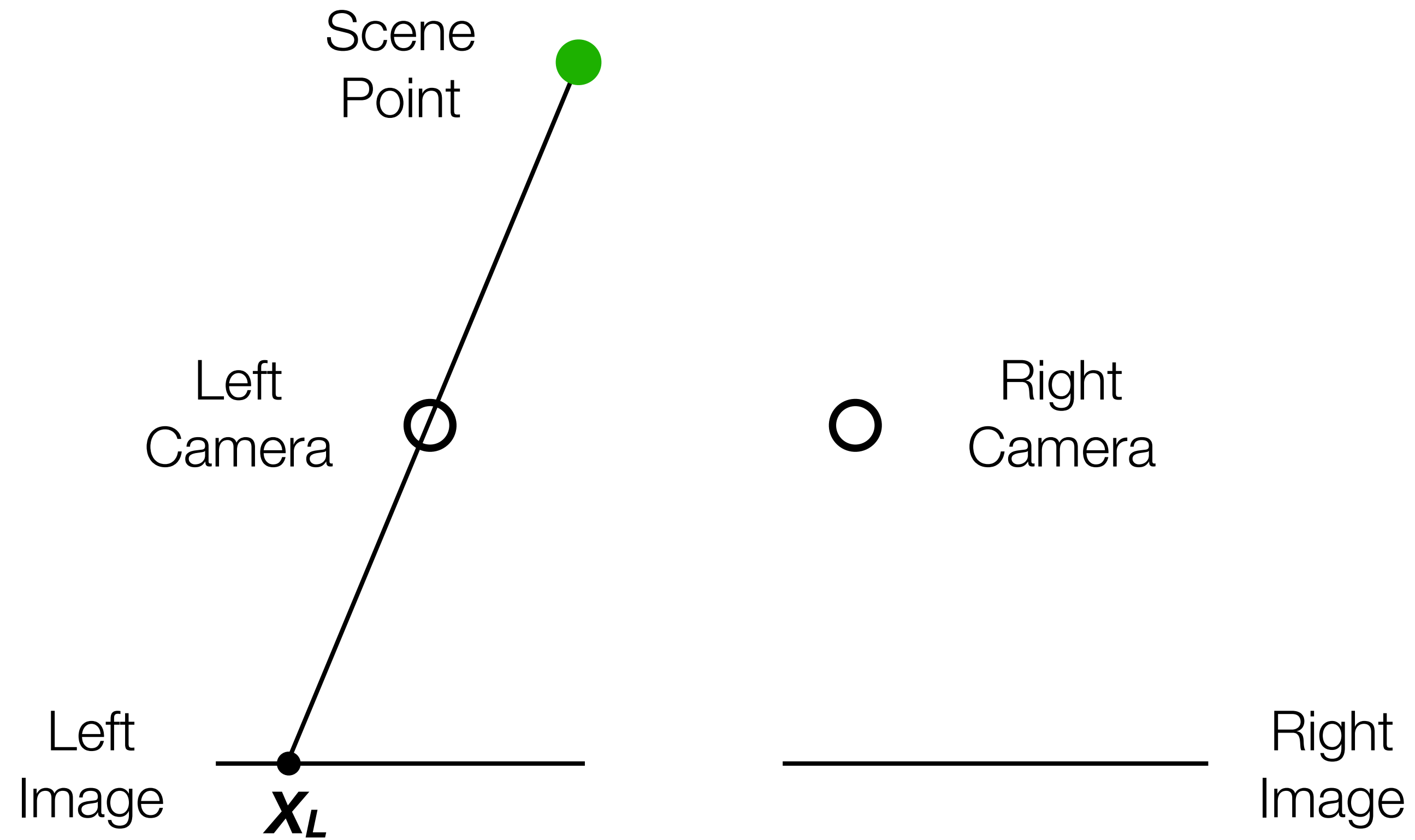




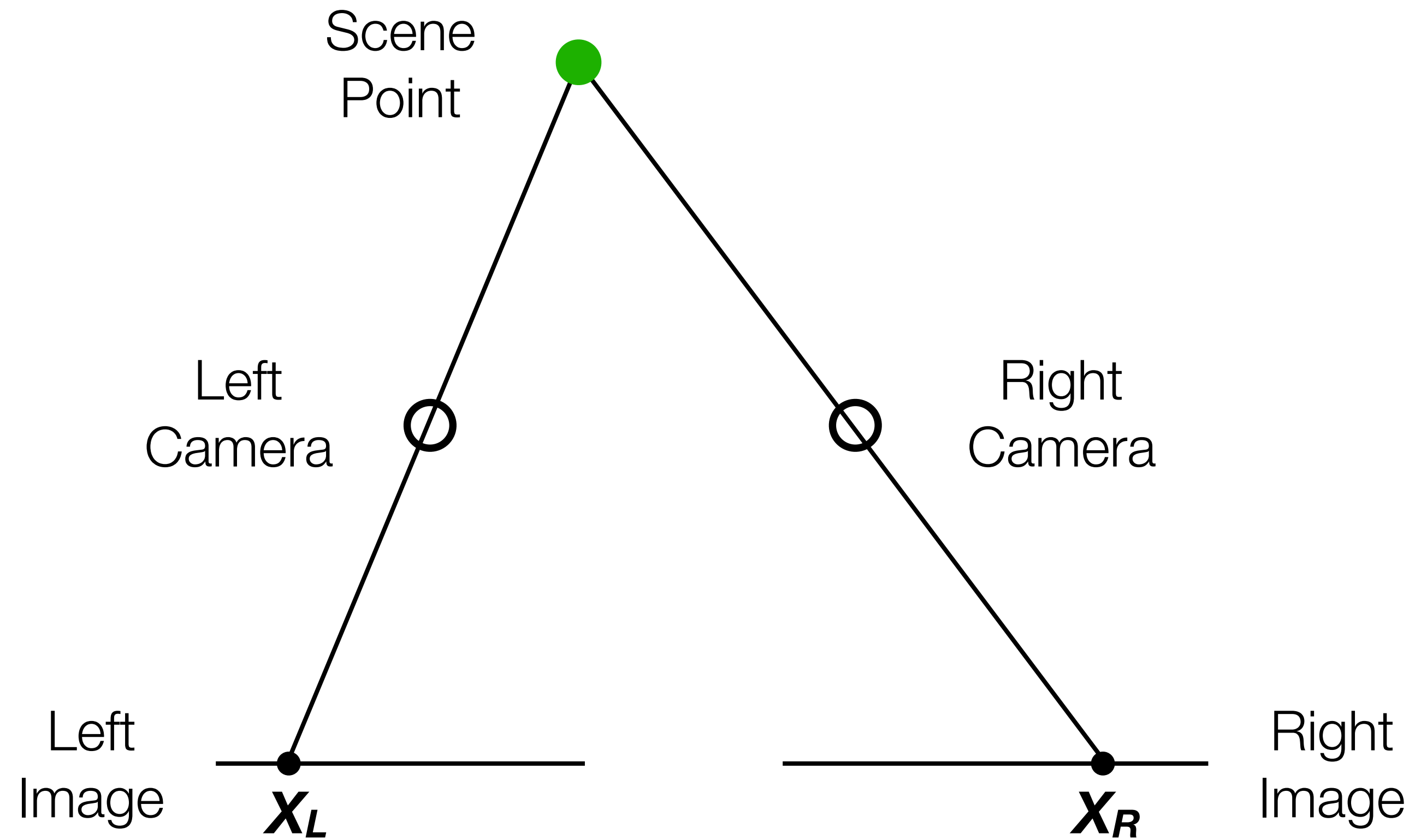
# Depth from Stereo (Triangulation)



