Robot, lidar, RGB+D camera

Karel Zimmermann

http://cmp.felk.cvut.cz/~zimmerk/

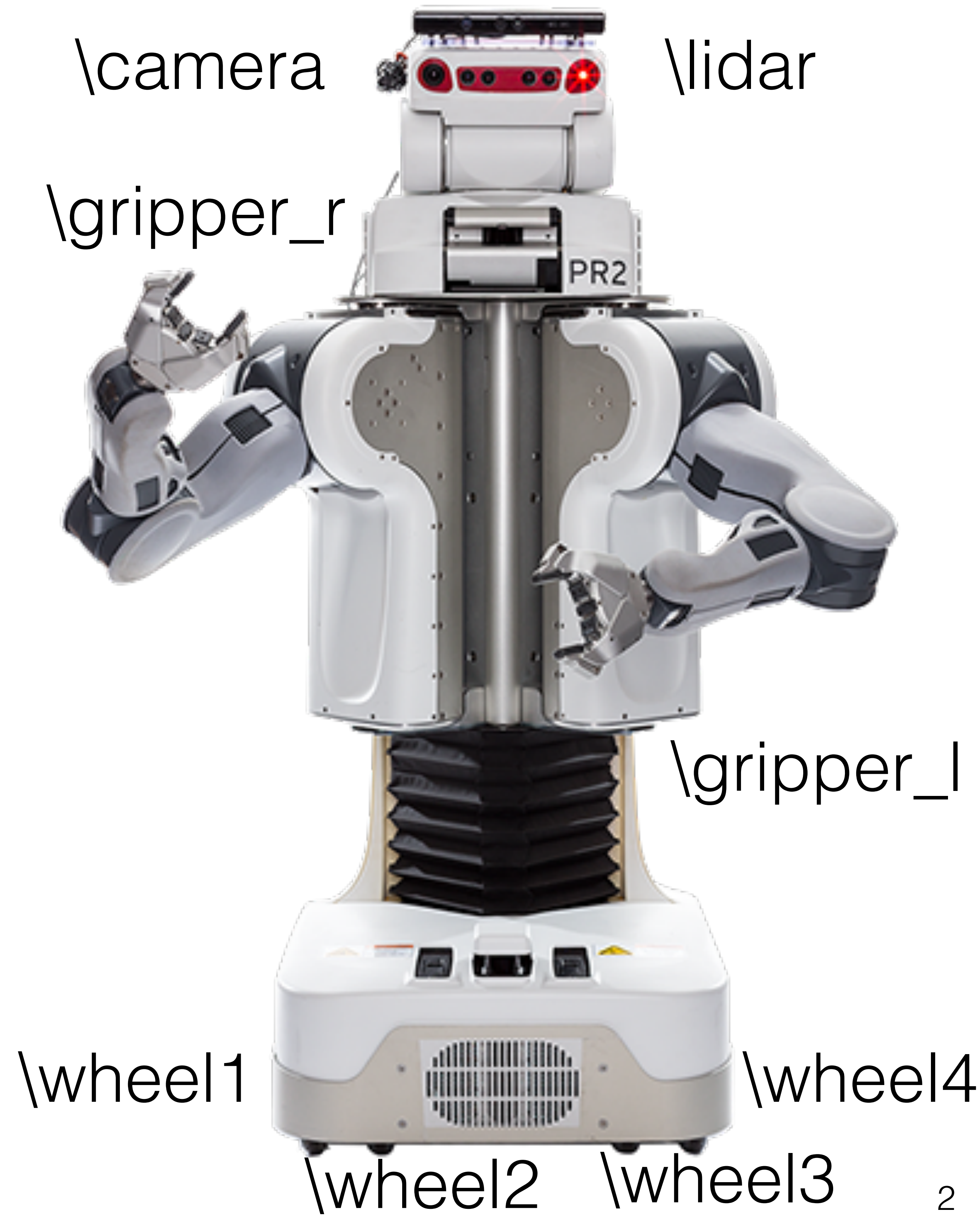
Vision for Robotics and Autonomous Systems
https://cyber.felk.cvut.cz/vras/

