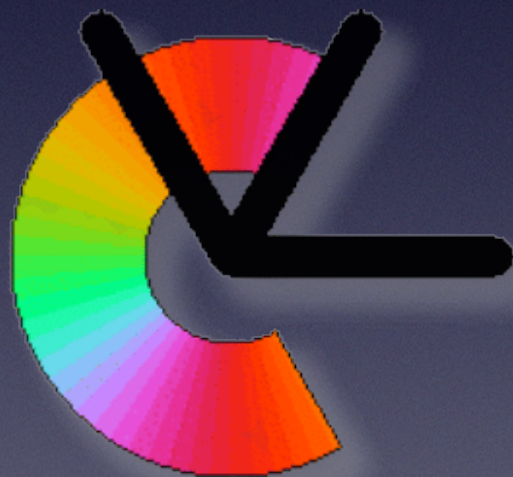


# Range cameras II

ToF cameras, Lidar, and point-set registration



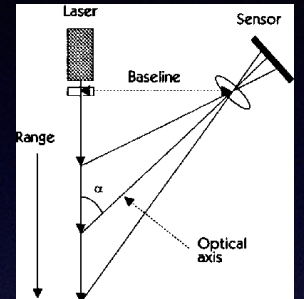
**Per-Erik Forssén**  
**Division of Computer Vision**  
**Department of Electrical Engineering**  
**Linköping University**



# Common 3D Cameras

- **Sheet-of-light laser triangulation**

e.g. SICK, previous lecture.



- **Structured light**

e.g. Kinect v1, 2010, Apple Face ID sensor 2017, previous lecture.



- **Fringe pattern cameras**

e.g. Micro Epsilon, previous lecture?



- **Time-of-flight cameras**

e.g. Kinect v2, 2013. Hololens, Azure Kinect, Today.



- **LIDAR sensors**

Velodyne, Ouster. Work well outdoors. Today.





# Common 3D Cameras



- Tekniska museet, Stockholm autumn 2021.  
Kinect v1. Introduced Nov 2010



# The time-of-flight principle

- Classic way to measure depth, used in e.g. RADAR.
- Emit electromagnetic radiation and measure delay until echo is received.
- $\text{distance} = \text{time} * \text{speed-of-light} / 2$



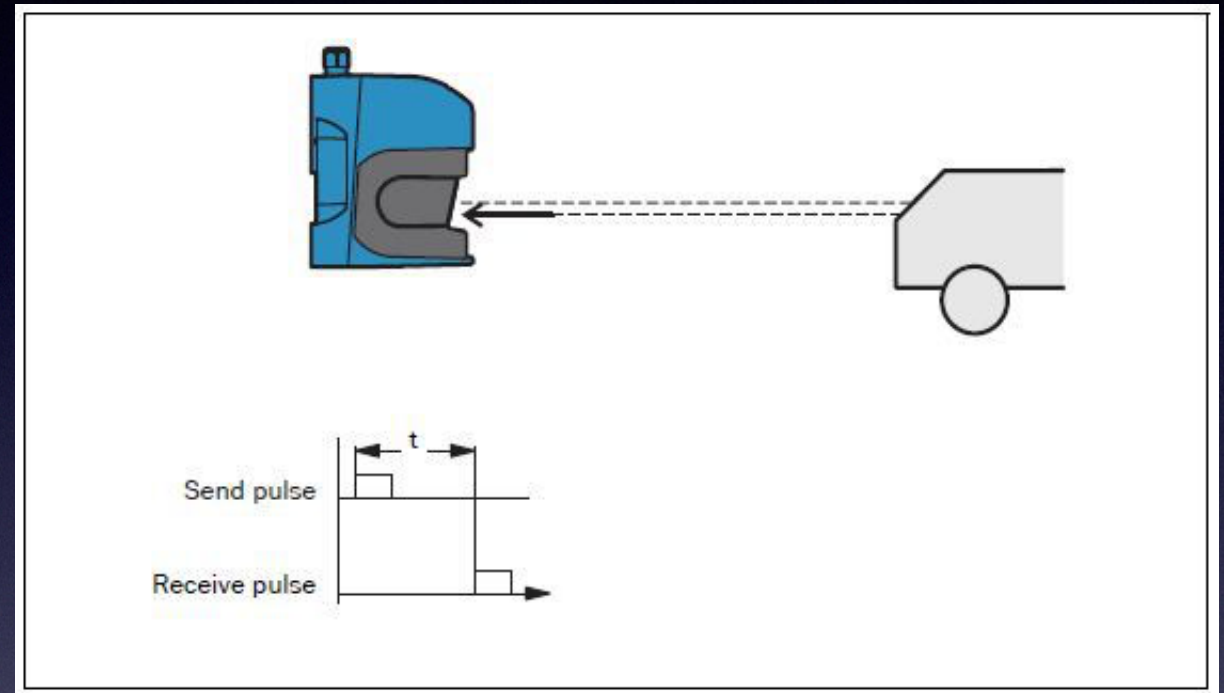
From Wikimedia Commons, the free media repository



# The time-of-flight principle

- LIDAR  
(light detection and ranging)  
Same idea as RADAR:

Emit a pulse and count the time until it returns (i.e. time-of-flight).



Source: SICK

- Measurement relation:

$$\text{distance} = \text{time} * \text{speed-of-light} / 2$$



# The time-of-flight principle

- Requires an extremely accurate clock as  $v=3*10^8$  m/s.

clock accuracy	depth accuracy
1 millisecond	$\pm 150\text{km}$
1 nanosecond ( $10^{-9}$ )	$\pm 1.5\text{dm}$
1 picosecond ( $10^{-12}$ )	$\pm 0.15\text{mm}$

- Techniques like **continuous-wave**, and **pulsed** time-of-flight maintain high depth accuracy with a less accurate clock.



# Time-of-flight cameras

- Continuous wave (CW-ToF)  
+Reduced requirements on clock

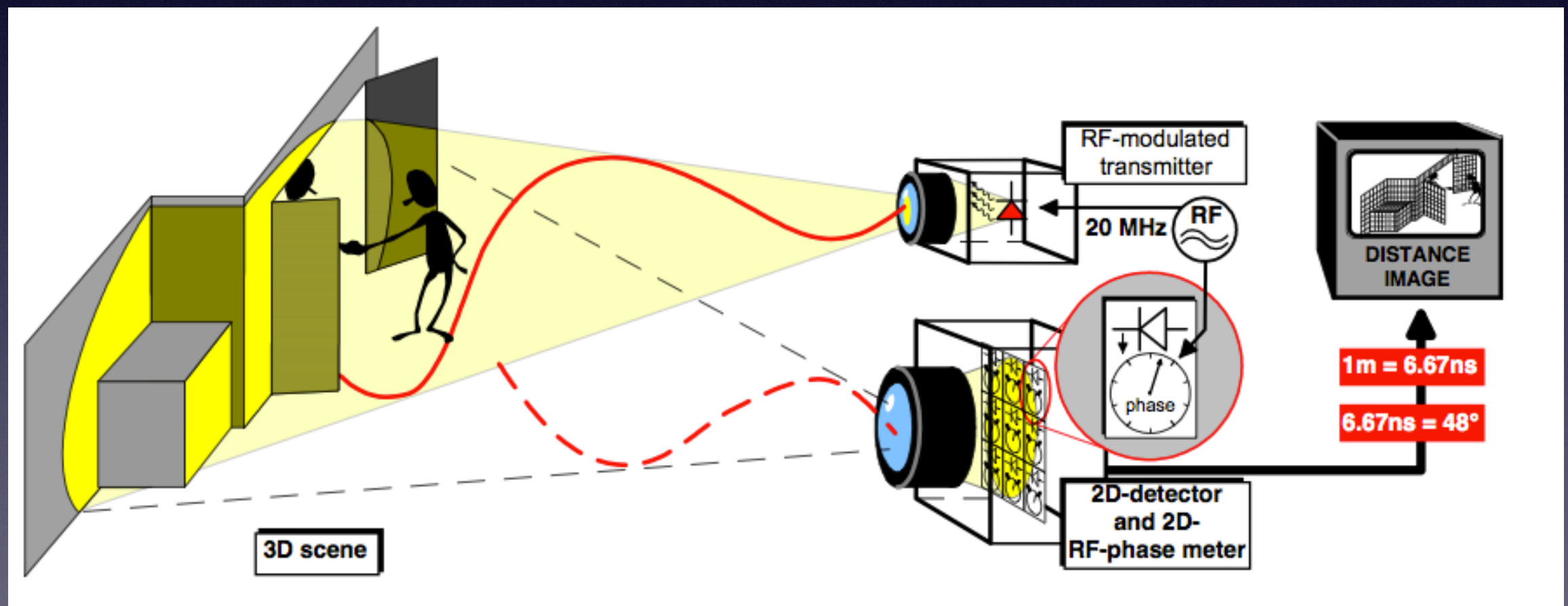
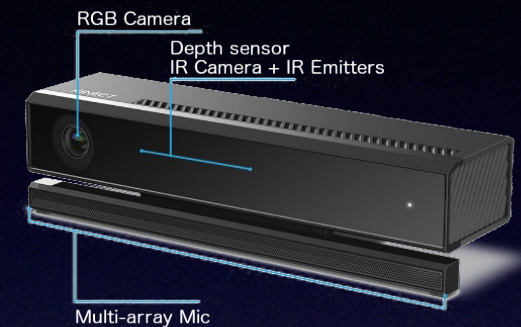
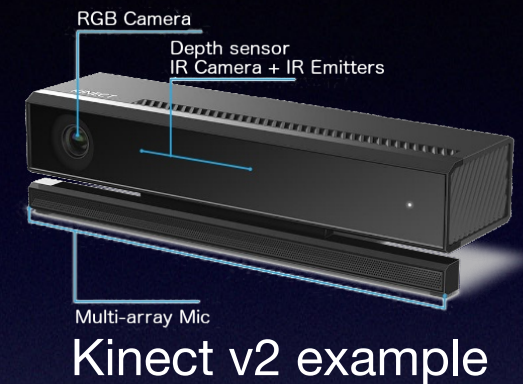


Image: R. Lange PhD thesis, 2000



# Time-of-flight cameras



- ToF decoding for continuous wave (CW-ToF)

- Emitted signal is amplitude modulated (not frequency modulated)

$$e(t) = I_0(1 + \cos(2\pi f_m t))$$

- Received signal will be

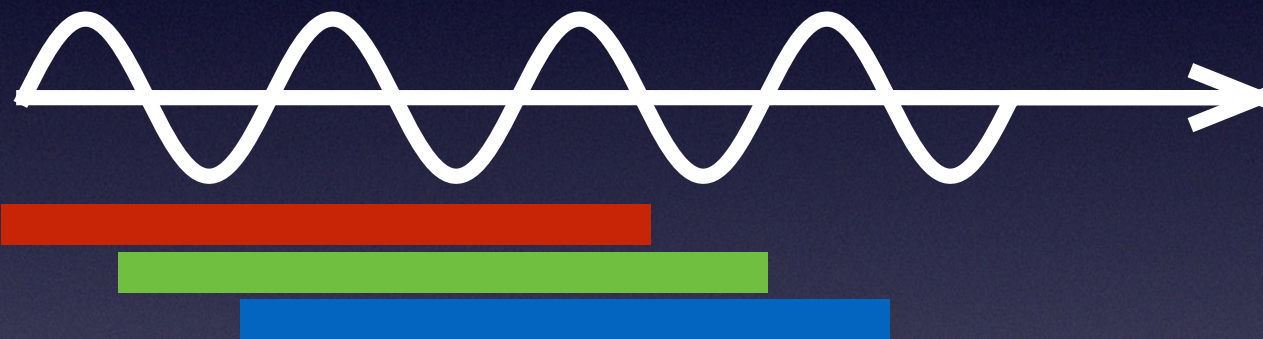
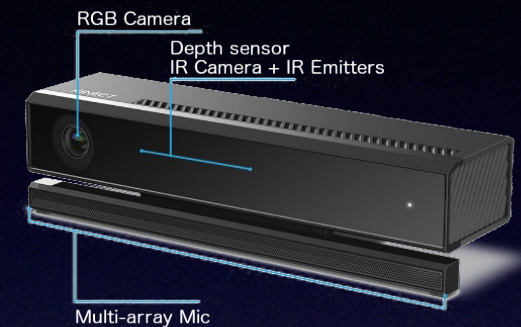
$$r(t) = I_r(1 + \cos(2\pi f_m t - \phi))$$

- A correlation is computed in each pixel with three reference signals, to extract  $\phi$ , the phase shift.



# Time-of-flight cameras

- ToF decoding for continuous wave (CW-ToF)



$$r(t) = I_r(1 + \cos(2\pi f_m t - \phi))$$

- Create three time-shifted versions of the input

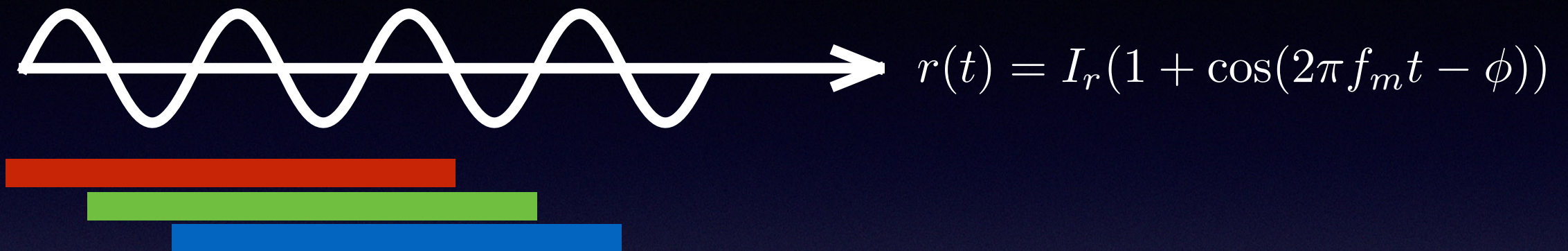
$$s_0(t) = I_0(1 + \cos(2\pi f_m t + 0^\circ))$$

$$s_1(t) = I_0(1 + \cos(2\pi f_m t + 120^\circ))$$

$$s_2(t) = I_0(1 + \cos(2\pi f_m t + 240^\circ))$$



# Time-of-flight cameras

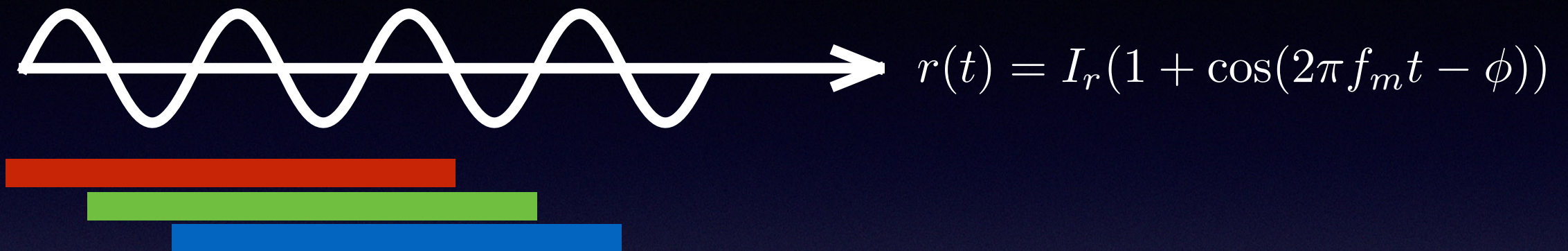


- The sensor correlates the received signal  $r(t)$  with the three  $s_k(t)$  (integration time  $t_1 - t_0 \approx 5\text{ms}$ )

$$v_k = \int_{t_0}^{t_1} s_k(t) r(t) dt$$



# Time-of-flight cameras



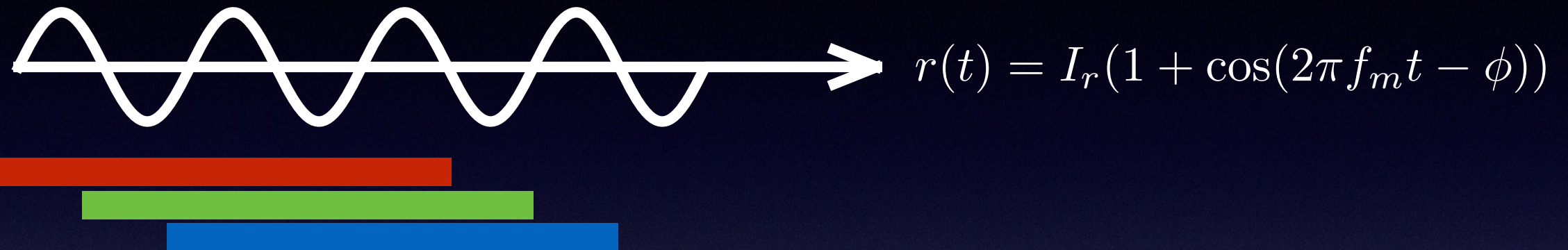
- The sensor correlates the received signal  $r(t)$  with the three  $s_k(t)$  (integration time  $t_1 - t_0 \approx 5\text{ms}$ )

$$v_k = \int_{t_0}^{t_1} s_k(t) r(t) dt$$

- From  $v_0$ ,  $v_1$  and  $v_2$  we can now accurately estimate the phase shift