# Depth from Stereo (Triangulation)

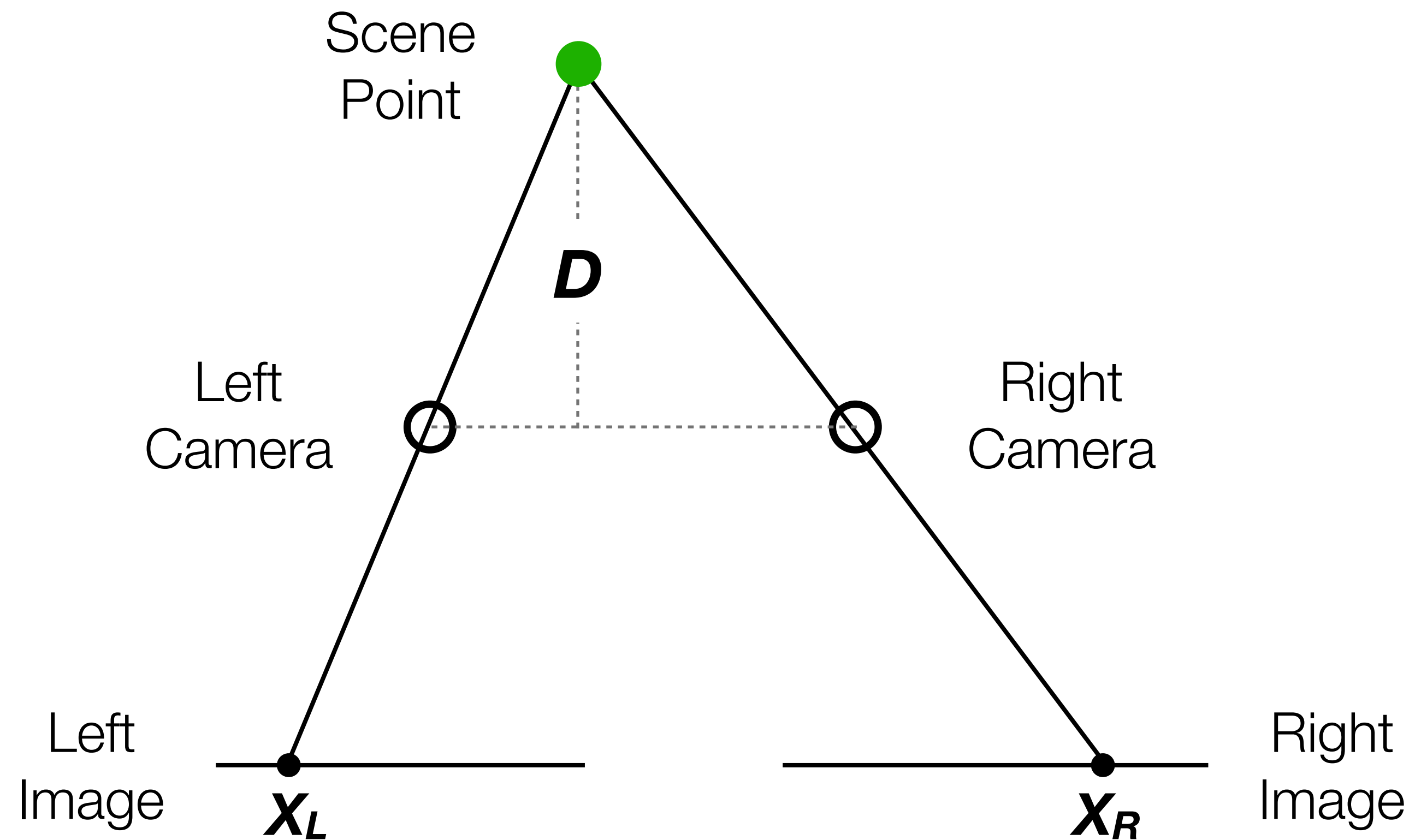


# Depth from Stereo (Triangulation)

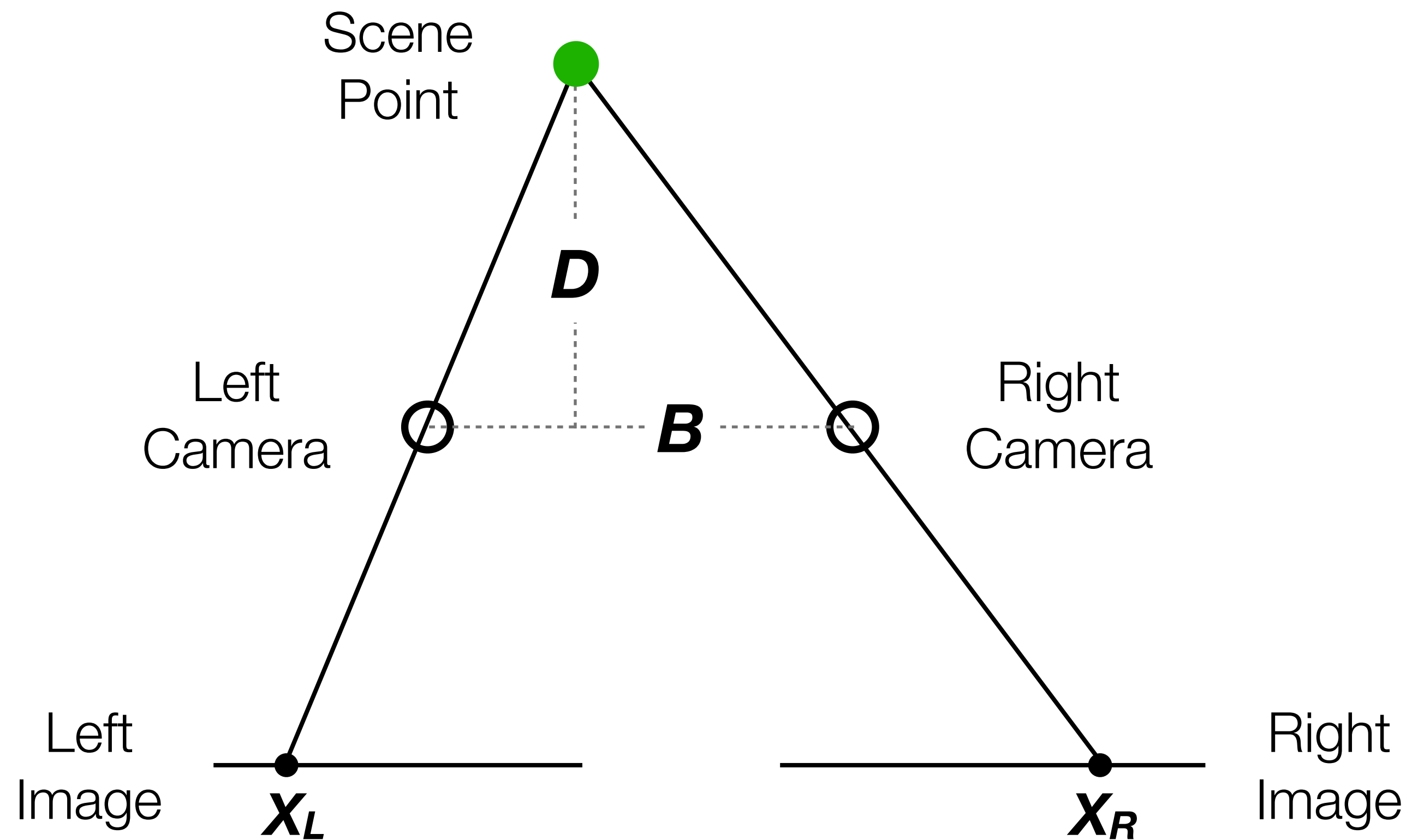




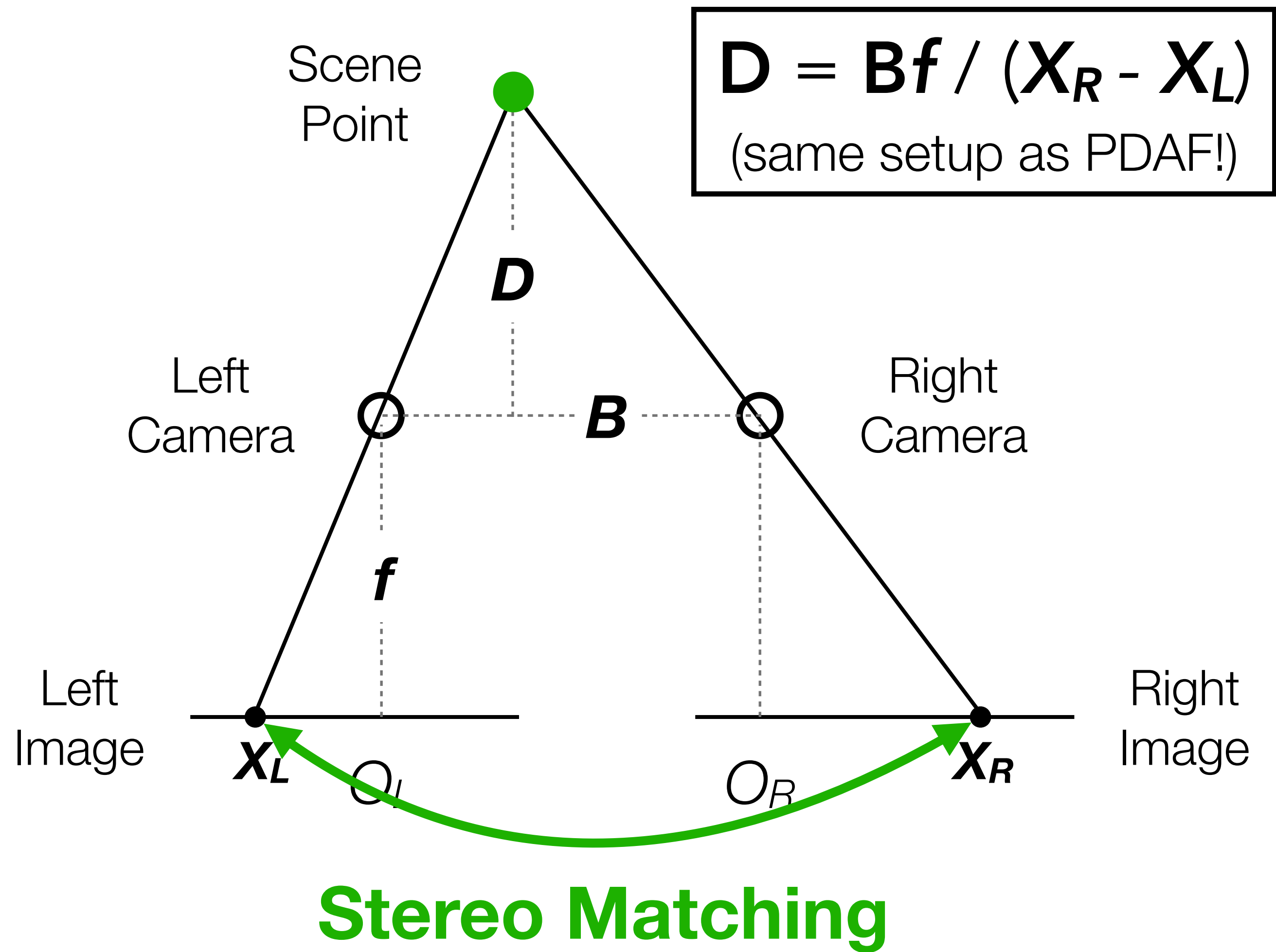
# Depth from Stereo (Triangulation)