Center for Machine Perception
https://cmp.felk.cvut.cz

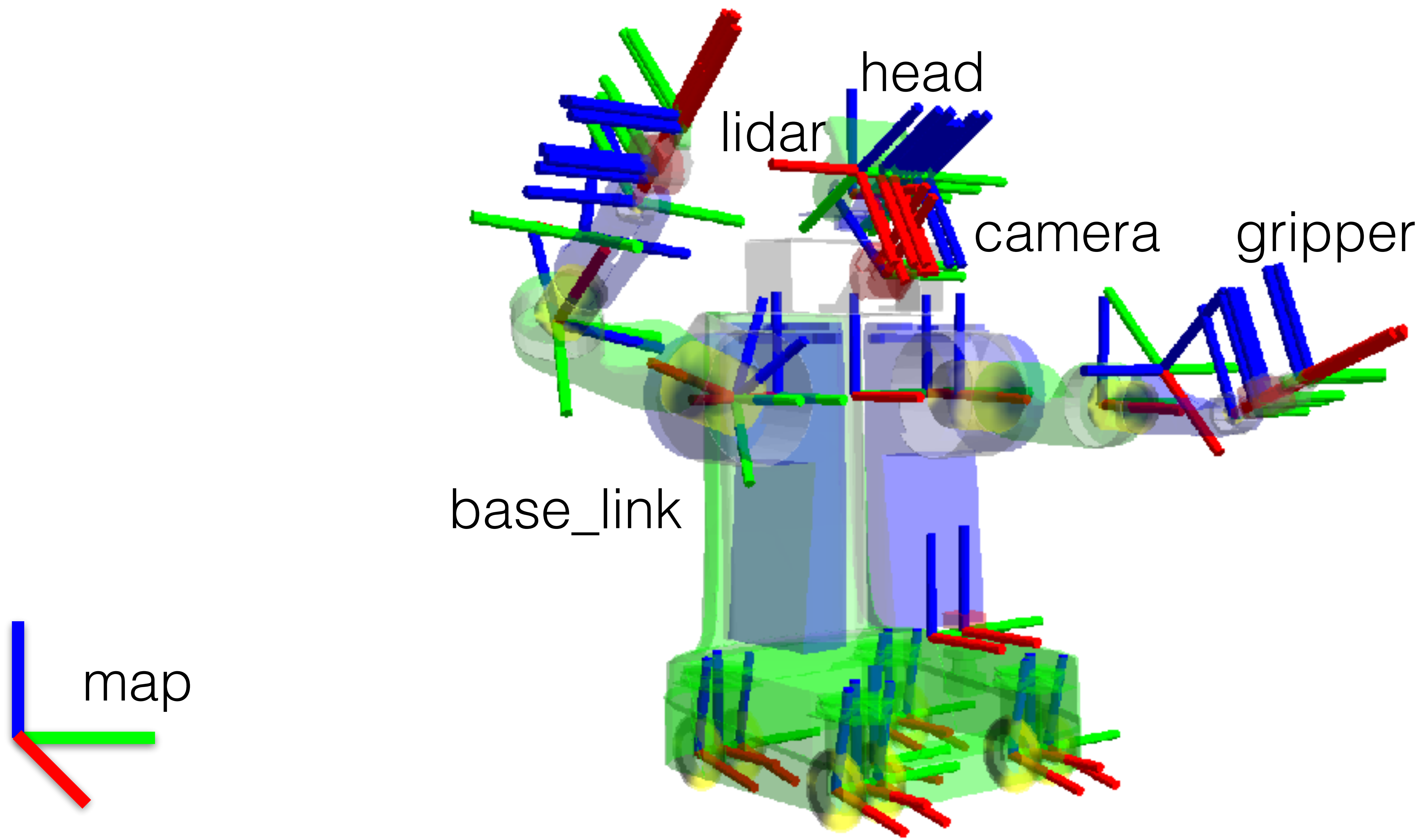
Department for Cybernetics
Faculty of Electrical Engineering
Czech Technical University in Prague