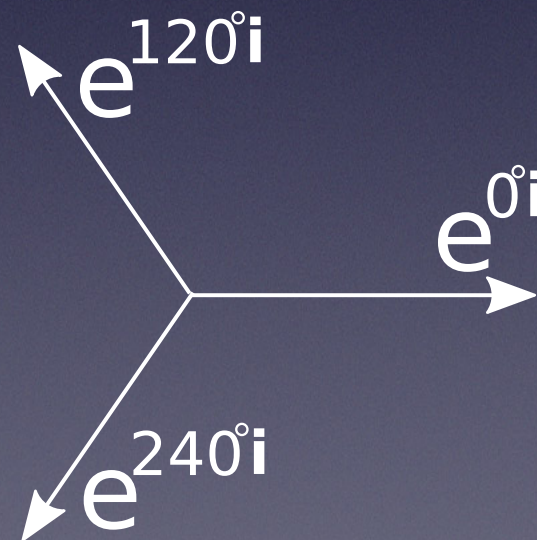


# Time-of-flight cameras



- This is done using a vector summation of the correlations:

$$\mathbf{z} = \sum_{k=0}^2 v_k e^{-120^\circ \cdot \mathbf{i}k}$$

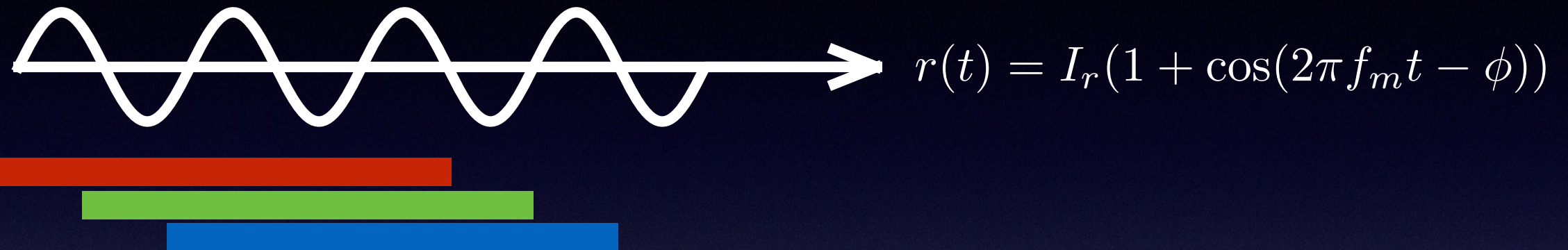


For a derivation of this expression, see:

Frank et al. Theoretical and experimental error analysis of continuous-wave time-of-flight range cameras , OE 2009

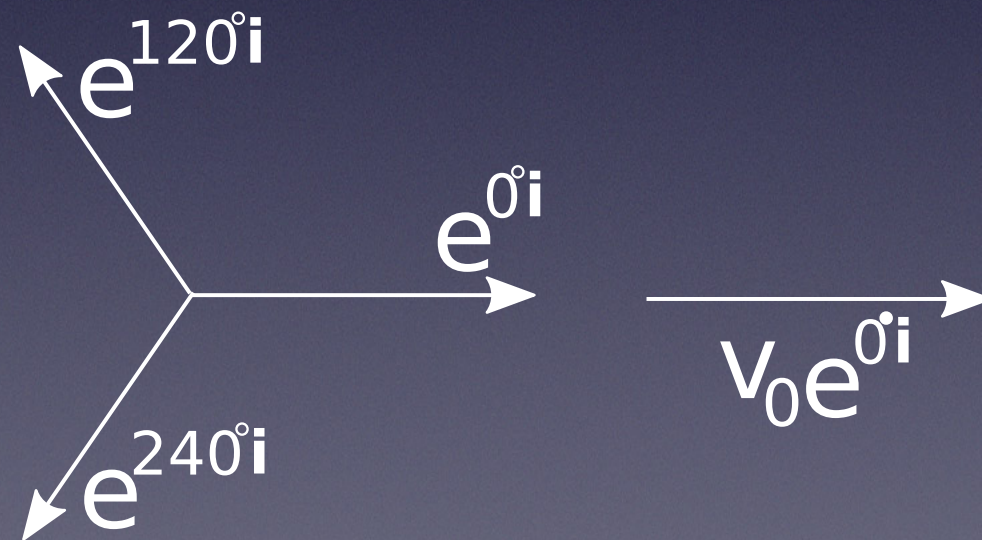


# Time-of-flight cameras



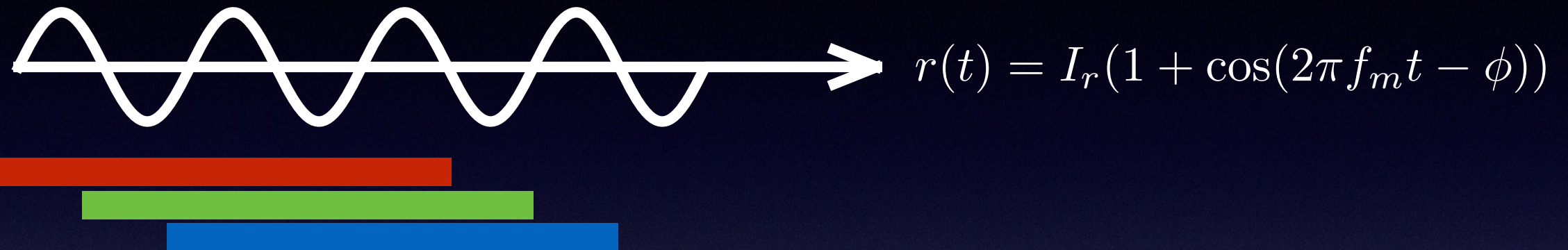
- This is done using a vector summation of the correlations:

$$\mathbf{z} = \sum_{k=0}^2 v_k e^{-120^\circ \cdot \mathbf{i}k}$$





# Time-of-flight cameras



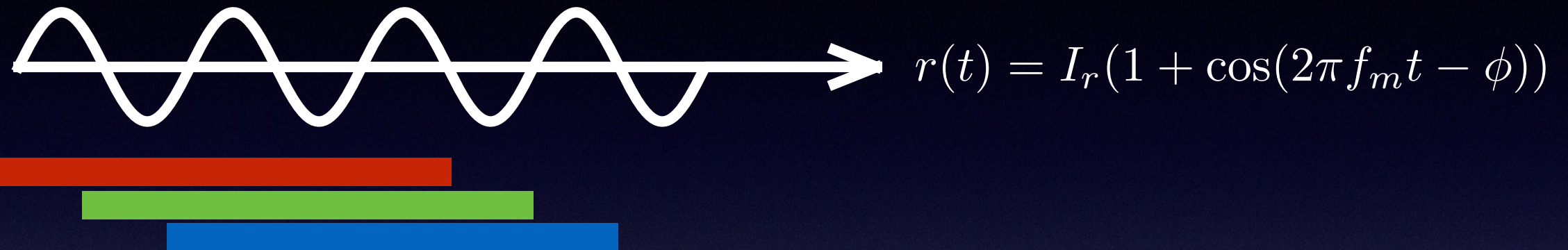
- This is done using a vector summation of the correlations:

$$\mathbf{z} = \sum_{k=0}^2 v_k e^{-120^\circ \cdot \mathbf{i}k}$$

The diagram illustrates the vector summation process in the complex plane. It shows three vectors originating from the same point:  $e^{120^\circ \mathbf{i}}$  (pointing up and to the left),  $e^{240^\circ \mathbf{i}}$  (pointing down and to the left), and  $e^{0^\circ \mathbf{i}}$  (pointing to the right). These vectors are summed to produce a resultant vector  $v_0 e^{0^\circ \mathbf{i}}$  (pointing to the right). A second vector  $v_1 e^{120^\circ \mathbf{i}}$  is shown pointing up and to the right, representing the next step in the summation process.

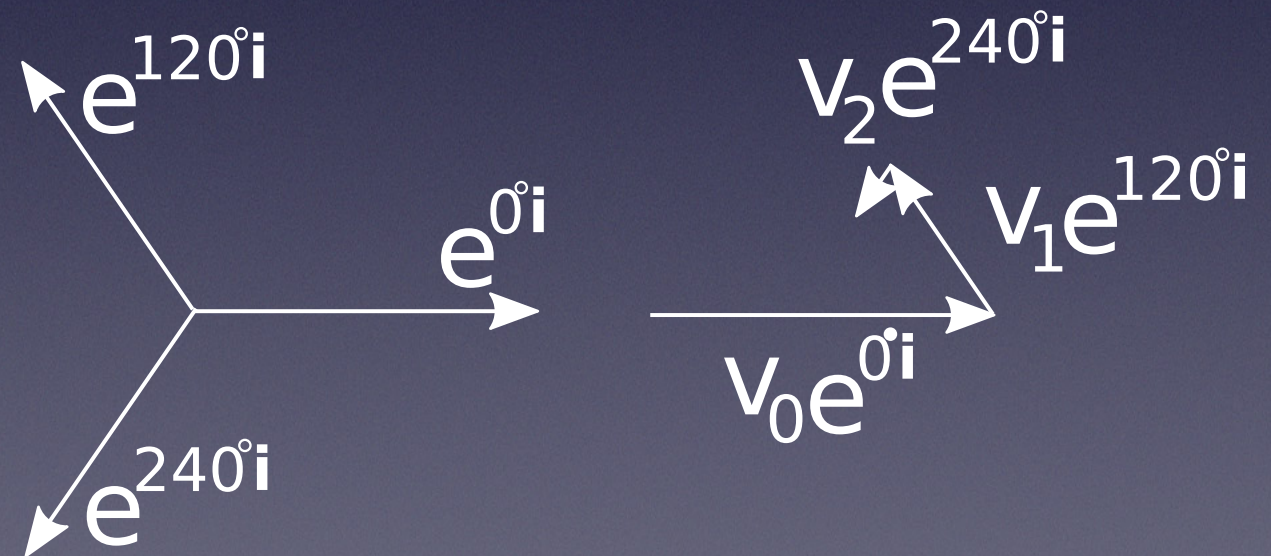


# Time-of-flight cameras



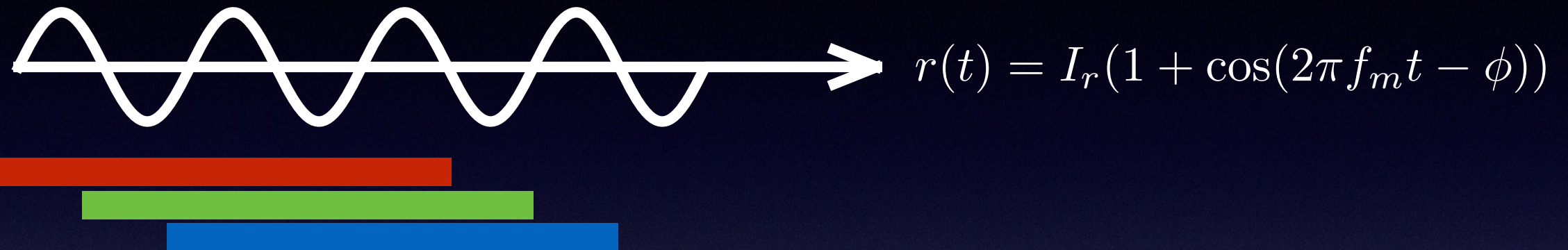
- This is done using a vector summation of the correlations:

$$\mathbf{z} = \sum_{k=0}^2 v_k e^{-120^\circ \cdot \mathbf{i}k}$$



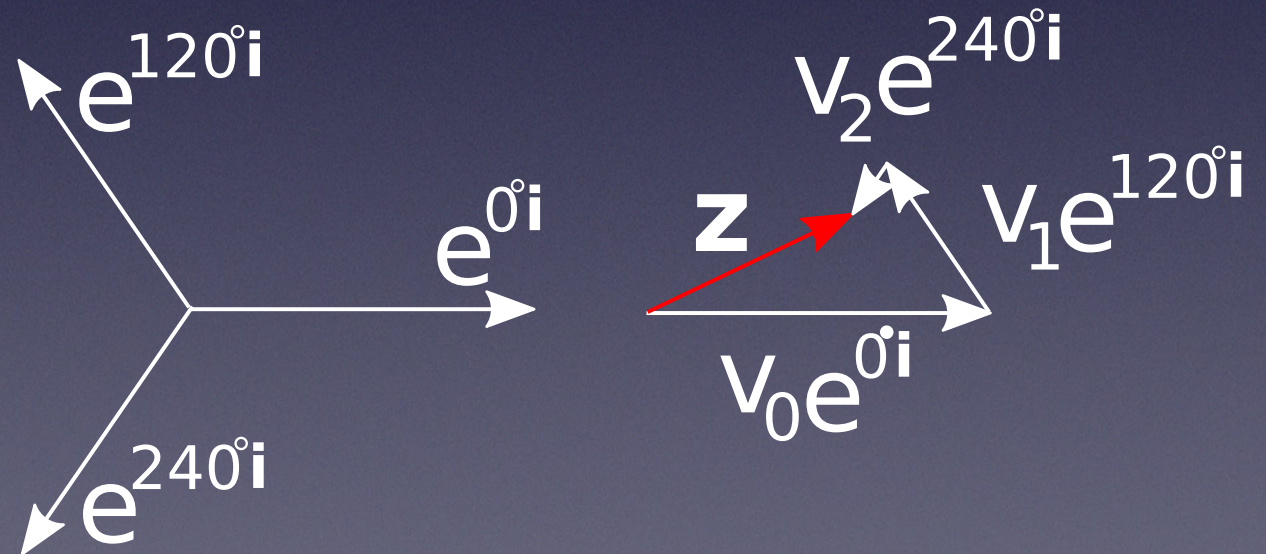


# Time-of-flight cameras



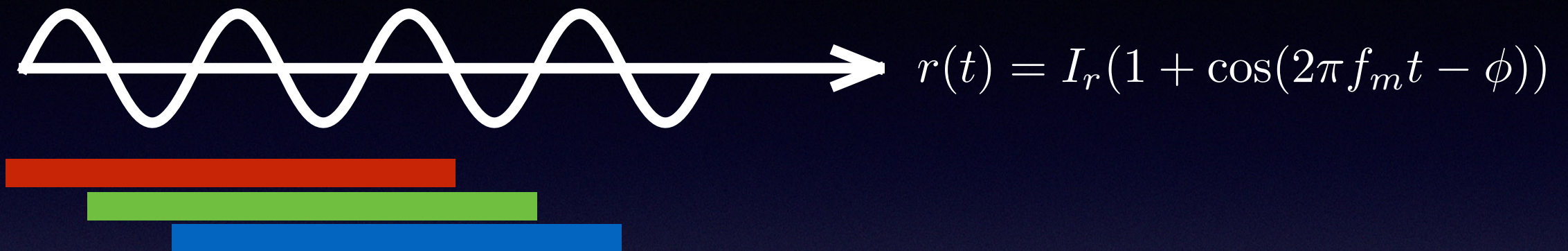
- This is done using a vector summation of the correlations:

$$\mathbf{z} = \sum_{k=0}^2 v_k e^{-120^\circ \cdot \mathbf{i}k}$$





# Time-of-flight cameras

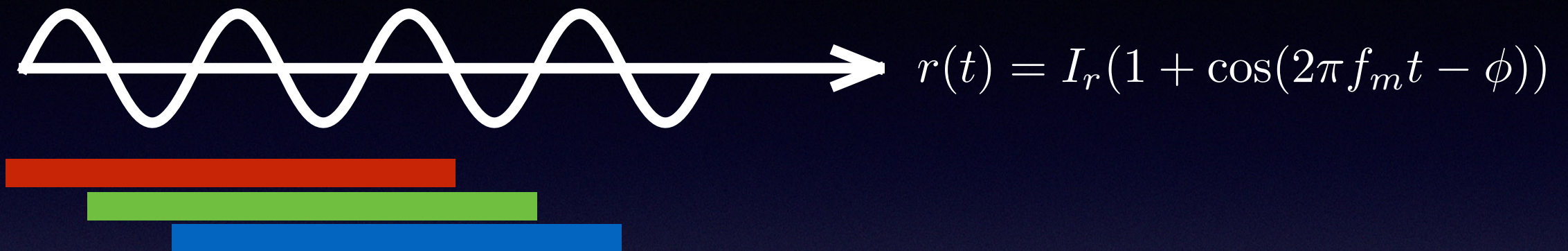


- This is done using a vector summation of the correlations:

$$\mathbf{z} = \sum_{k=0}^2 v_k e^{-120^\circ \cdot \mathbf{i}k} \quad \hat{\phi} = \arg \mathbf{z}$$
$$\hat{a} = |\mathbf{z}|$$