# Depth from Stereo (Triangulation)

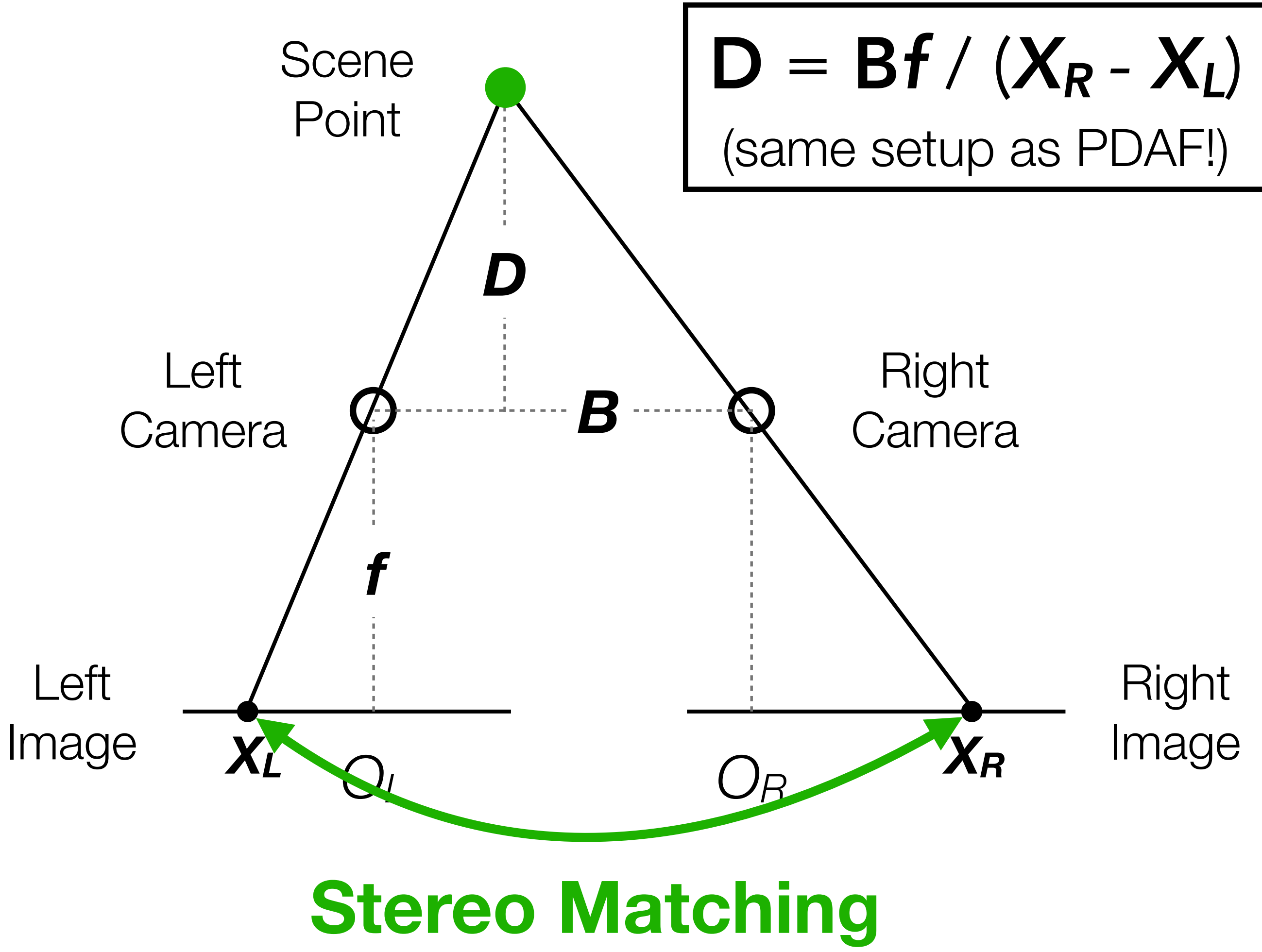


# Depth from Stereo (Triangulation)





# Depth from Stereo (Triangulation)



Left



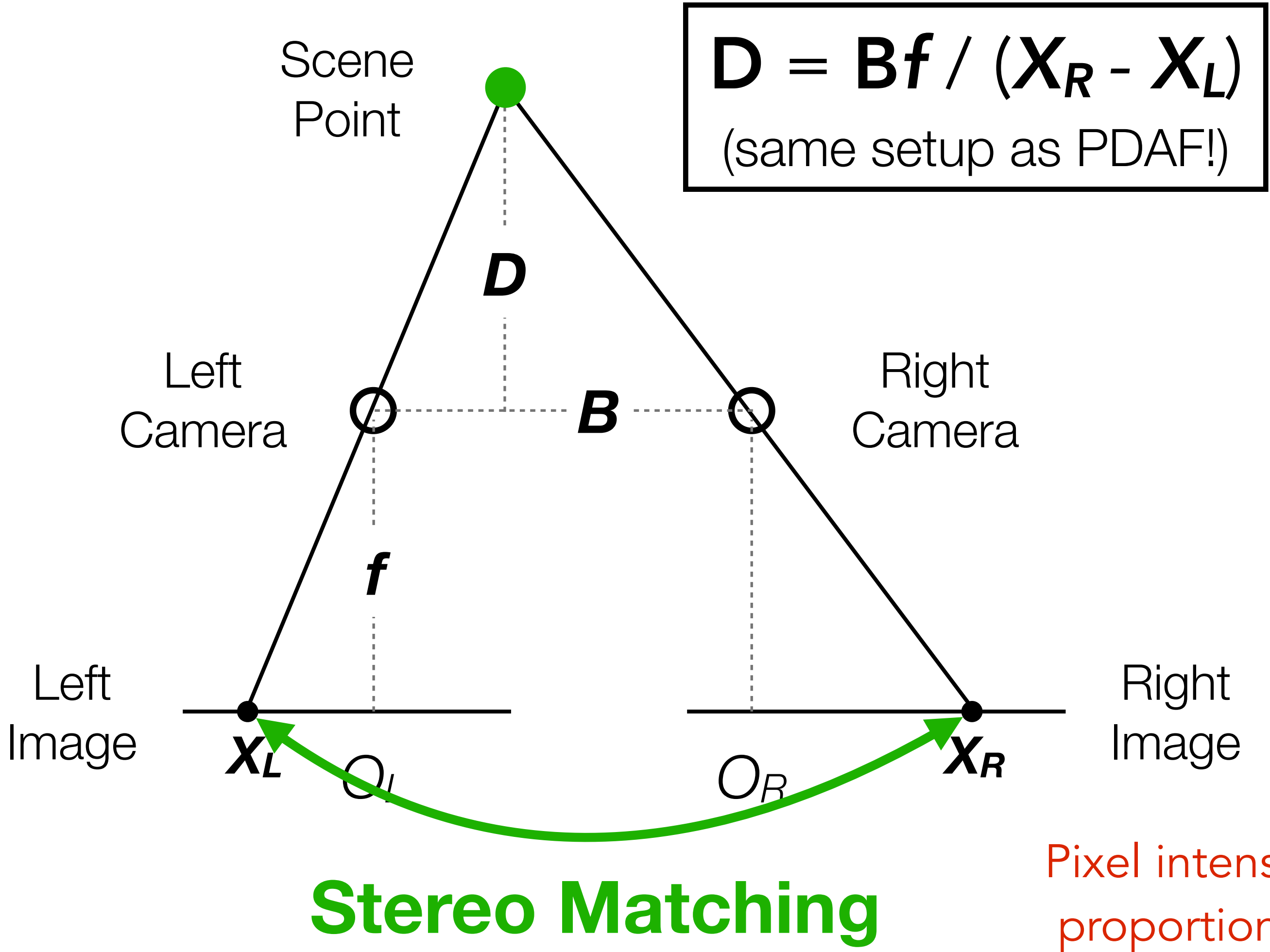
Right



Depth map  
(for left)