# Time-of-flight cameras

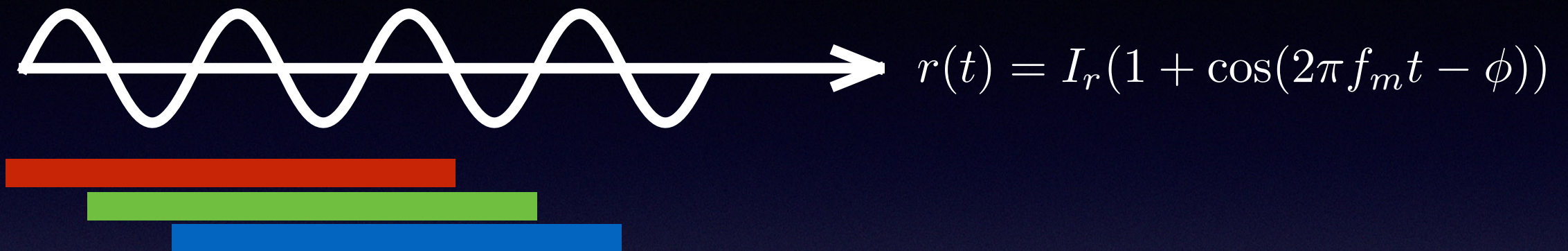


- This is done using a vector summation of the correlations:

$$\mathbf{z} = \sum_{k=0}^2 v_k e^{-120^\circ \cdot \mathbf{i}k} \quad \hat{\phi} = \arg \mathbf{z} \quad d = \frac{c\phi}{4\pi f_m}$$
$$\hat{a} = |\mathbf{z}|$$



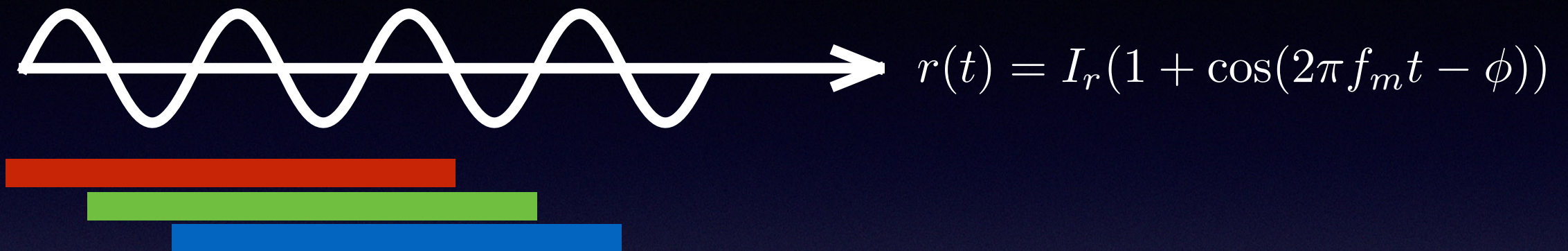
# Time-of-flight cameras



- For three or more correlations:
  1. the amplitude  $\hat{a} = |\mathbf{z}|$  will correspond to **coherence**, i.e. we can use it to detect whether a sinusoidal signal is actually received.
  2. a constant ambient offset will not matter.
  3.  $\hat{\phi}$  is also insensitive to attenuation of  $r(t)$ .



# Time-of-flight cameras



- The decoded depth  $\hat{\phi}$  will have a *periodic ambiguity*:

$$d = \frac{c\phi}{4\pi f_m} \quad \phi \approx \hat{\phi} + 2\pi n \quad n \in \mathbb{Z}$$

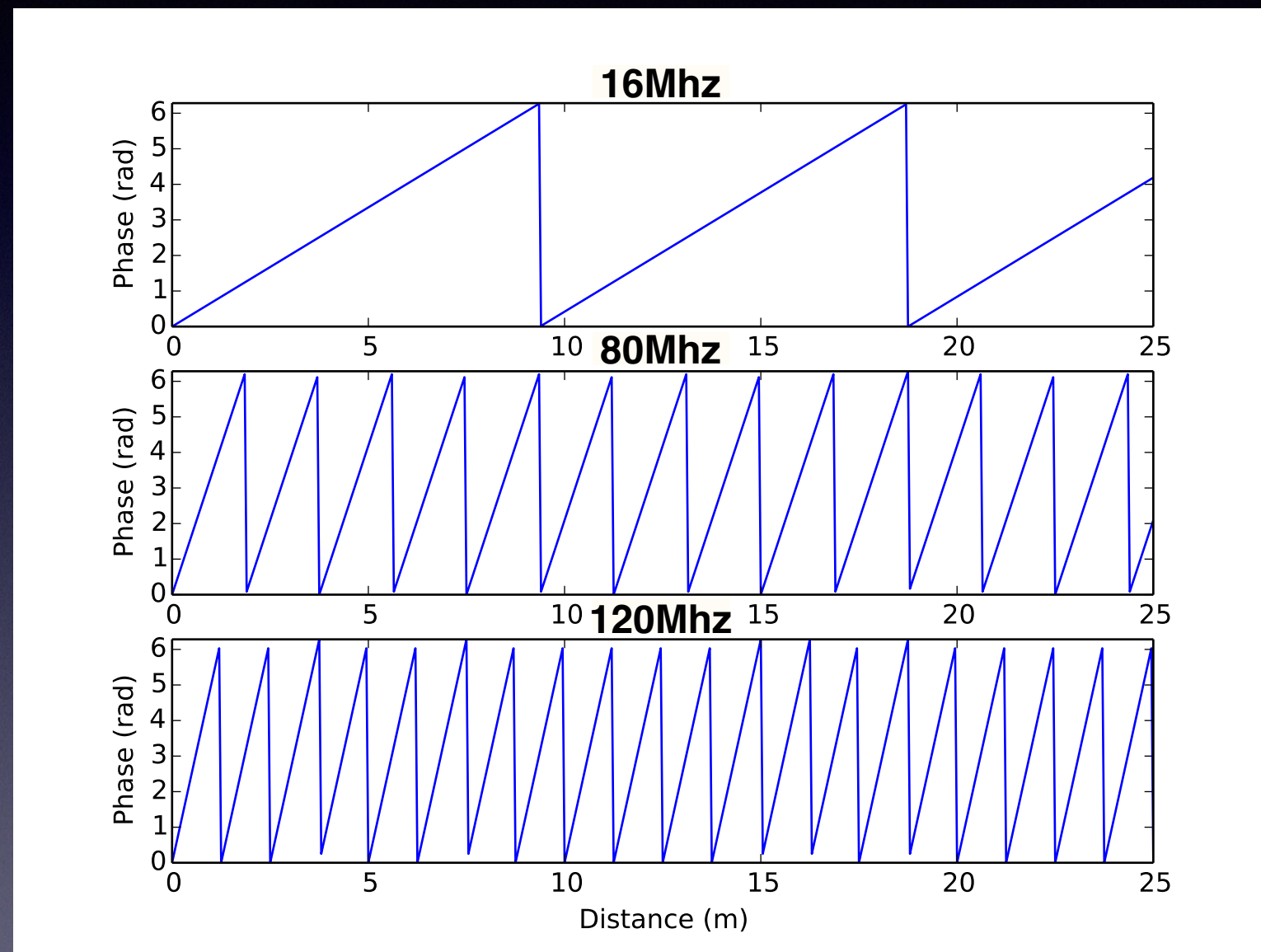


# Kinect v2 depth decoding

- Distance travelled:  $d = \frac{c\phi}{4\pi f_m}$
- Problem:  $\phi \approx \hat{\phi} + 2\pi n \quad n \in \mathbb{Z}$
- As a fix, Kinect v2 uses three modulation frequencies.
- This means 3 phases x 3 frequencies = 9 images are recorded for each depth frame.



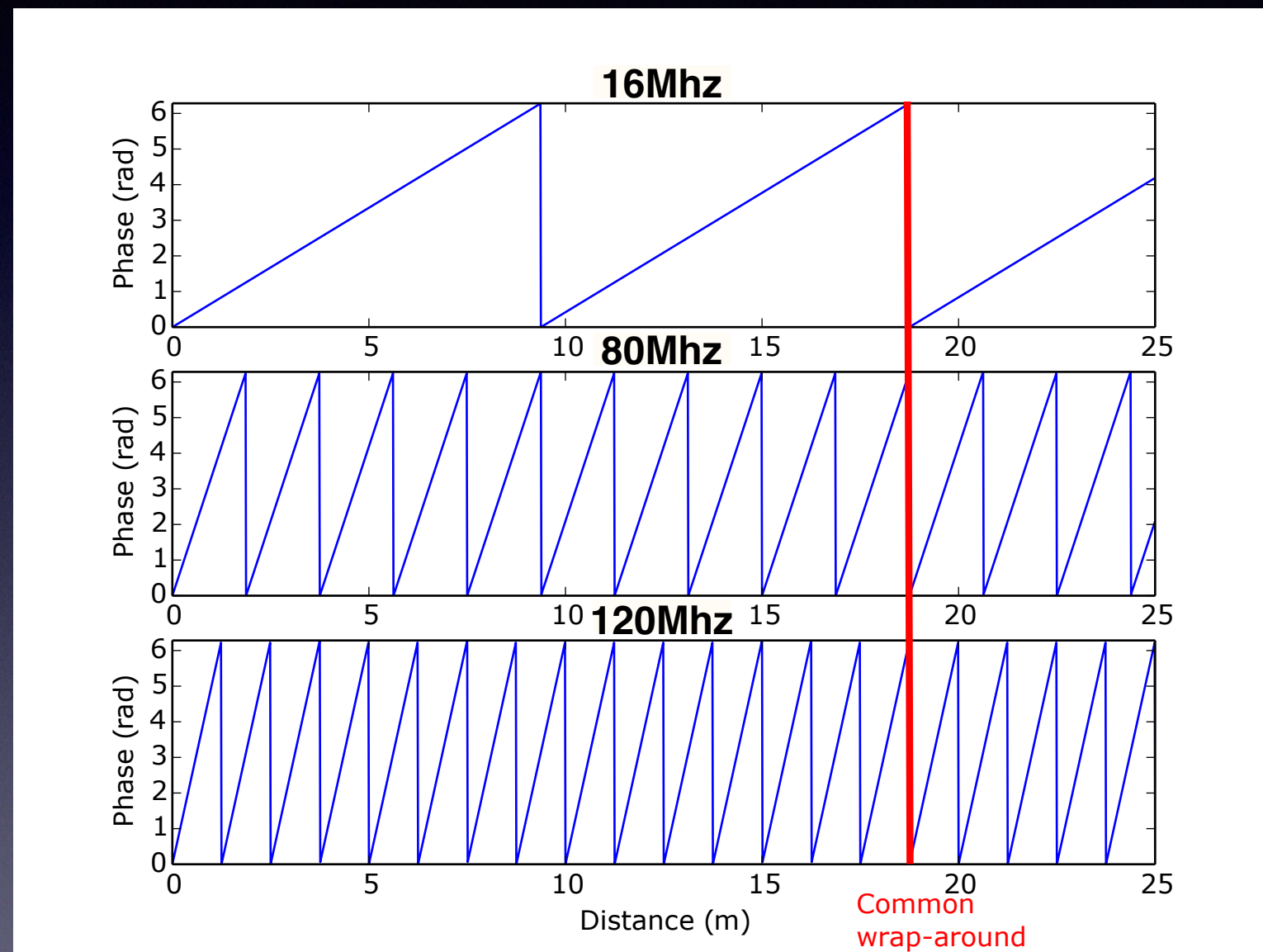
# Kinect v2 depth decoding



- Three modulation frequencies



# Kinect v2 depth decoding



- Felix Järemo Lawin, Per-Erik Forssén, Hannes Ovrén, "Efficient Multi-Frequency Phase Unwrapping using Kernel Density Estimation", **ECCV16**, October 2016



# **Efficient Multi-Frequency Phase Unwrapping using Kernel Density Estimation**

**Felix Järemo Lawin, Per-Erik Forssén,  
Hannes Ovrén**

- ECCV16 Video Example



# Pulsed ToF decoding

- Decoding for pulsed ToF (aka. Flash LIDAR)

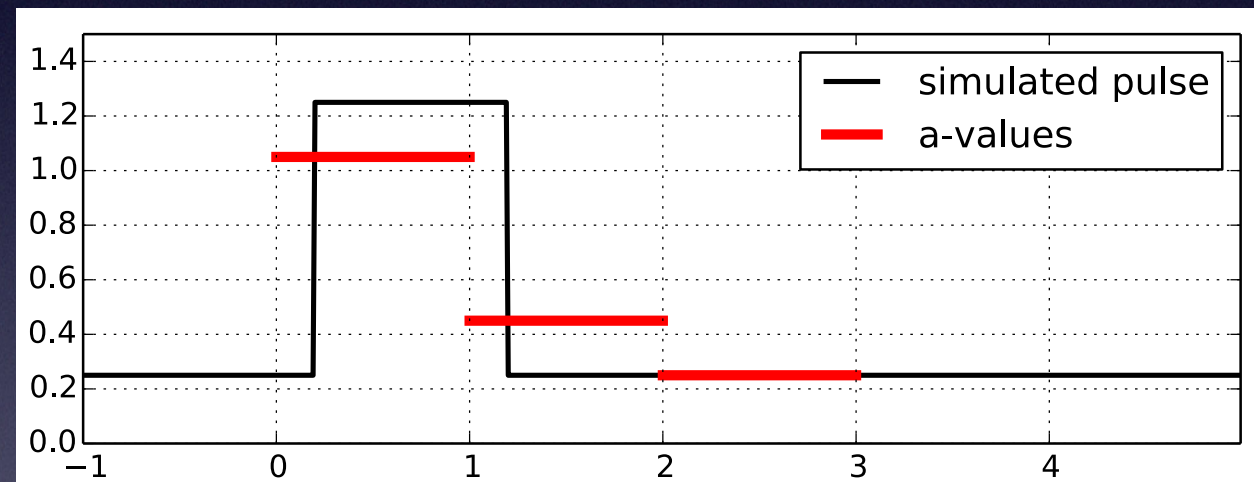


Fotonic (Magna/Veoneer)  
Prototype ToF camera



# Pulsed ToF decoding

- Decoding for pulsed ToF (aka. Flash LIDAR)



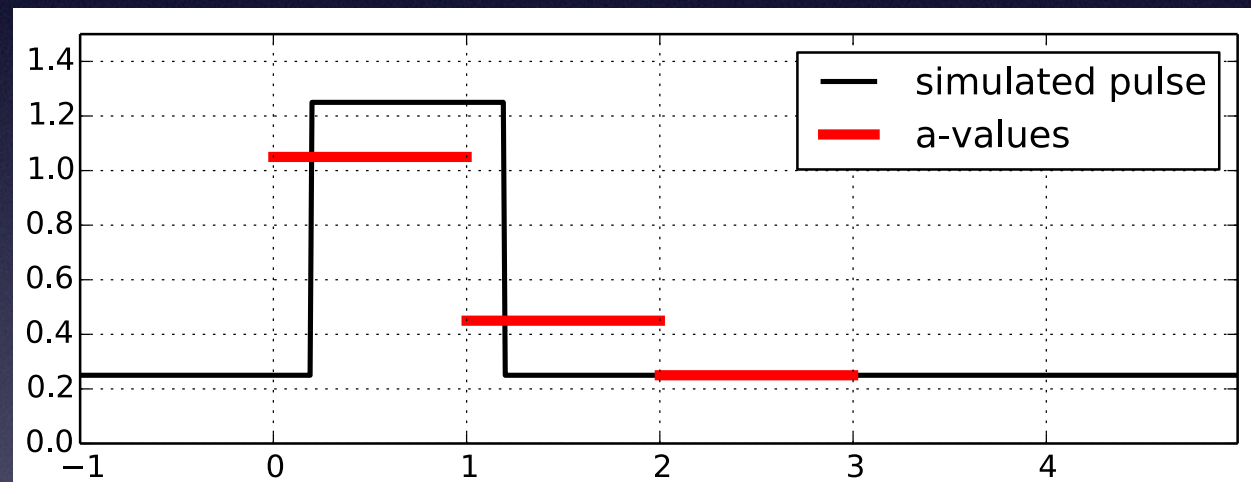
Fotonic (Magna/Veoneer)  
Prototype ToF camera

- Measure the reflected light in three short intervals (with length equal to light pulse length)



# Pulsed ToF decoding

- Decoding for pulsed ToF (aka. Flash LIDAR)



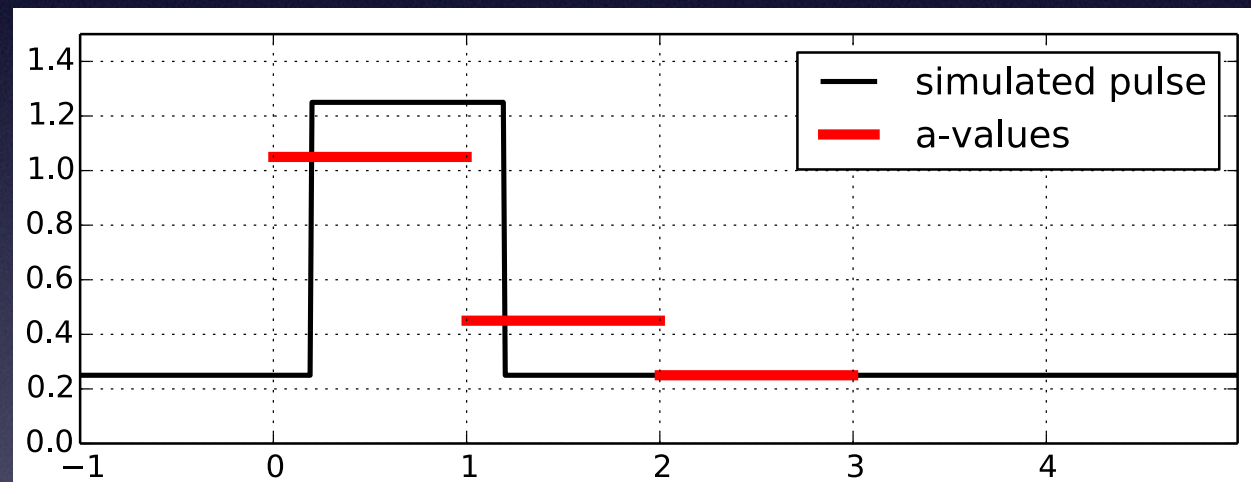
$$b = a_2 \quad (\text{ambient light estimate})$$

$$m = \sum_{k=0}^2 a_k - 3b \quad (\text{pulse energy})$$



# Pulsed ToF decoding

- Decoding for pulsed ToF (aka. Flash LIDAR)



$$b = a_2 \quad (\text{ambient light estimate})$$

$$m = \sum_{k=0}^2 a_k - 3b \quad (\text{pulse energy})$$

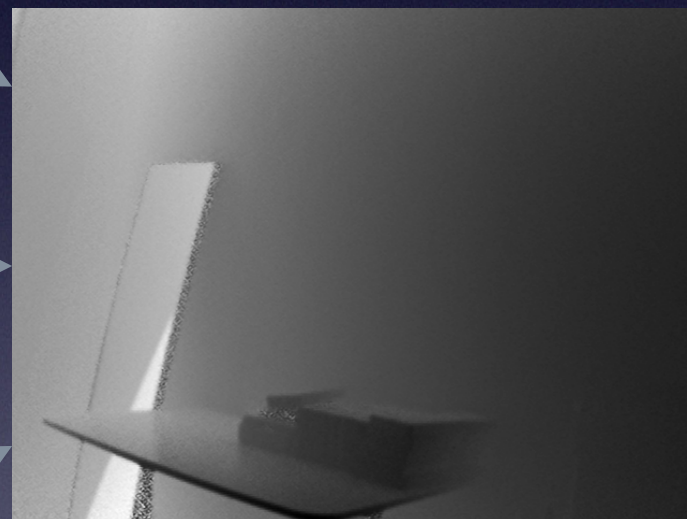
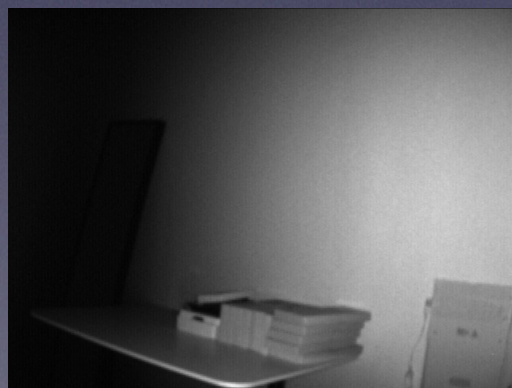
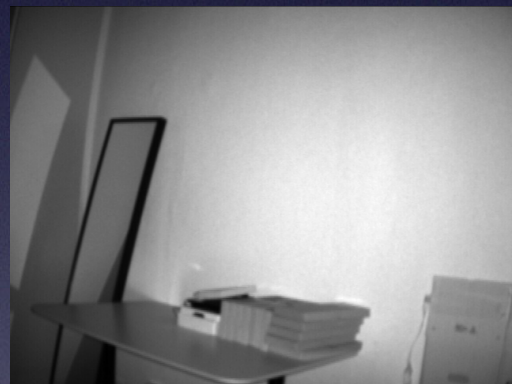
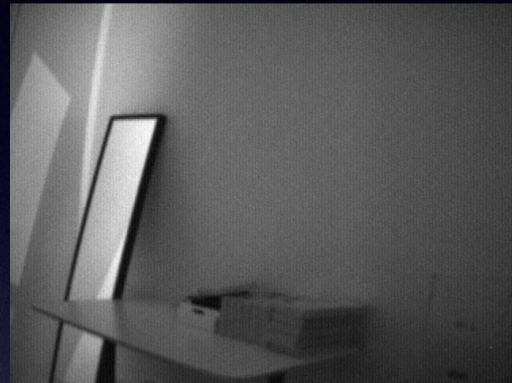
- Assuming pulse in  $[0,2]$  interval:

$$t_d = (t_1 - t_0) \frac{a_1 - b}{m}$$



# Pulsed ToF decoding

sensor images



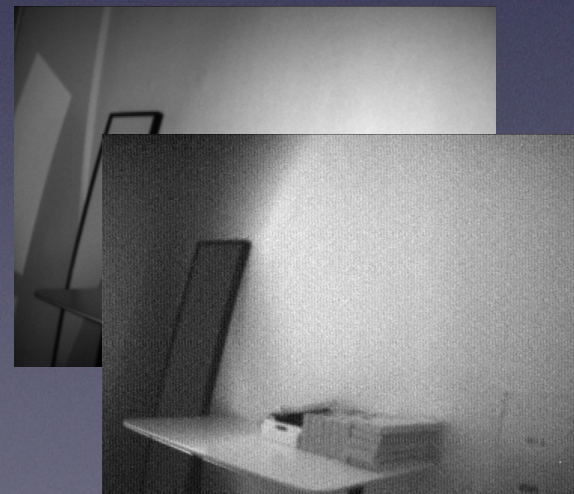
Decoded depth

More general expression:

$$b = \min(a_0, a_1, a_2)$$

$$m = \sum_{k=0}^2 (a_k - b)$$

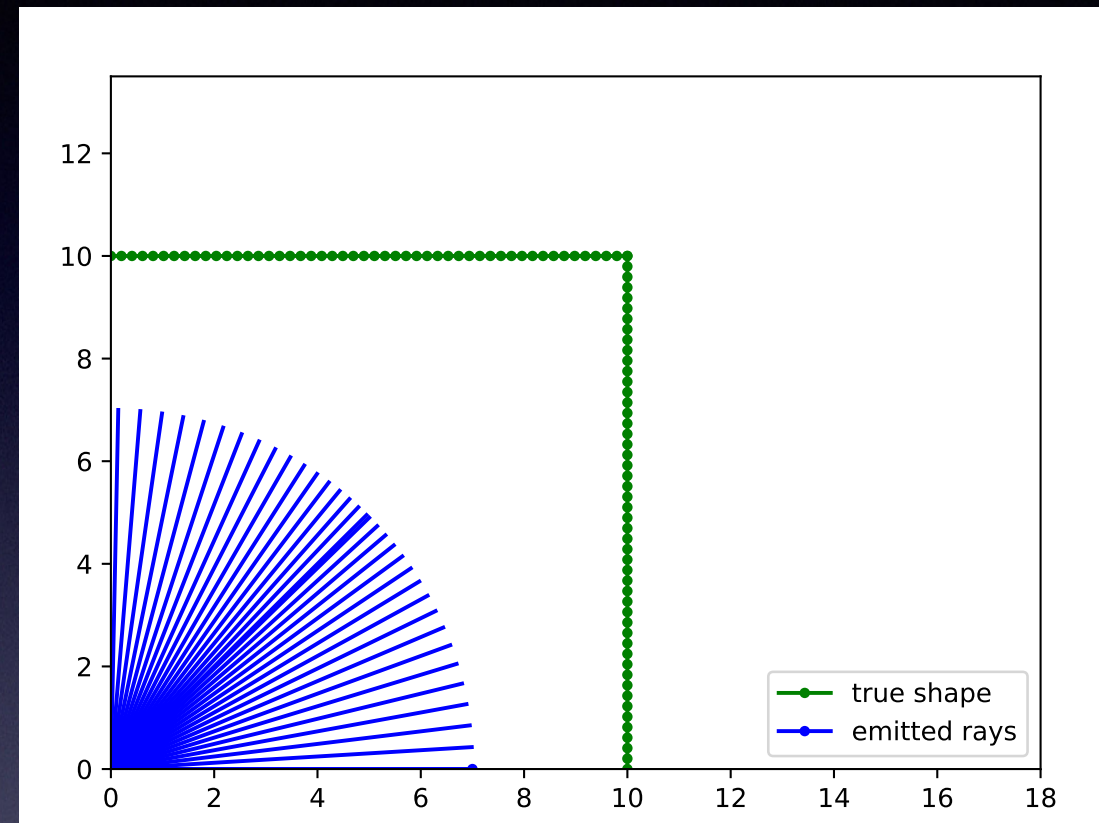
$$t_d = \frac{1}{m} \sum_{k=0}^2 t_k (a_k - b)$$





# Multipath interference

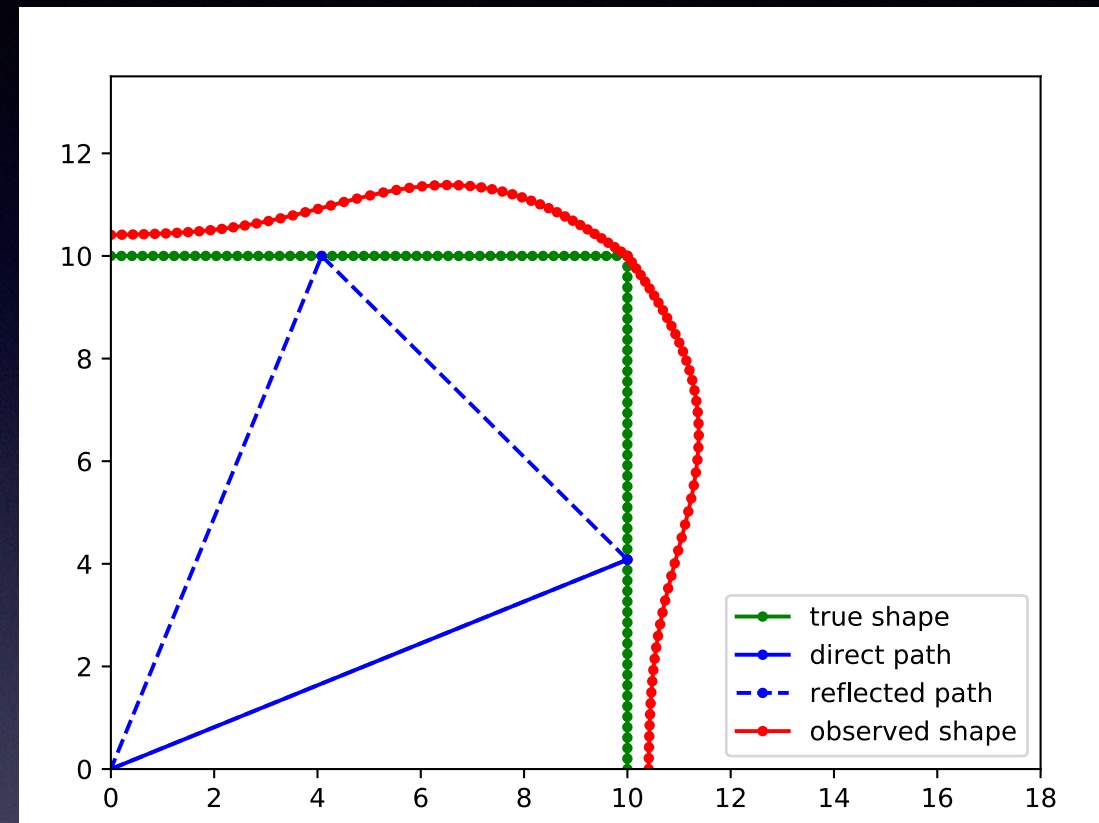
- Many light rays emitted at once.
- $\Rightarrow$  depth distortion on concave surfaces.





# Multipath interference

- Many light rays emitted at once.
- $\Rightarrow$  depth distortion on concave surfaces.
- Obtained depth is an intensity-weighted average of the path lengths (a material dependent distortion).





# Multipath interference

- To reduce Multipath interference (MPI), Flash Lidar sensors often introduce line or column scanning.
- MPI can still occur along the line though.