# Depth from Stereo (Triangulation)



Left



Right



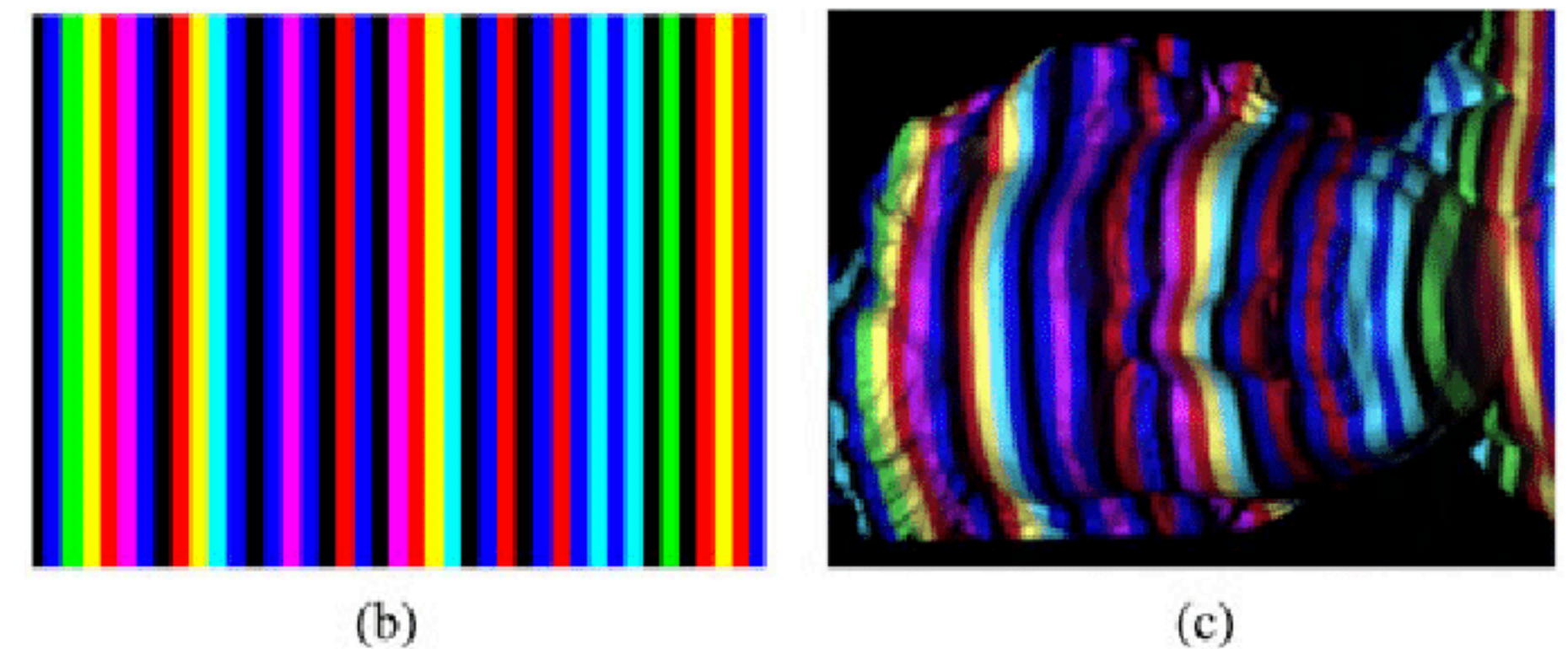
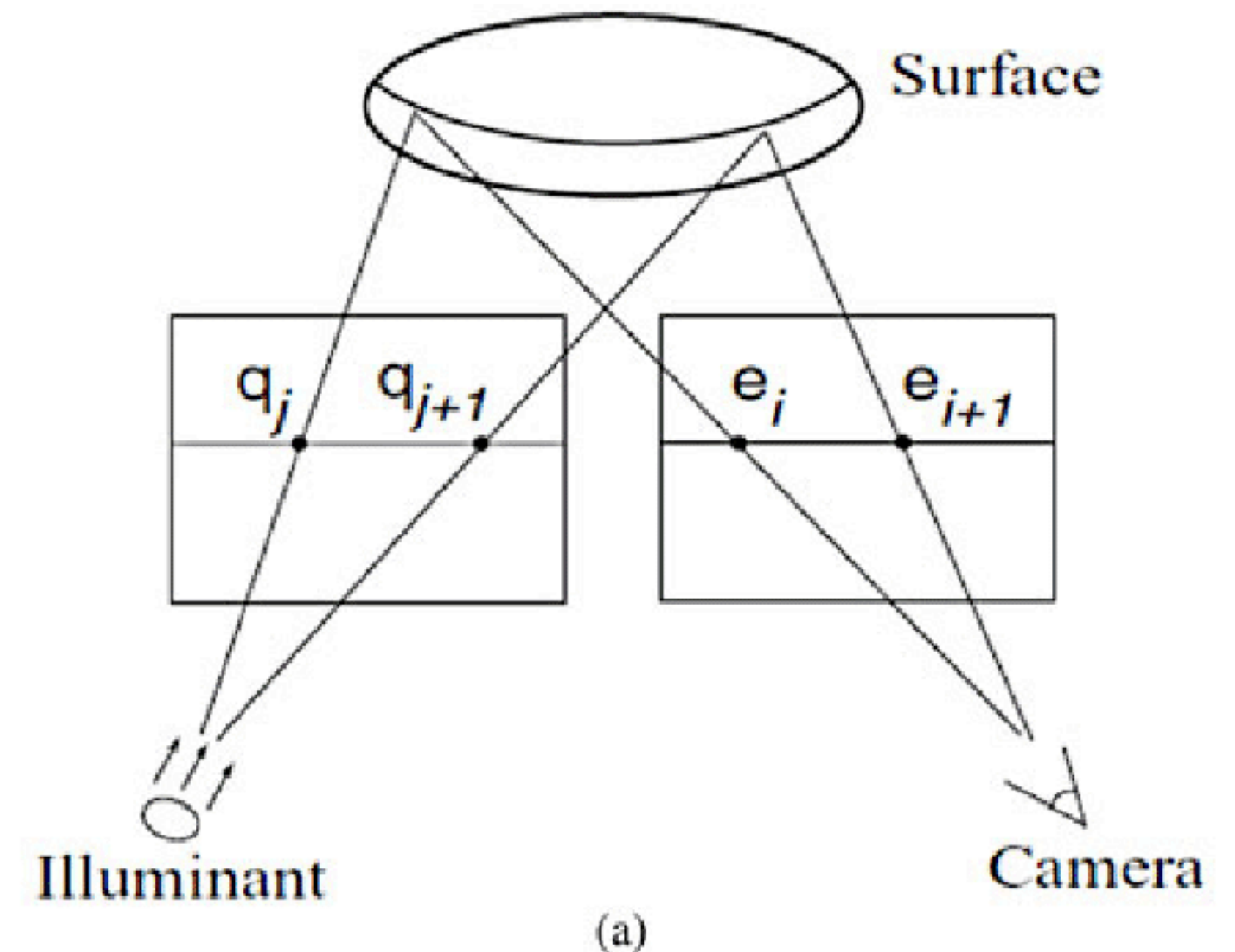
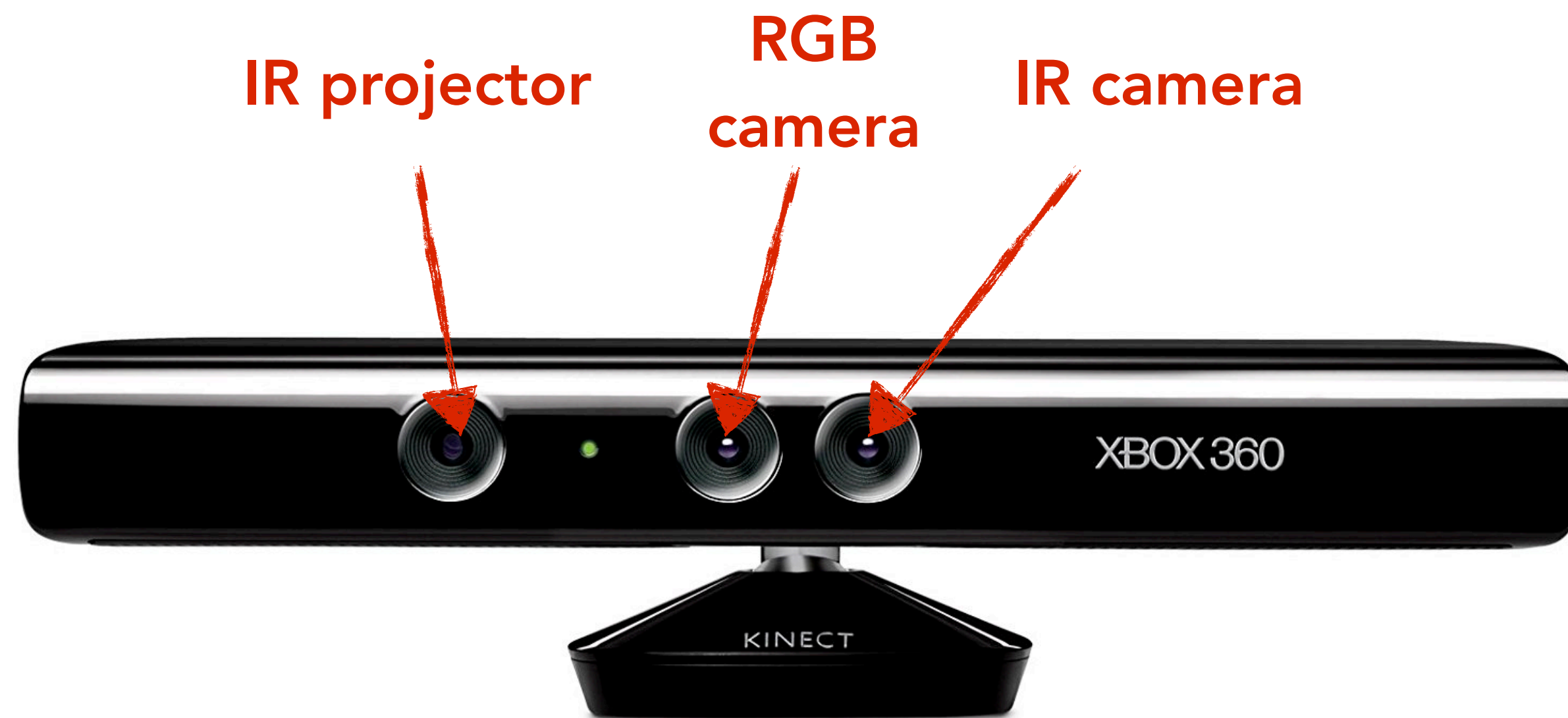
Depth map (for left)