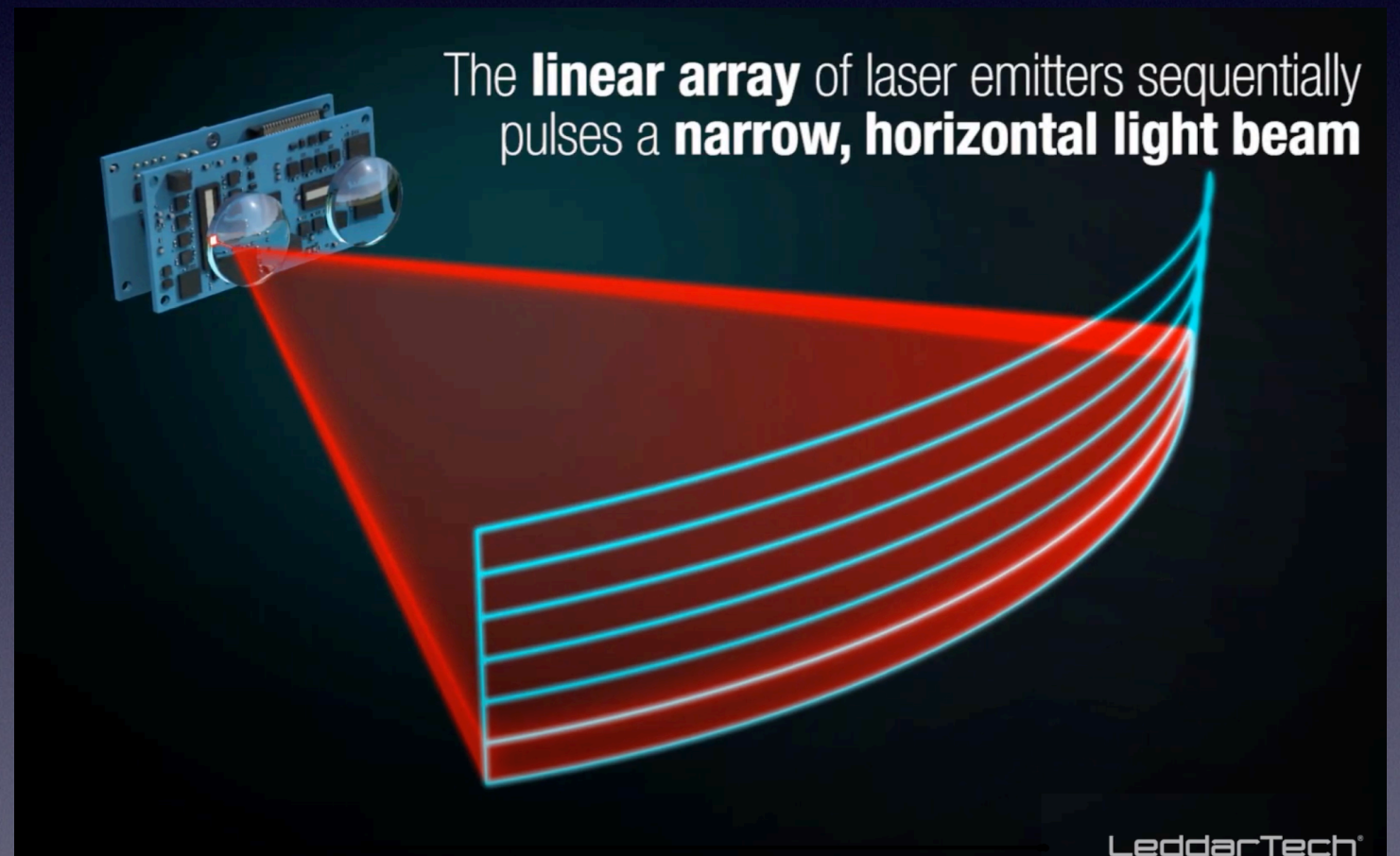


Image source: LeddarTech



# Depth sensors for automation

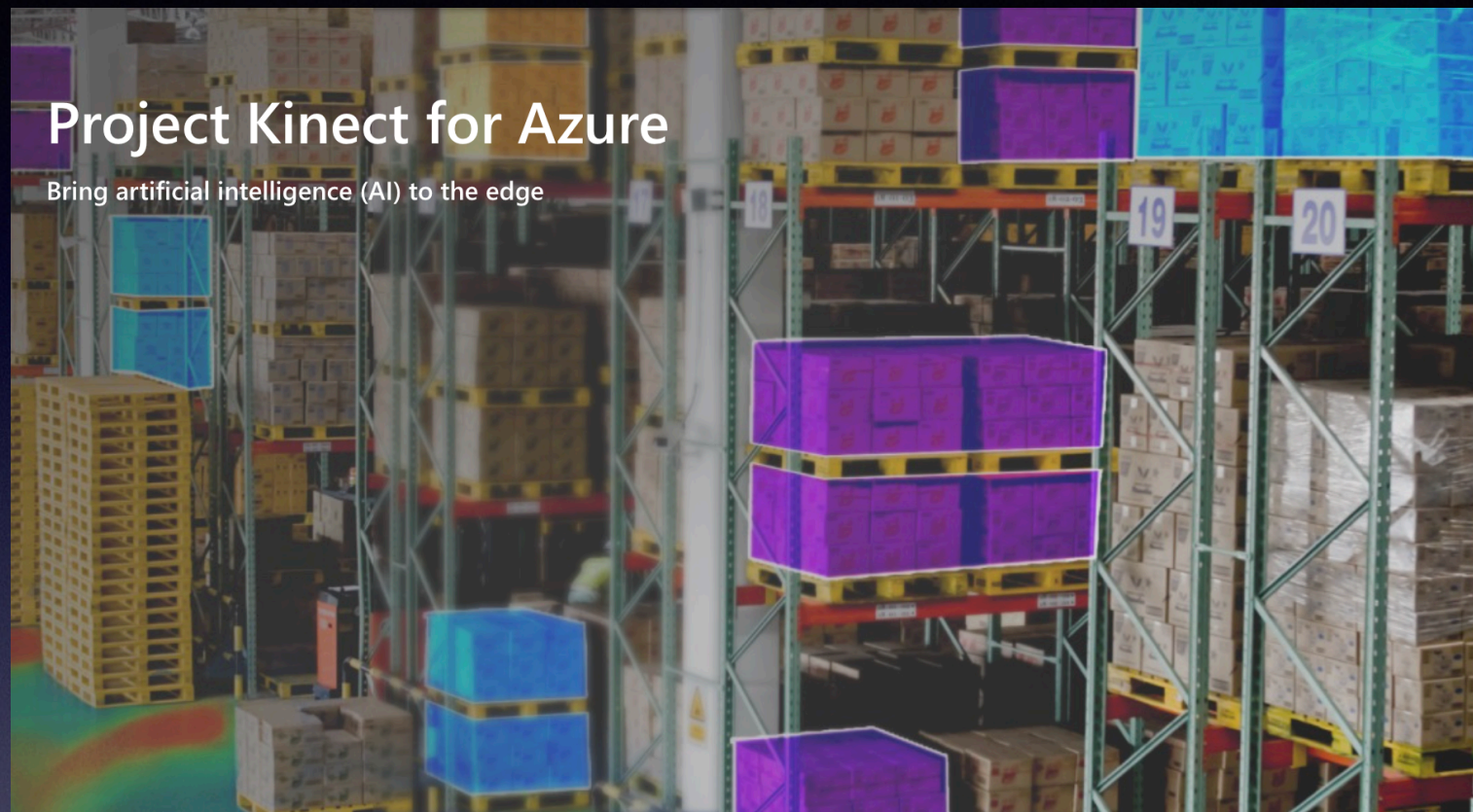
- In warehouse automation SICK LMS and other line-scanning sensors have been the first hand choice.
- As they are line scanning they only observe obstacles in one plane.
- For obstacle avoidance, full 3D is safer.
- ToF cameras or Flash LIDAR have long range and are relatively inexpensive.



Source: SICK TiM571



# Depth sensors for automation



Source: Microsoft, <https://azure.microsoft.com/en-us/campaigns/kinect/>

- Microsoft's most recent ToF camera: Kinect for Azure





# Depth sensors for automation



Source: ifm electronic

- Autonomous cleaning robots is another application (this one is for Deutsche Bahn).



# 2D Lidar

- SICK LMS291

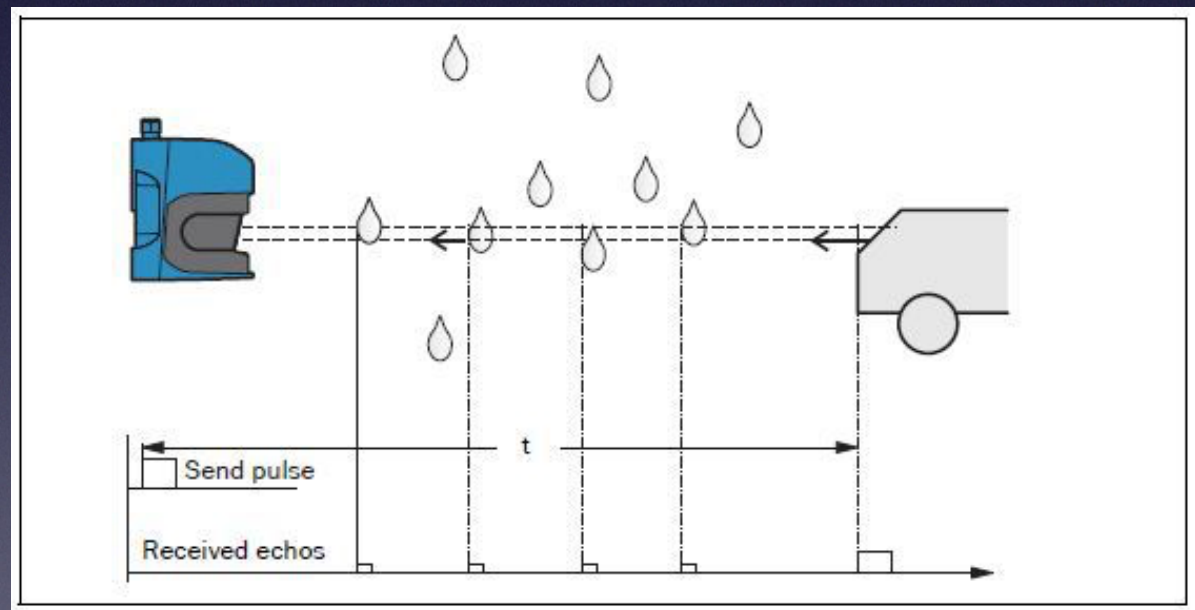
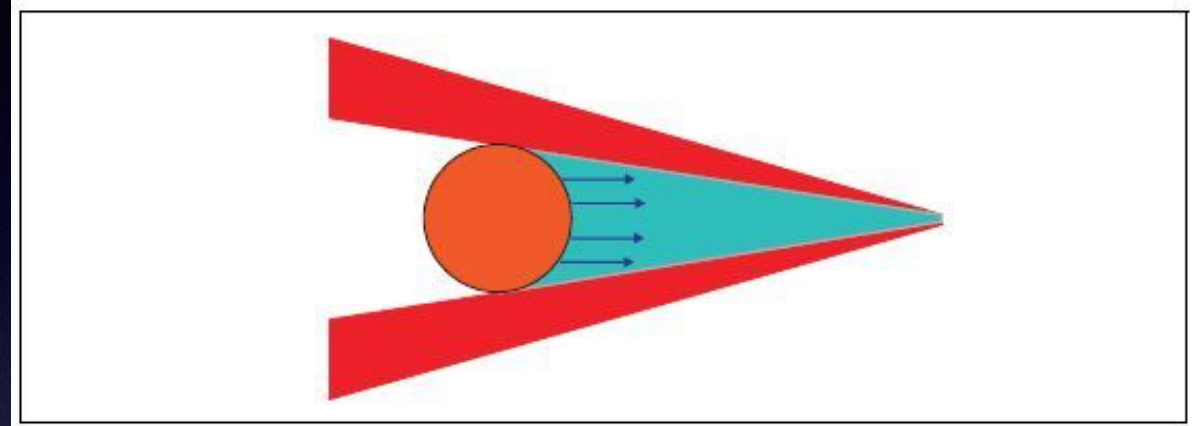


Source: David Kohanbash:  
<http://robotsforroboticists.com/sick-lms-lidar-teardown/>



# 2D Lidar

- Uses direct ToF measurement using an accurate clock that is read out triggered by an SPAD detector.
- Challenges for Lidar:
  - Dot size grows with range.
  - Small objects cause multiple echos. Mitigated on recent sensors using **multiple return**: Two or more echoes and their strengths are recorded in each direction.
  - Motion distortion



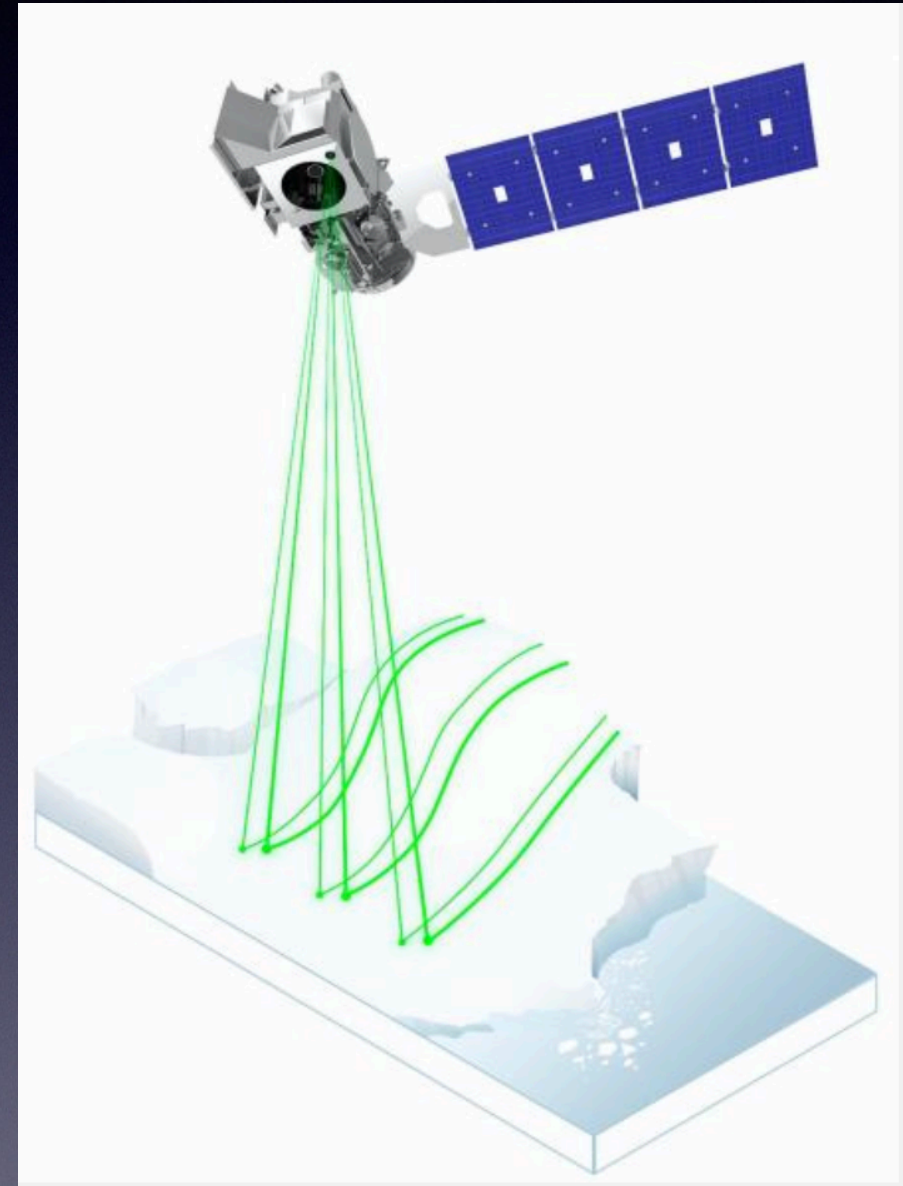
Source: SICK

+ One light direction at a time  $\Rightarrow$  no multipath interference.