Pixel intensity proportional to depth



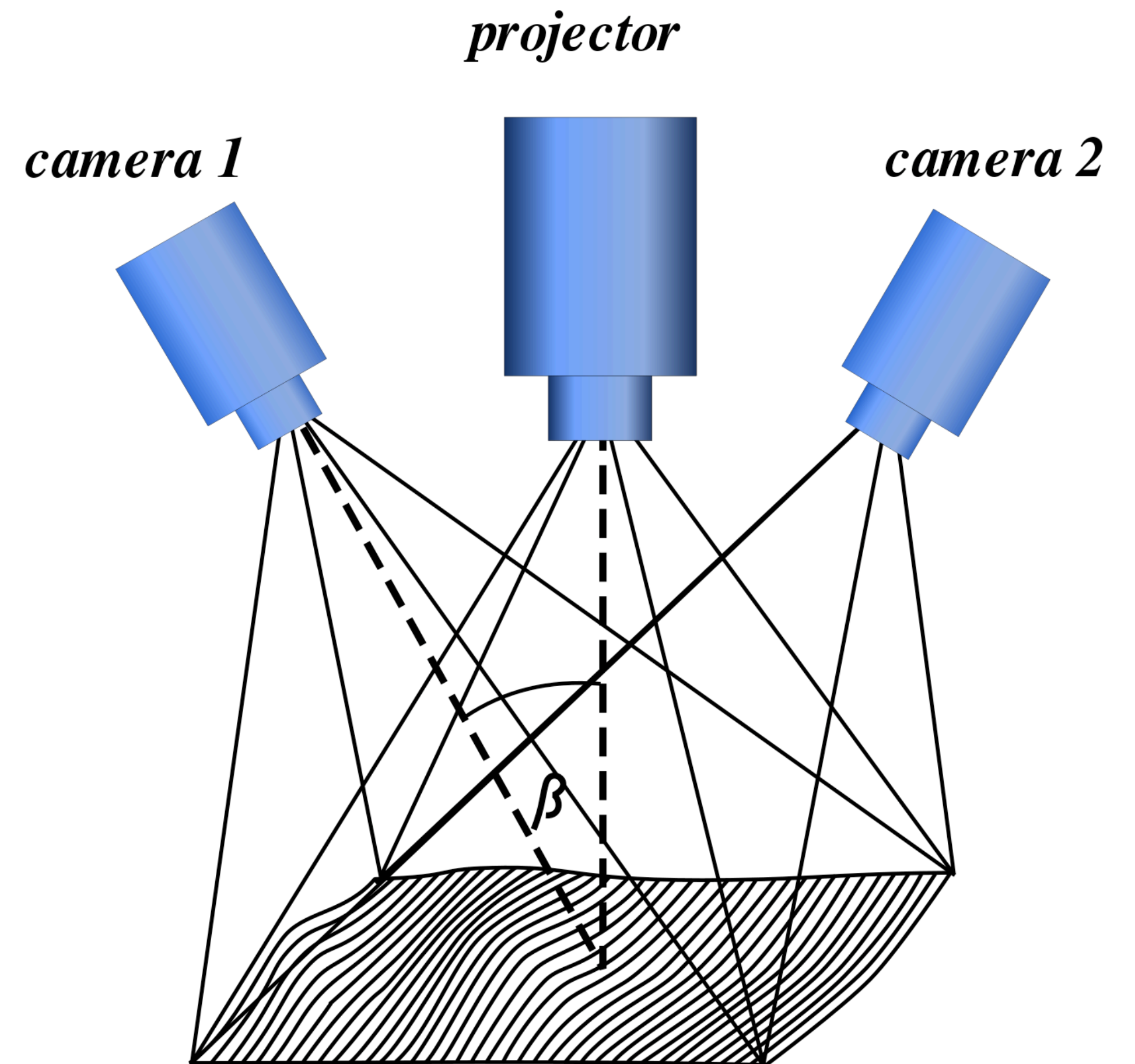
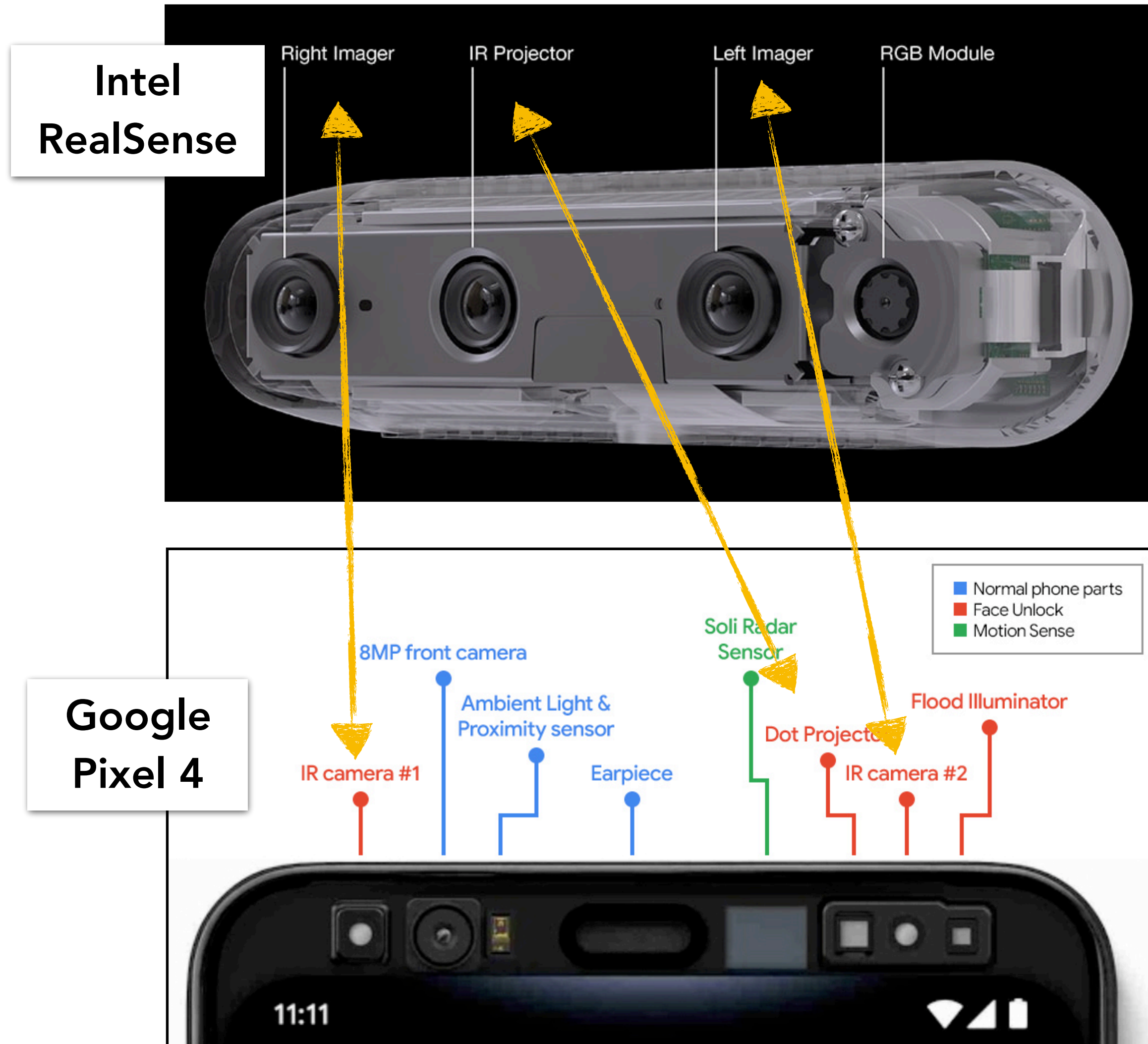
# Structured Light