# 2D Lidar

- LIDAR even works from space!
- NASA's ICESat-2 is used to map Antarctica's ice.  
(has high reflectance)
- Swath of 6 beams
- ICESat2 has cm range accuracy  
(but a large spatial average)
- For more detailed elevation maps stereo is used (Maxar in Linköping)





# 2D Lidar

- 3D Lidar evolved from 2D Lidars mounted on cars
- SICK LMS at 2005 DARPA challenge



Stanford "Stanley". The car that won the 2005 DARPA challenge



# 3D Lidar

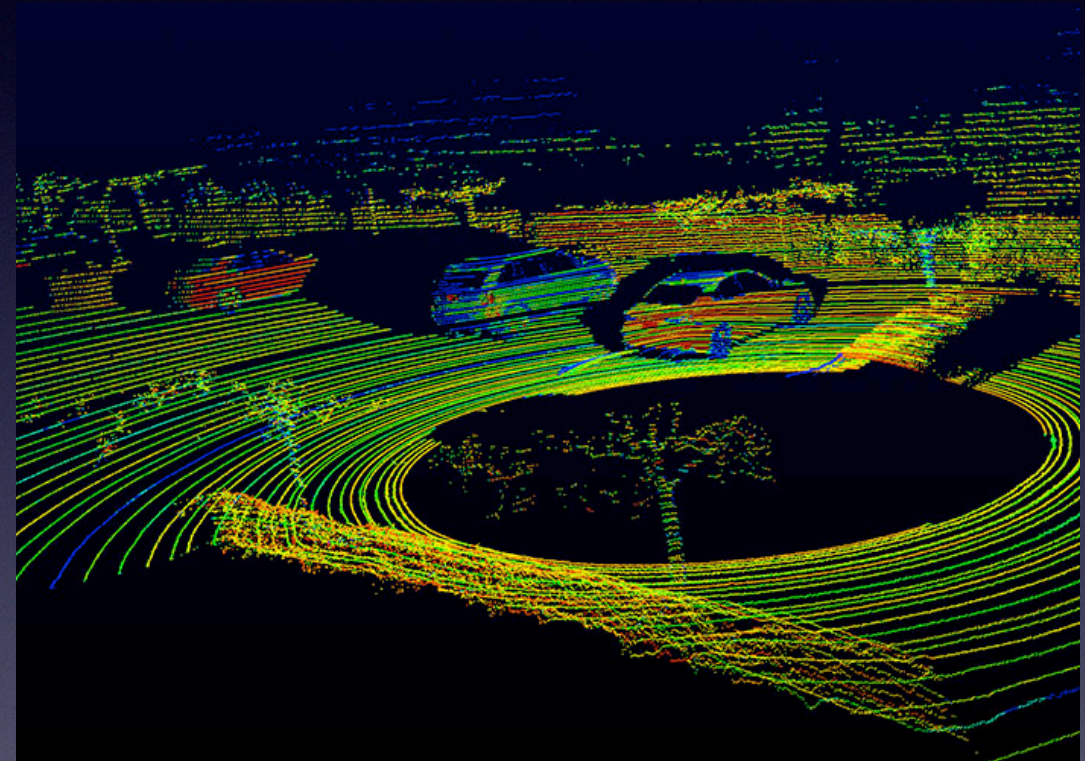
- **2007-now:** Velodyne HDL-64E



<https://www.cvlibs.net/datasets/kitti/>



<https://velodynelidar.com/hdl-64e.html>



- A 64-line ToF sensor that scans 360°



# 3D Lidar

- The Velodyne principle: rotate entire sensor package



Source: Velodyne

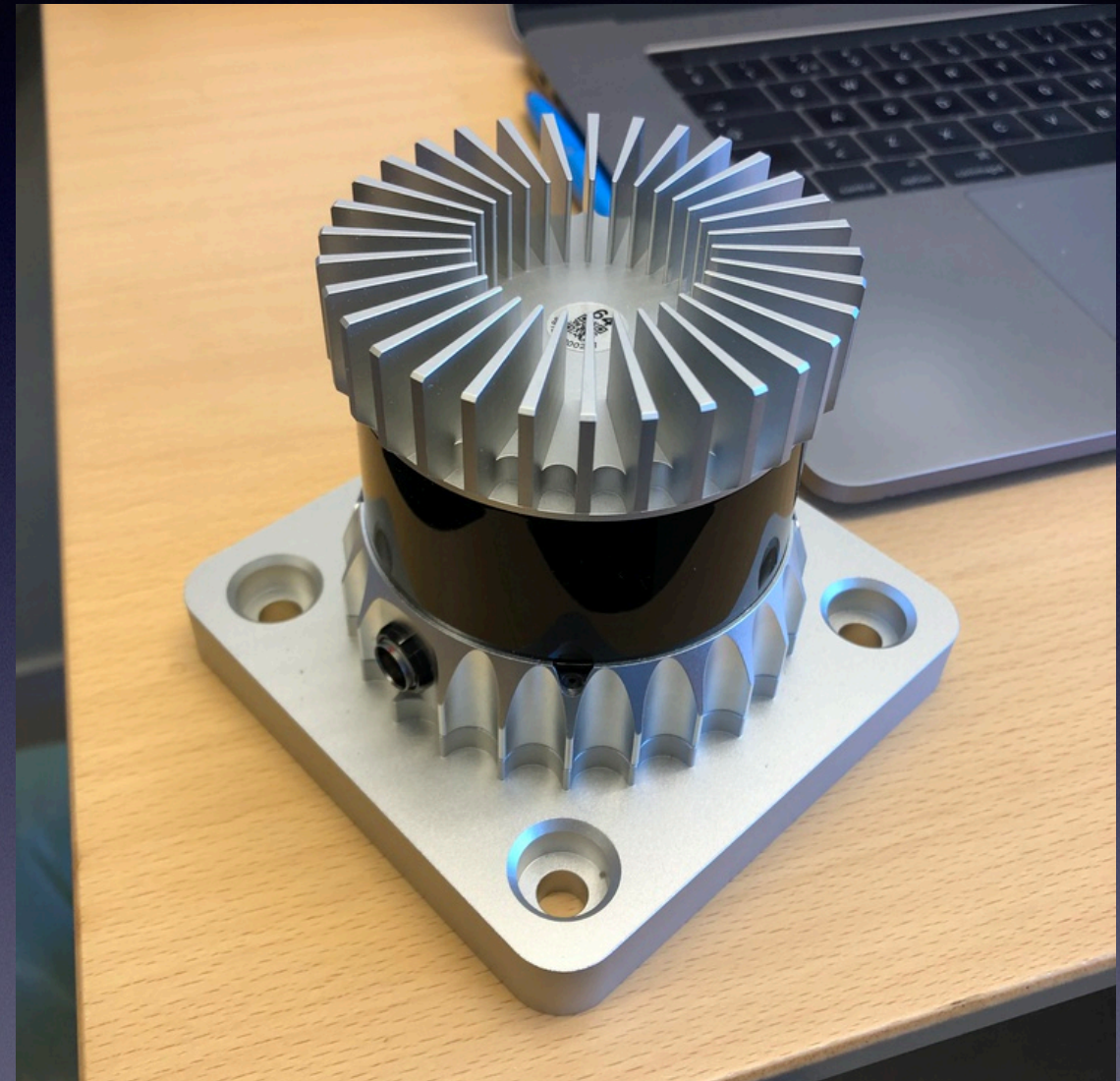


Source: Matt McFarland,  
<https://www.youtube.com/watch?v=klrMaPTENB0>



# The Ouster OS-1 64

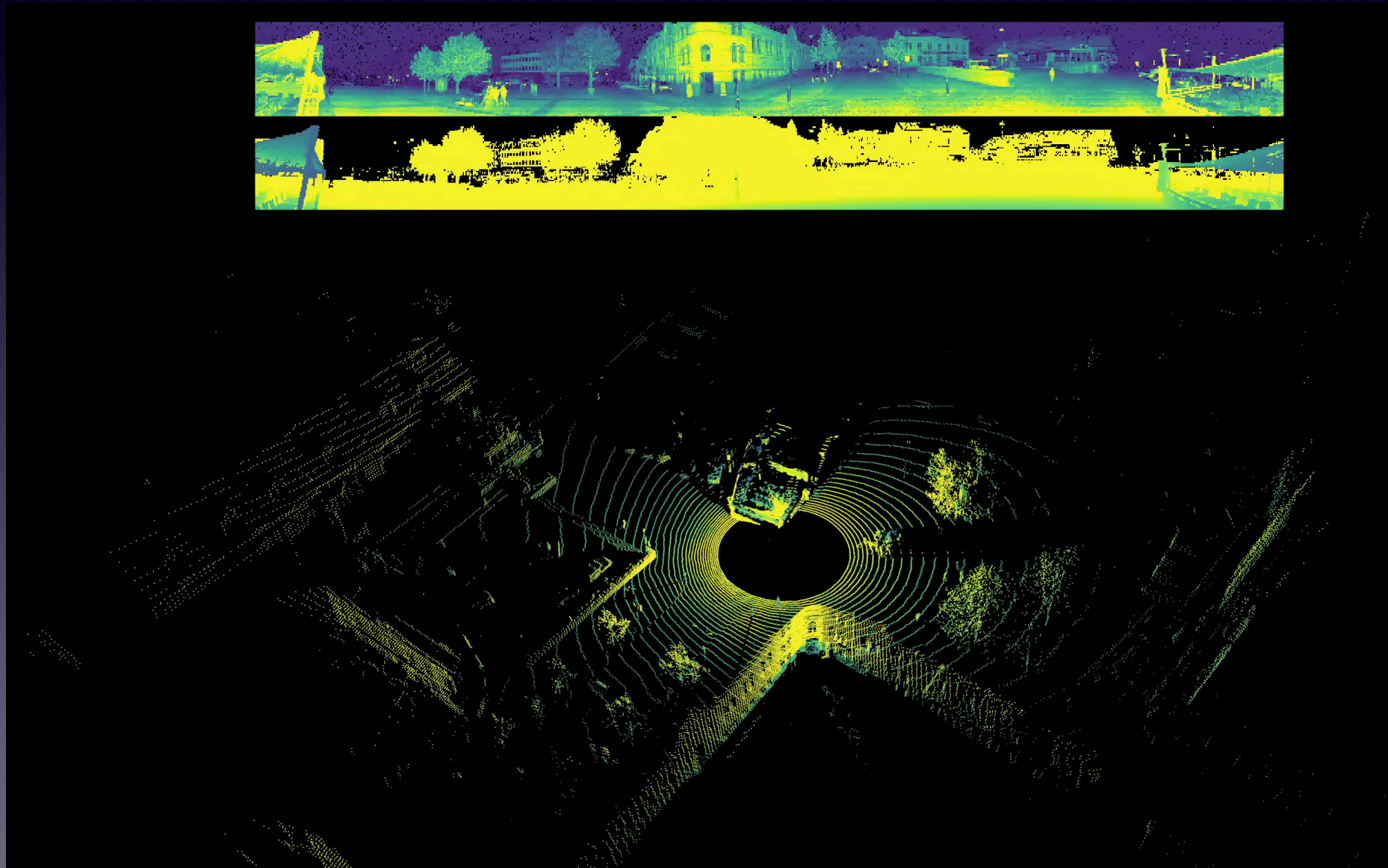
- Size WxH = 8.5x7.3cm  
Weight 396g
- 64 lines  
2048 directions @10Hz  
33.2°x 360°
- 6 axis IMU (Gyro,Acc)  
@100Hz
- Range: 105m or more.





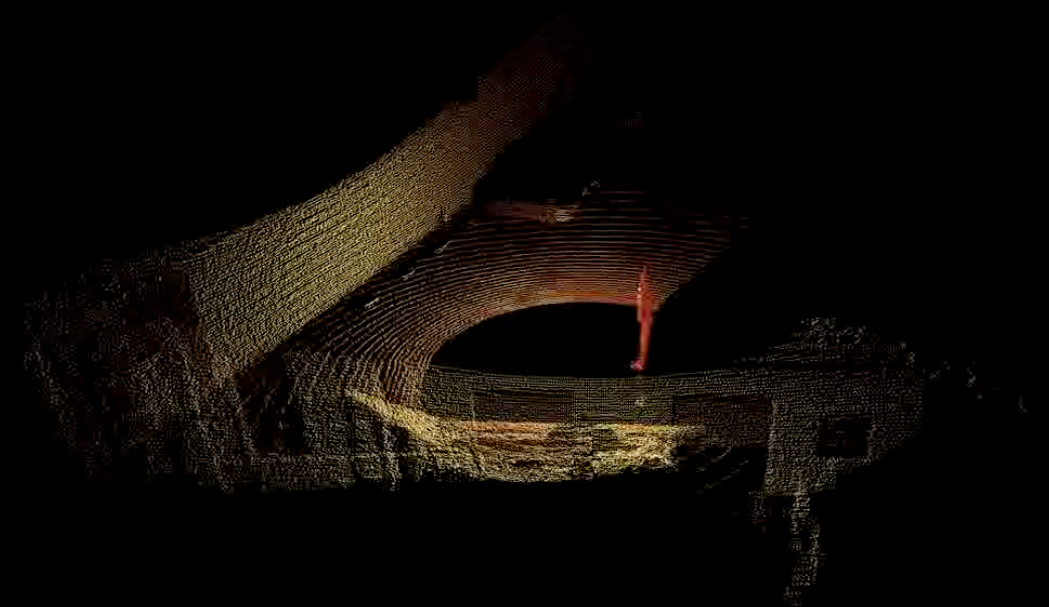
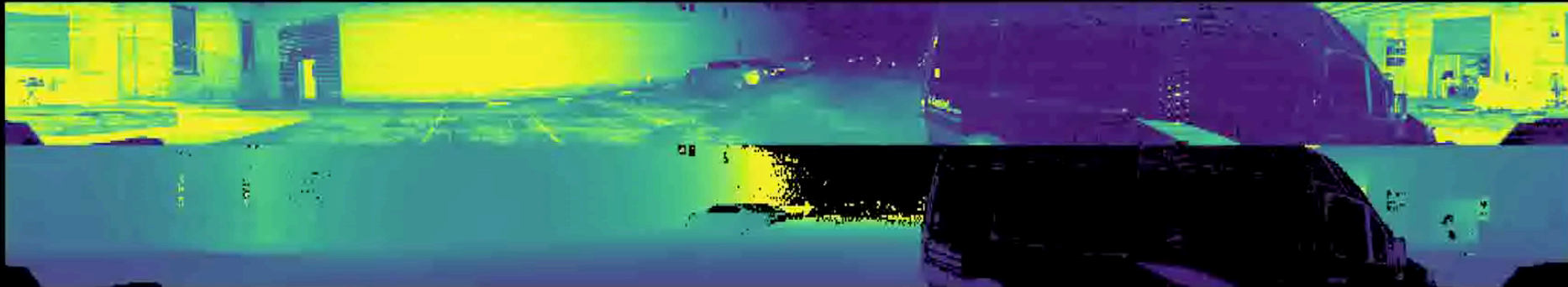
# 3D Lidar

- Hand-held Ouster OS1-64 Lidar in Västervik





# Mapping with 3D Lidar



<https://www.youtube.com/watch?v=4QYnqbO1eT0>



# Terrestrial Laser Scanners

- Terrestrial Laser Scanner (TLS)
- Sensor is placed on a tripod at a number of selected locations. One scan takes several minutes.
- Used when accuracy is important, and speed less so. mm range resolution. E.g. 1200 x 2048 points per scan (pan,tilt).
- Many different variants, but usually:
  - a tilting mirror for vertical resolution
  - a step motor for horizontal resolution
  - a colour camera gives a spherical panorama, and an RGB value for each depth point.