Correspondence is hard to establish if the surface is textureless. Structured light solves it by emitting lights with fixed pattern, creating textures.





# Another Setup of Structured Light



<https://ai.googleblog.com/2020/04/udepth-real-time-3d-depth-sensing-on.html>

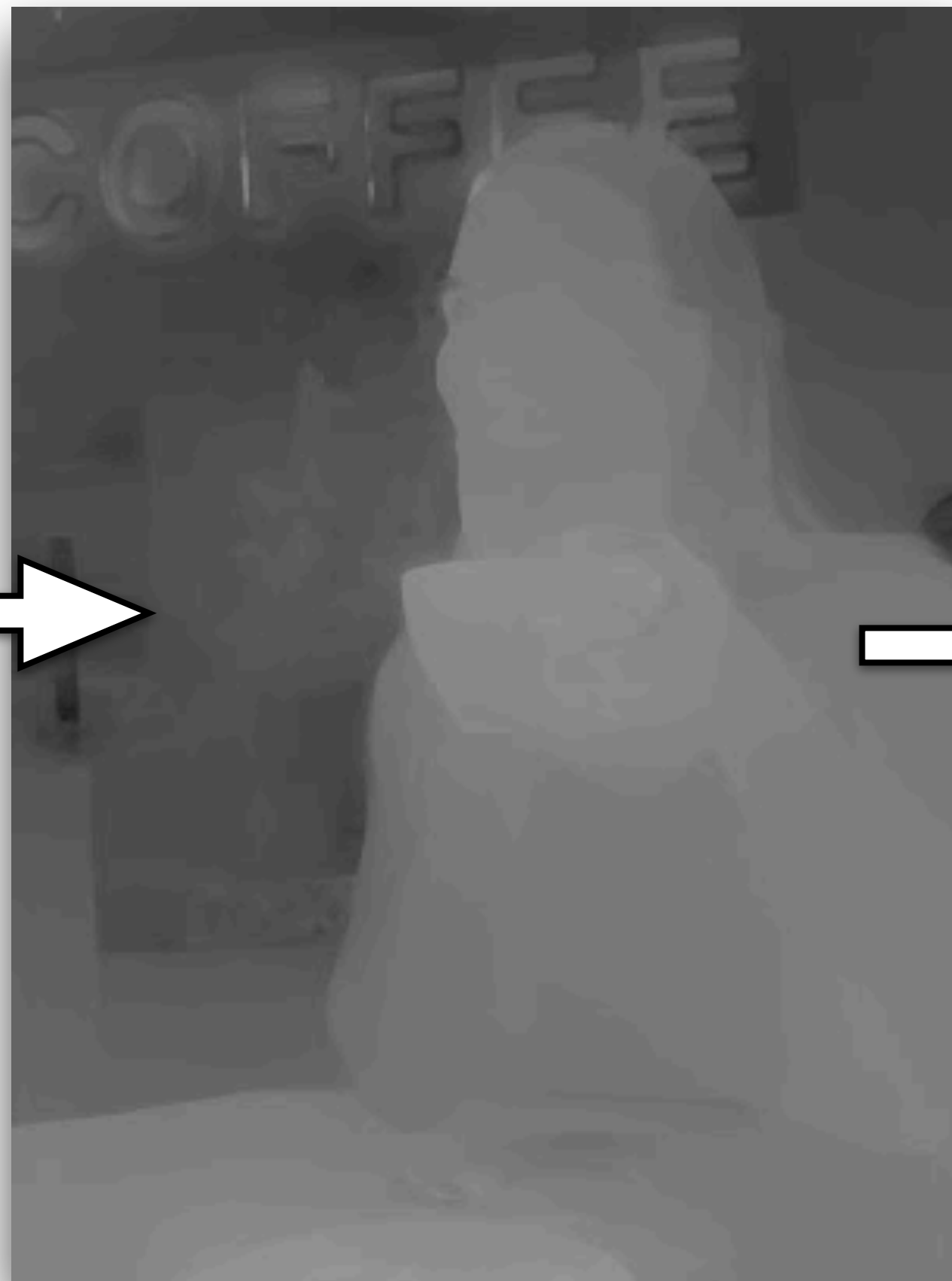


# Synthetic Depth of Field

Shot an all-in-focus photo



Estimate pixel depths

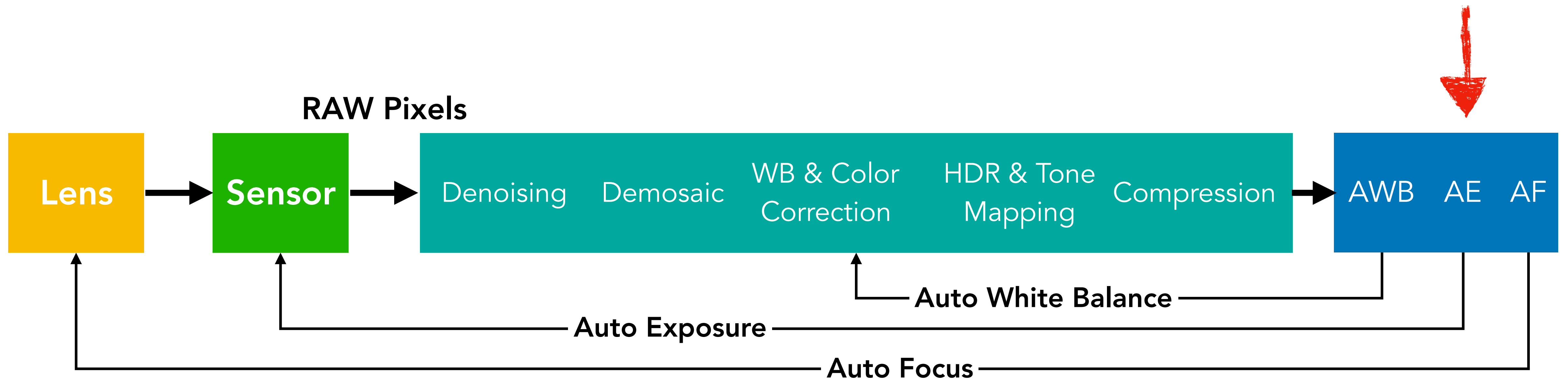


Blur background pixels



Google Pixel 2

# Auto Exposure





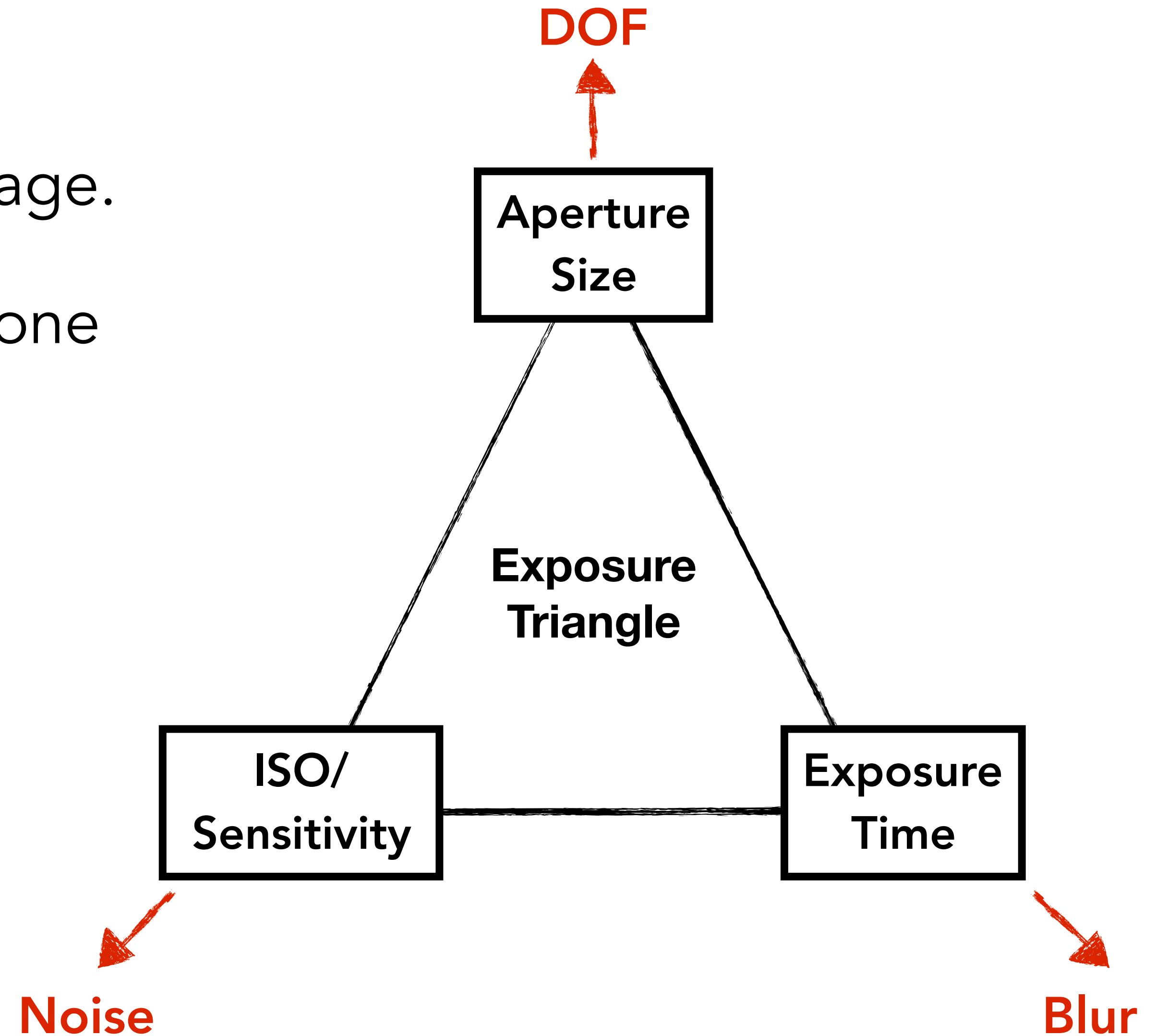
# Auto Exposure

No single "correct" exposure of an image.

For HDR scenes, it's impossible to set one single proper exposure.

Things you could change (**knobs**):

- Aperture size (A)
- Exposure time (T)
- ISO (gain). Last resort really...



# Auto Exposure

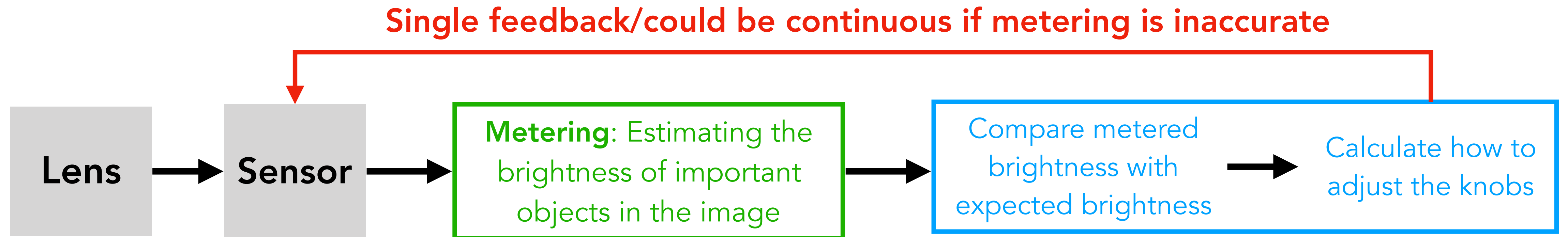
Let's assume that the scene has standard DR (HDR imaging discussed before), and the goal of AE is to allow "important" objects to be properly lit.

- That is, important objects' pixel values should be properly set.

It's a guessing game:

- What are "important" objects for the photographer?
- What is the "correct" pixel value for a properly-exposed object?
- How to set the knobs given the properly exposure level?

# Auto Exposure: General Workflow





# Metering

Estimating/measuring the brightness of the “important” subjects.

Informally, we compare the metering result with an ideal/“correct” target, i.e., compare how bright an image is and how bright a visually-pleasing image should be, from which we calculate how to adjust the exposure knobs.

We meter using RAW pixel values, which are proportional to **luminance**.

- Or convert to XYZ and use the Y value, which directly corresponds to luminance.

Key challenge: what are “important” objects to meter?

- Camera doesn't know what you think is important



# Metering is Hard



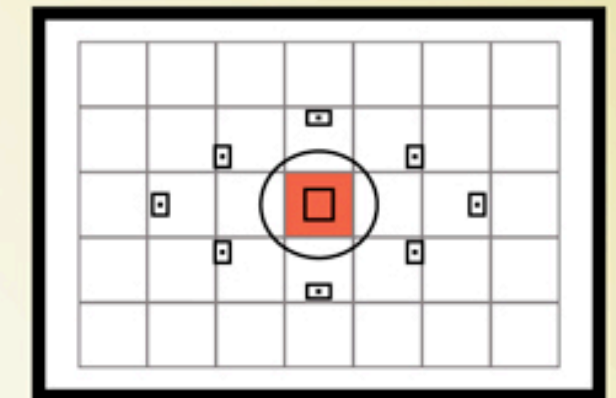
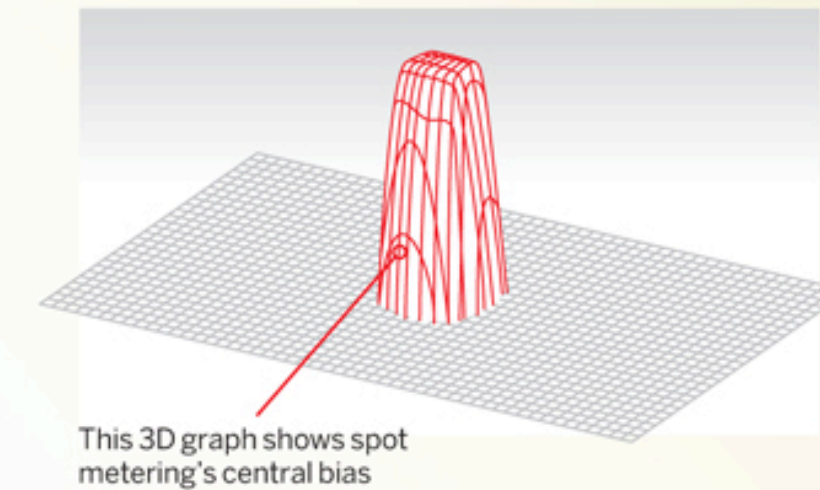
Spot metering