Image source: Wikimedia Commons  
[https://commons.wikimedia.org/wiki/File:Lidar\\_P1270901.jpg](https://commons.wikimedia.org/wiki/File:Lidar_P1270901.jpg)



# Terrestrial Laser Scanners

## Time-of-flight (TOF) method



Leica RTC360



Optech Polaris



Riegl VZ400i



Topcon GLS-2000



Trimble X7

## Phase based method



Artec Ray / Surphaser



FARO Focus<sup>S</sup>



Leica RTC360



Z+F Imager 5016

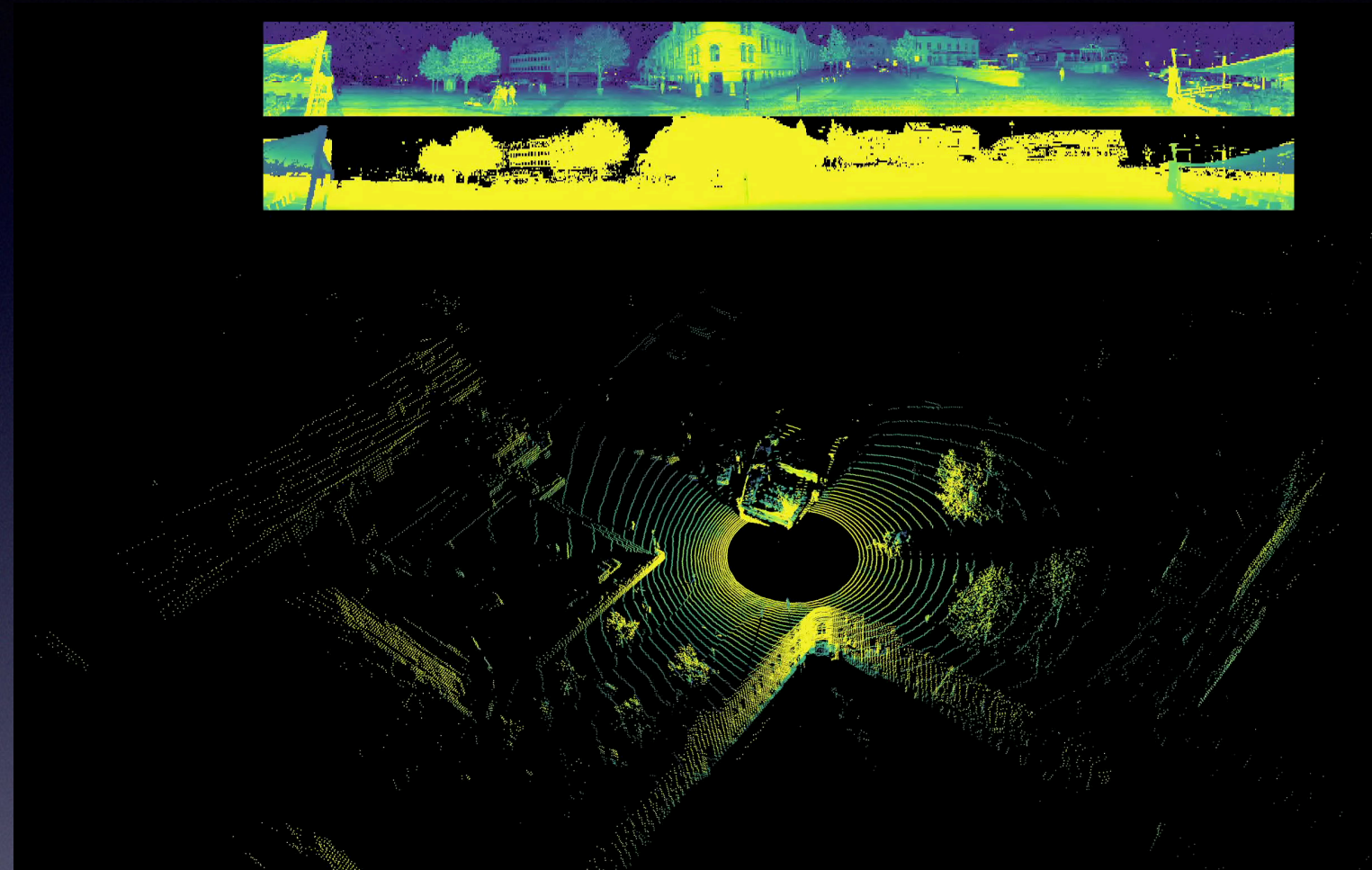
<https://www.laserscanning-europe.com/en/servicesdevices/terrestrial-laser-scanners>

- The different scans are then stitched together using Point Set Registration.
- Applications include: Forensic scene documentation, Archeaology, Digital Twins of Industrial installations, Civil Engineering, Surveying.



# From range to 3D

- The output from the OS-1 Lidar is a **reflectance image** (top) with  $64 \times 2048$  pixels  $i_{kl}$ , and a **range image** (middle) with pixels  $r_{kl}$ .
- The **point cloud** (bottom) is calculated from these and sensor parameters.
- Note the wobble in the video, we will now analyze this.





# From range to 3D

- To unpack the *range image*  $r_{kl}$  to points in 3D do:

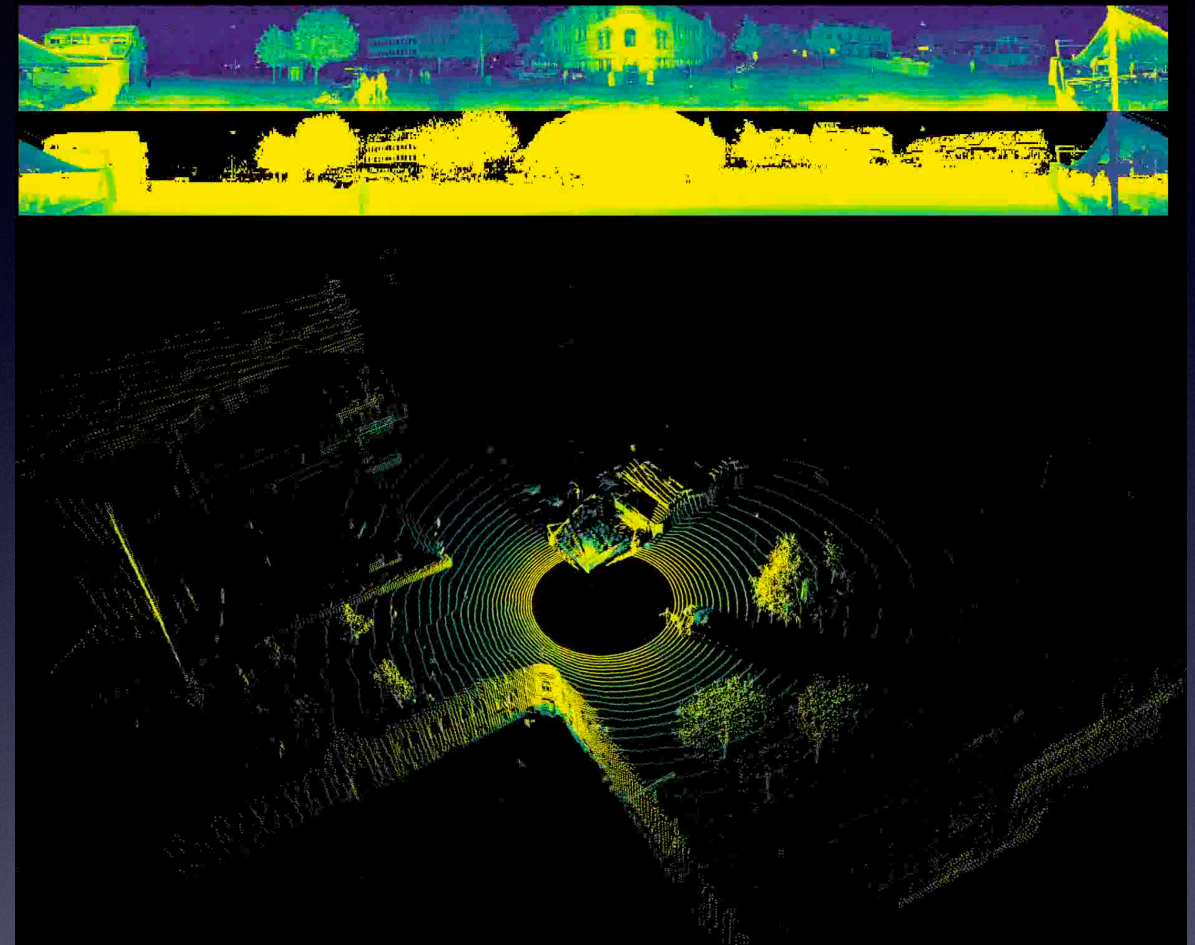
$$\mathbf{X}_k^{\text{raw}}(t_l) = r_{kl} \hat{\mathbf{x}}_k$$

- 64 ray directions from sensor calibration table:

$$\hat{\mathbf{x}}_k = \begin{bmatrix} \cos \theta_k \cos \phi_k \\ -\sin \theta_k \cos \phi_k \\ \sin \phi_k \end{bmatrix}$$

- Next apply the sensor rotation:

$$\begin{bmatrix} \mathbf{X}_k^{\text{car}}(t_l) \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R}(t_l) & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} \mathbf{X}_k^{\text{raw}}(t_l) \\ 1 \end{bmatrix}$$





# From range to 3D

- To unpack the *range image*  $r_{kl}$  to points in 3D do:

$$\mathbf{X}_k^{\text{raw}}(t_l) = r_{kl} \hat{\mathbf{x}}_k$$

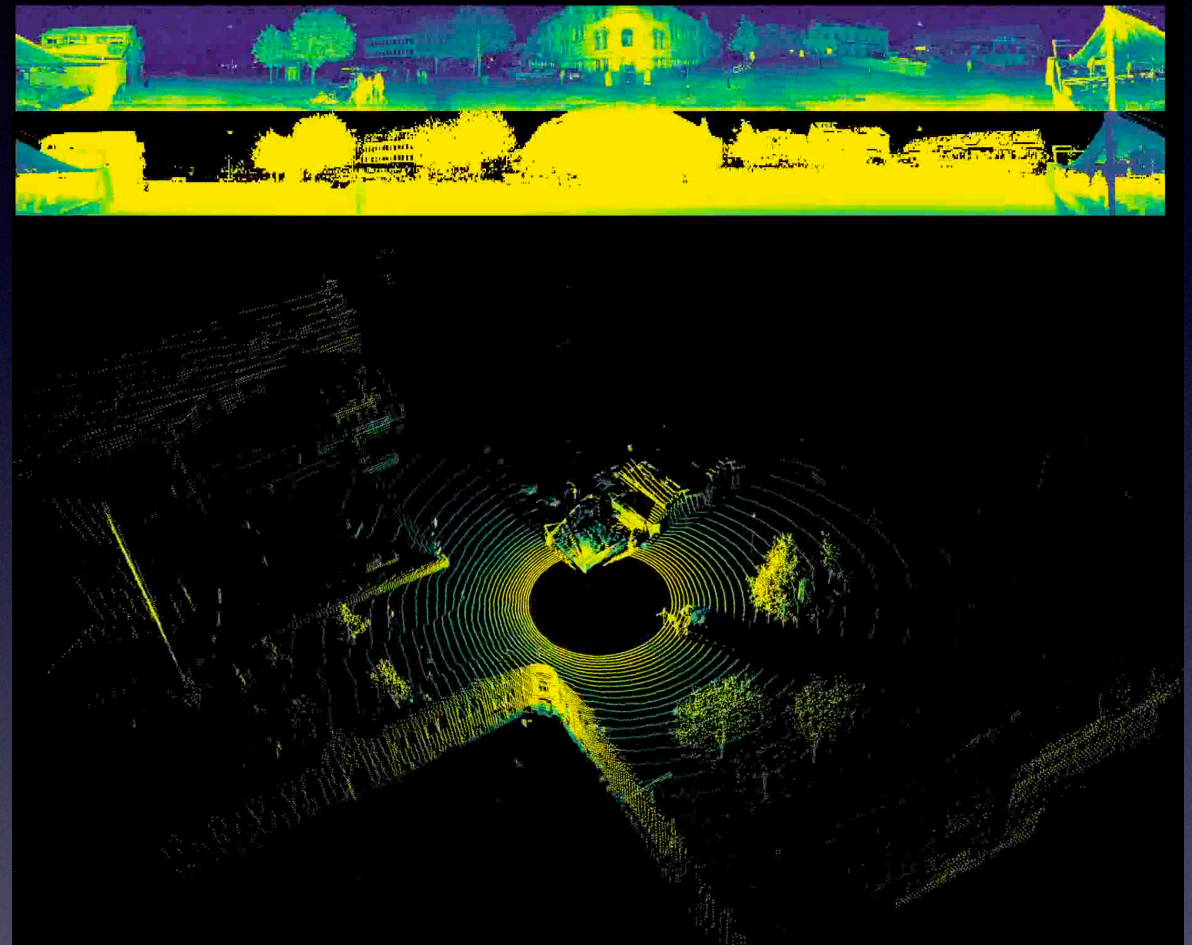
- 64 ray directions from sensor calibration table:

$$\hat{\mathbf{x}}_k = \begin{bmatrix} \cos \theta_k \cos \phi_k \\ -\sin \theta_k \cos \phi_k \\ \sin \phi_k \end{bmatrix}$$

- Next apply the sensor rotation:

$$\begin{bmatrix} \mathbf{X}_k^{\text{car}}(t_l) \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R}(t_l) & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} \mathbf{X}_k^{\text{raw}}(t_l) \\ 1 \end{bmatrix}$$

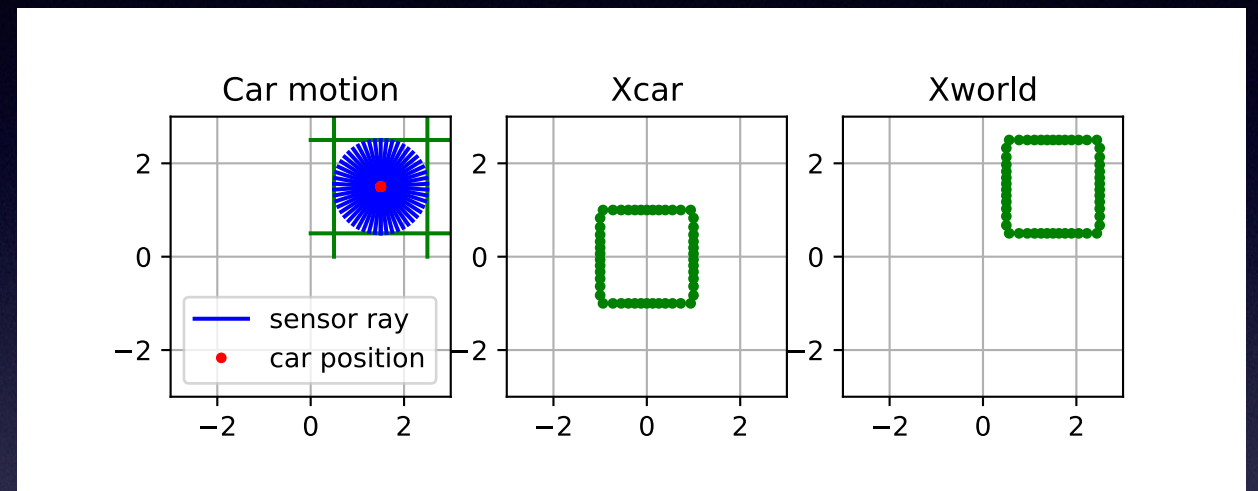
Note:  
One transformation  
per scan line,  $l$



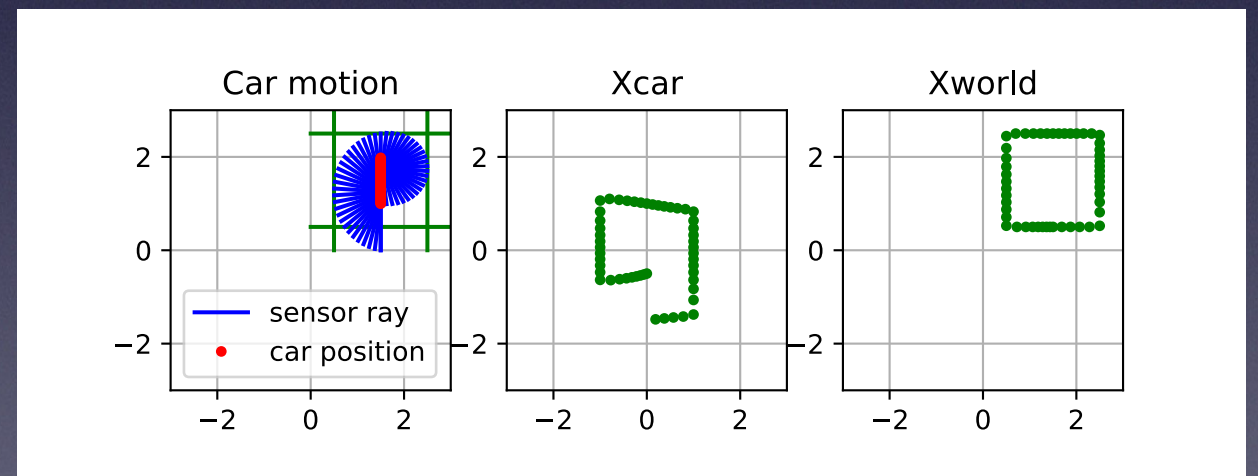


# Motion distortions

- Unpacking to car coordinates is sufficient in the stationary case



- But we get motion distortions if the car itself is also moving



- These can be corrected if we know the car motion.



# Motion distortion correction

- A car motion estimate  $(\Theta(t), \mathbf{p}(t))$  can be obtained from the IMU (accelerometers, gyroscopes) built into the car or the Lidar. Alternatively from a preliminary scan registration.

$$\begin{bmatrix} \mathbf{X}_{\text{world}}(t) \\ 1 \end{bmatrix} = \mathbf{T}_{12i}^{-1} \begin{bmatrix} \Theta(t) & \mathbf{p}(t) \\ \mathbf{0}^T & 1 \end{bmatrix} \mathbf{T}_{12i} \begin{bmatrix} \mathbf{X}_{\text{car}}(t) \\ 1 \end{bmatrix}$$

- Note that the transformation from the Lidar to the IMU,  $\mathbf{T}_{12i}$  needs to be known. It is normally fixed and found through calibration.



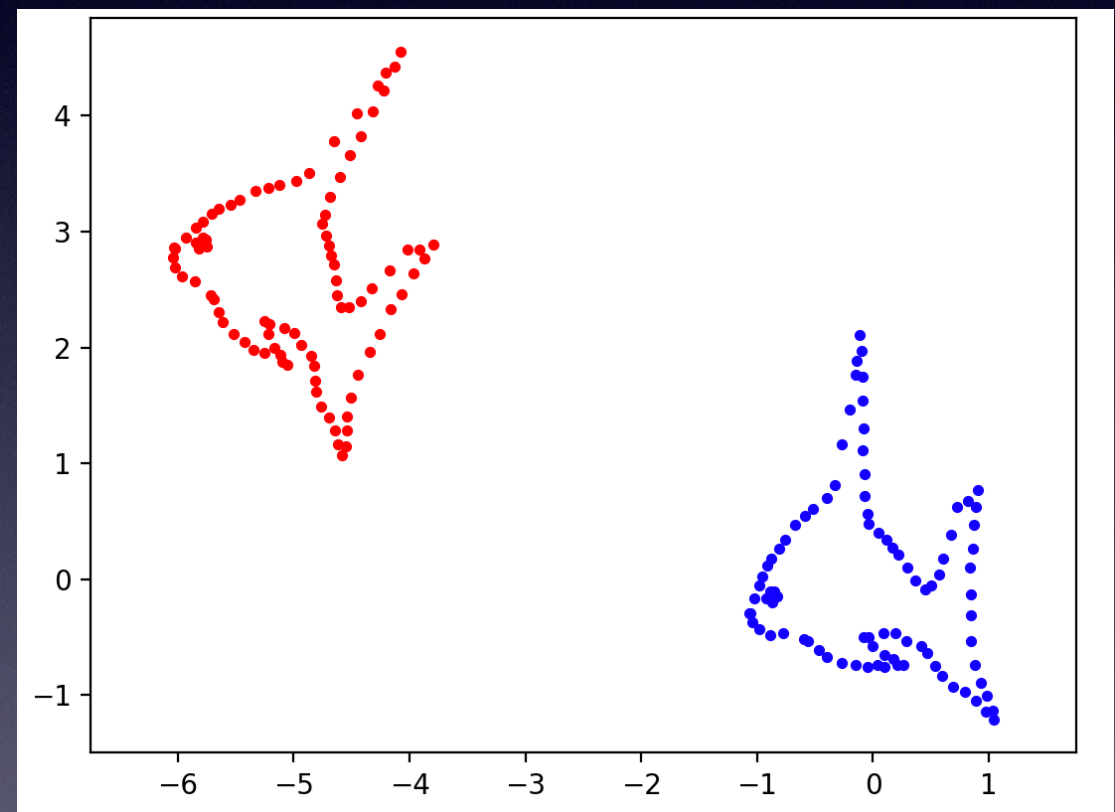
# Point-set registration

- Mapping with 3D sensors use *point-set registration*.

Goal: Find a transformation

$$\mathbf{Y}'_k = \mathbf{R}\mathbf{X}_k + \mathbf{t}$$

- That moves a set of points  $\{\mathbf{X}_k\}_{k=1}^K$
- to be aligned with another set of points  $\{\mathbf{Y}_l\}_{l=1}^L$
- ICP Classical method [Chen&Medioni ICRA91]

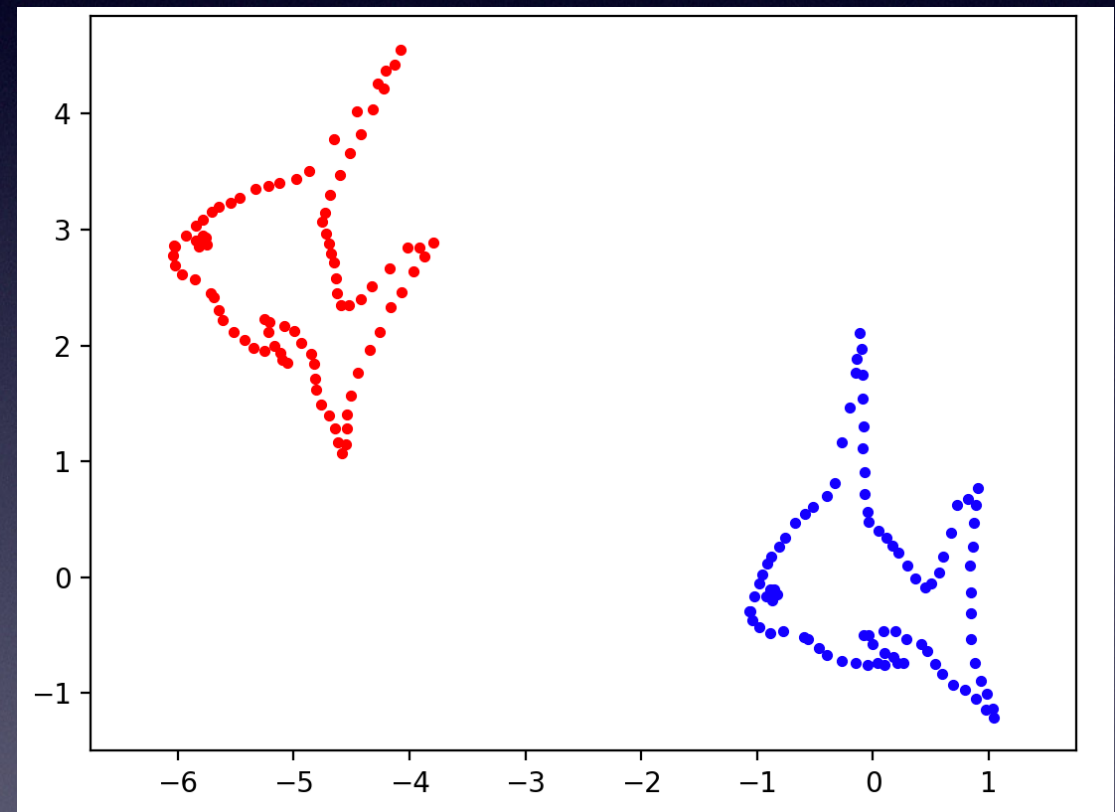




# Point-set registration

- Iterated Closest Point (ICP) [Chen&Medioni ICRA91]

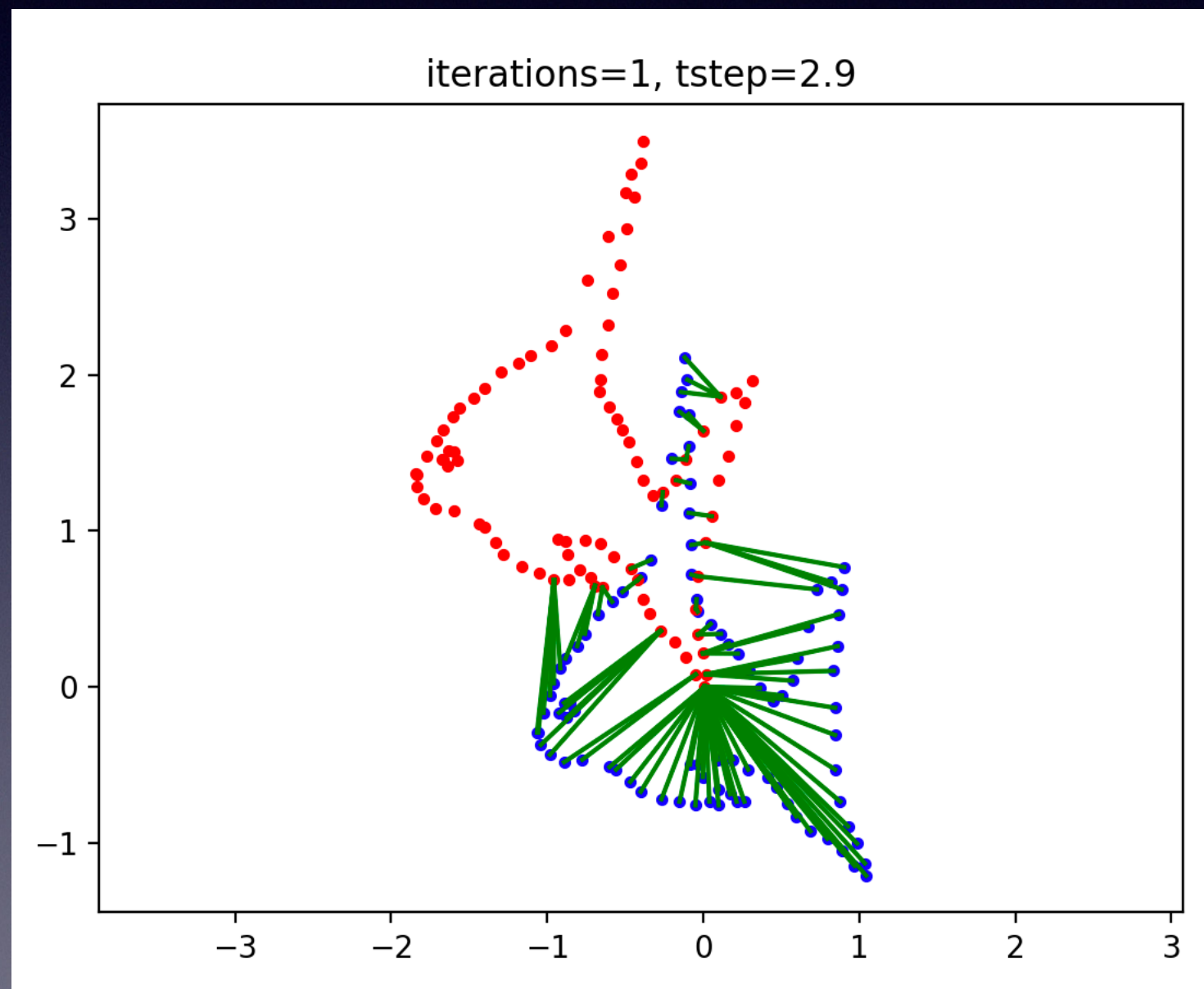
1. **Guess correspondence**  
For each point: find the closest point in the other set
2. **Align correspondences**  
Center and solve the Orthogonal Procrustes Problem (OPP) to find  $R$ , and  $t$
3. **Apply transformation** and goto 1.





# Point-set registration

- ICP demo 2D





# Align correspondences

- Rigid point-set registration uses an Orthogonal Procrustes Problem (OPP) at its core:

$$\{\mathbf{R}, \mathbf{t}\} = \arg \min_{\mathbf{R}, \mathbf{t}} \|\mathbf{X} - \mathbf{R}\mathbf{Y} + \mathbf{t}\|^2$$

- Two point-sets  $\mathbf{X}$  and  $\mathbf{Y}$  (columns  $\mathbf{X}_{:,n}$  are points)

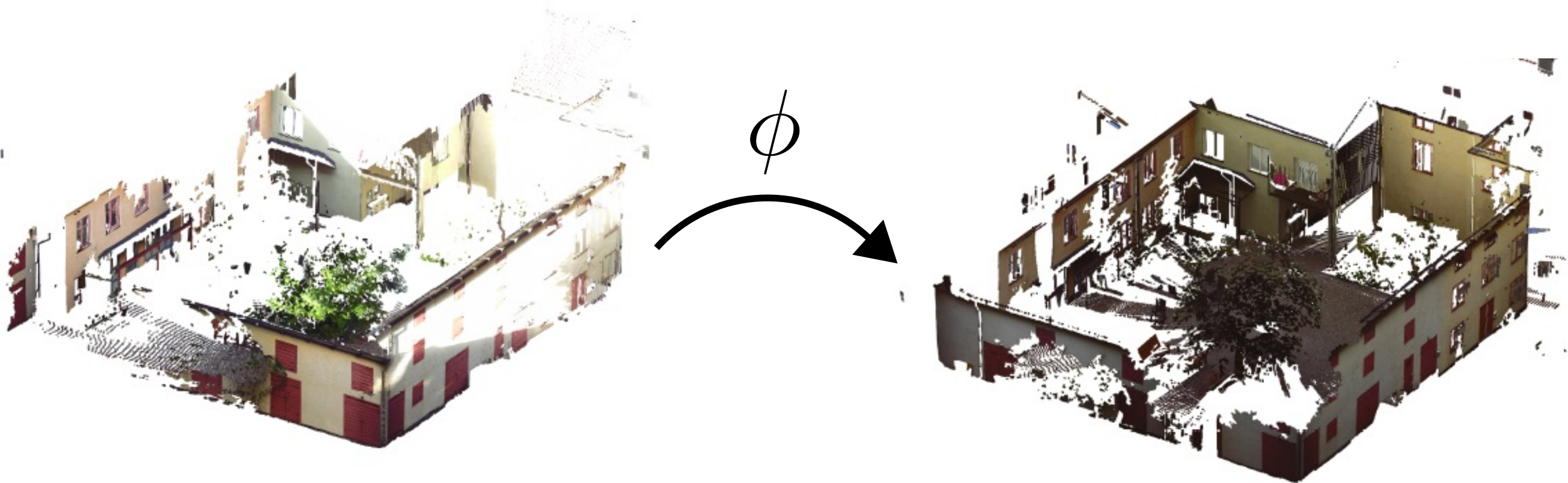
$$\begin{aligned} \mathbf{X}_m &= \mathbf{X} - \mathbf{X}_\mu & \mathbf{X}_\mu &= \frac{1}{N} \sum_{n=1}^N \mathbf{X}_{:,n} \\ \mathbf{Y}_m &= \mathbf{Y} - \mathbf{Y}_\mu \end{aligned}$$

- OPP:  $\mathbf{R} = \arg \min_{\mathbf{R}} \|\mathbf{X}_m - \mathbf{R}\mathbf{Y}_m\|^2$

- Translation:  $\mathbf{t} = \mathbf{X}_\mu - \mathbf{R}\mathbf{Y}_\mu$



# Point-set registration



$$\phi(\mathbf{x}) = R\mathbf{x} + \mathbf{t}$$

Illustration by Martin Danelljan



# Summary

- ToF cameras and Lidars both use the *time-of-flight* principle to sense depth.
- ToF cameras work well indoors, where they have better *range and spatial detail* than structured light and fringe pattern projection.
- Lidars work better outdoors, and avoid *multipath interference*, but can instead have *motion distortions*.
- *Point-set registration* (PSR) is used to create large 3D models from 3D cameras.
- Next lecture: Improvements of PSR