Center-weighted average metering



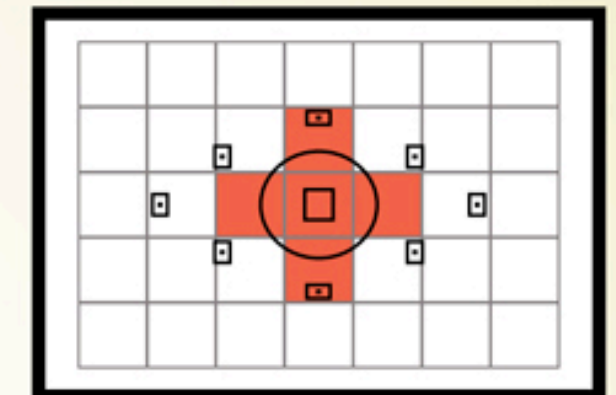
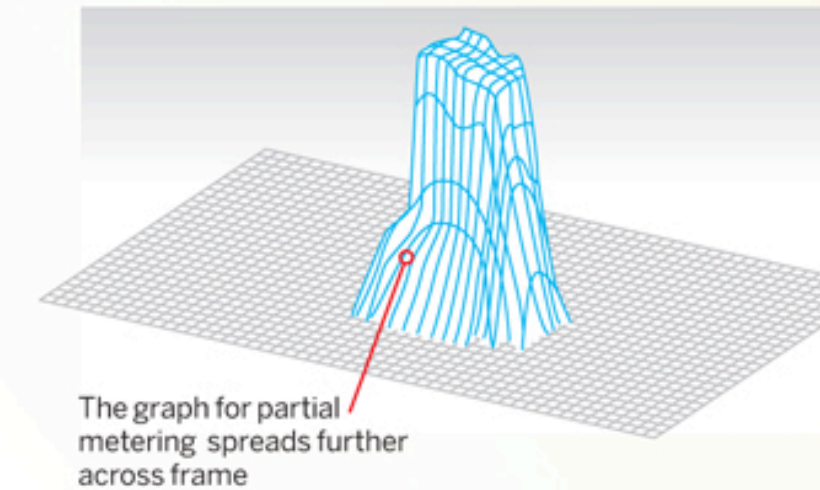
**Spot metering**  
Spot metering only measures the intensity of light over a small circular area in the centre of the viewfinder. The average is then calculated by measuring just 2-4% of the picture area.



The centre circle in the viewfinder gives a rough guide to a spot meter's coverage



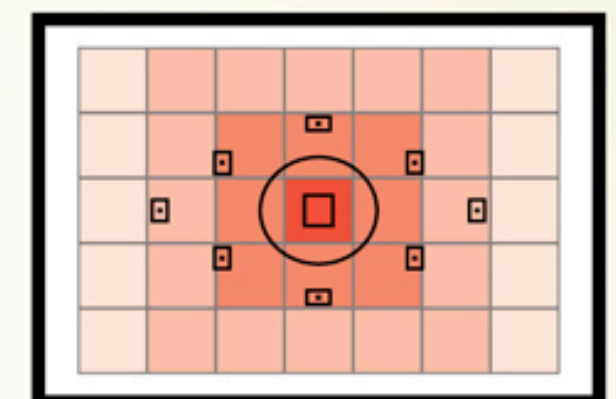
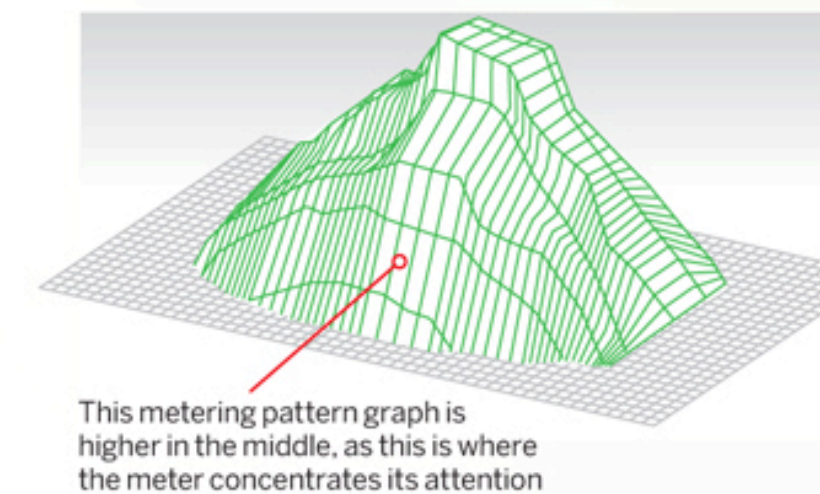
**Partial metering**  
This metering mode measures the intensity of the light over a larger circular area than in Spot mode. The average is then calculated by measuring 8-13% of the picture area.



The coverage of the partial meter spreads out slightly beyond the viewfinder's centre circle



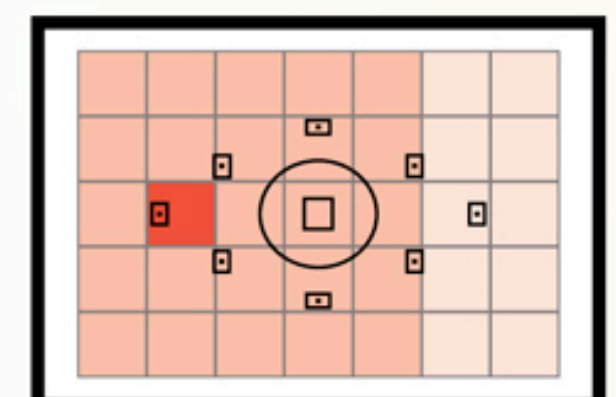
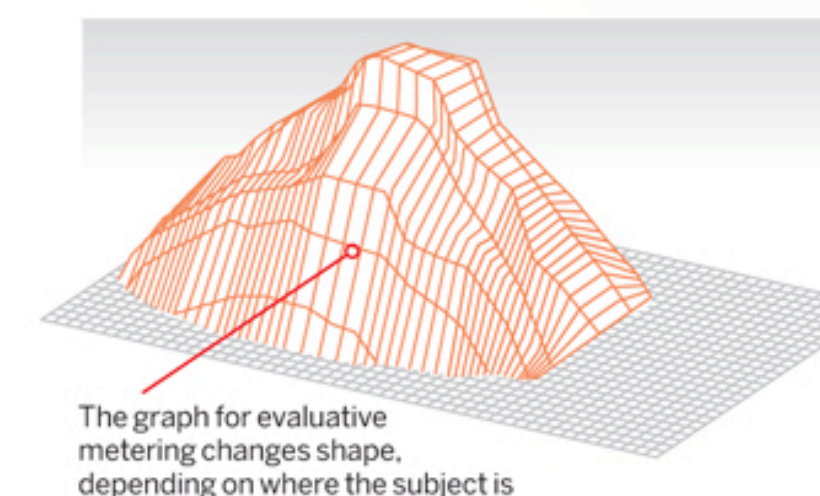
**Centre-weighted average metering**  
This light metering mode measures the light across the whole picture area, but strongly biases the reading to the centre of the viewfinder area. Unlike with Evaluative, it does not take the focus into account, so uses the same averaging pattern for every shot.



Main metering zone is bounded by the seven central focus points (SLRs with nine AF points)



**Evaluative metering**  
The default metering mode on many DSLRs, and the only option if you choose one of the basic automatic exposure modes. Measures light across the whole frame, but strongly biases the reading to the area around the autofocus point currently being used.



Main zone of interest will depend on which of the autofocus points has been used



# Metering is Hard

**Trade-off:** accurate metering requires big pixels, which limits sensor resolution.

- Low resolution image makes metering harder: hard to tell what's in the image to reason about metering results.
- Small pixels are easily saturated, so might need iterative metering (slow).

Could use a separate (small) sensor, i.e., a separate metering system. Or use the main image sensor.



# Metering is Hard

Is this a good exposure???

**Trade-off:** accurate metering requires big pixels, which limits sensor resolution.

- Low resolution image makes metering harder: hard to tell what's in the image to reason about metering results.
- Small pixels are easily saturated, so might need iterative metering (slow).

Could use a separate (small) sensor, i.e., a separate metering system. Or use the main image sensor.





# Metering is Hard

Is this a good exposure???

**Trade-off:** accurate metering requires big pixels, which limits sensor resolution.

- Low resolution image makes metering harder: hard to tell what's in the image to reason about metering results.
- Small pixels are easily saturated, so might need iterative metering (slow).

Could use a separate (small) sensor, i.e., a separate metering system. Or use the main image sensor.



# AF/AE/AWB: Why Are They Hard?

They are very subjective and personal.

- What are important objects that viewers/photographers want to focus on?
- What objects do viewers/photographers deem important and thus have to be well-lit?
- What will the illuminant be when a photo is viewed later?
- What objects do the viewers think should be perceived as white?

Fundamentally, these are all computer vision problems: understanding the semantics in the image. Can computer vision/AI help?

- Detect/track interesting objects?
- Photo capturing history?
- Emotion? Sound? Weather? Other hints/modalities?