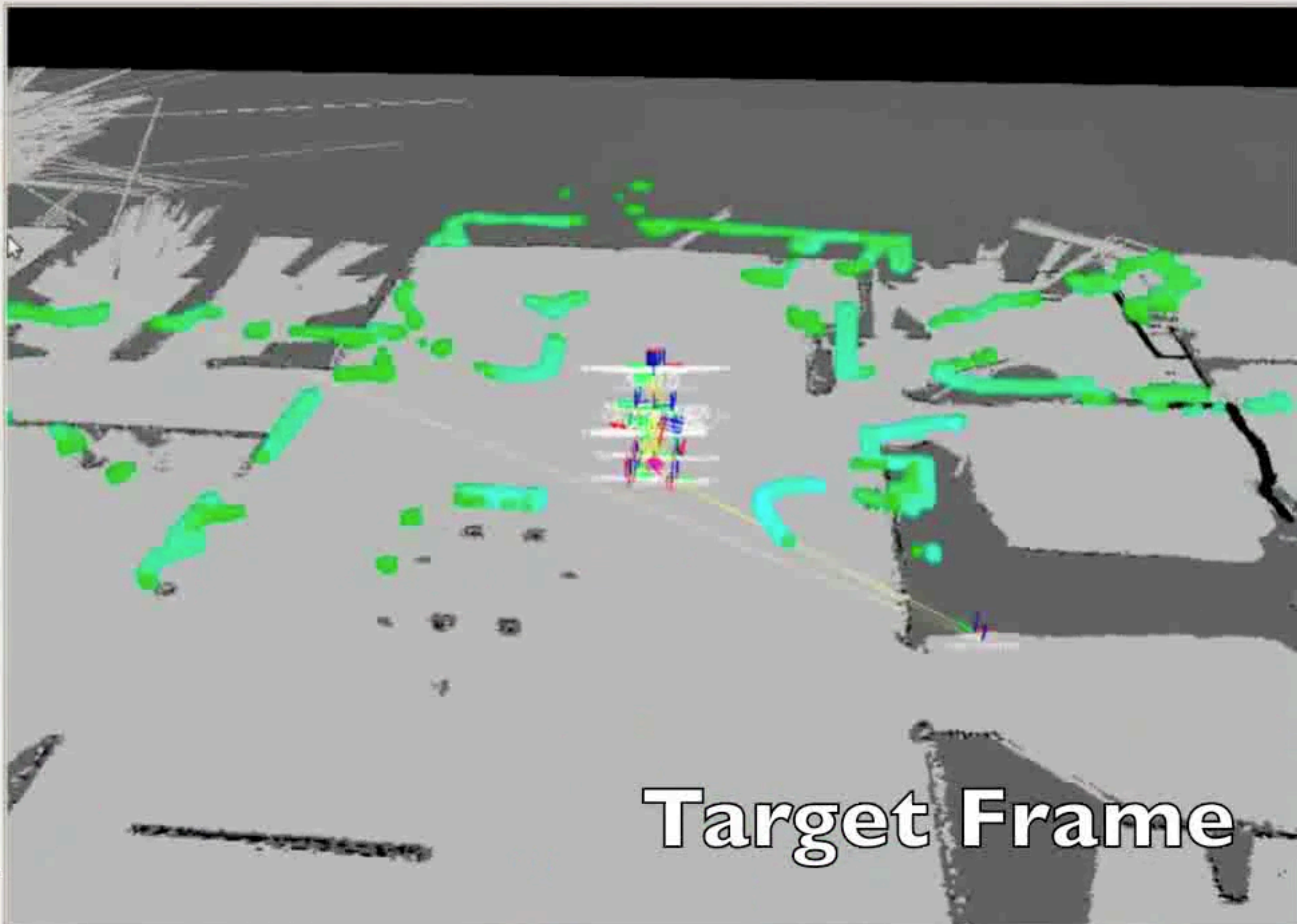
Why transformation among coordinate frames are important?

- Robot consist of many distributed components (sensors, actuators, joints)
- Each component operates in its own coordinate frame
- Robot moves => each coordinate frames changes in time.



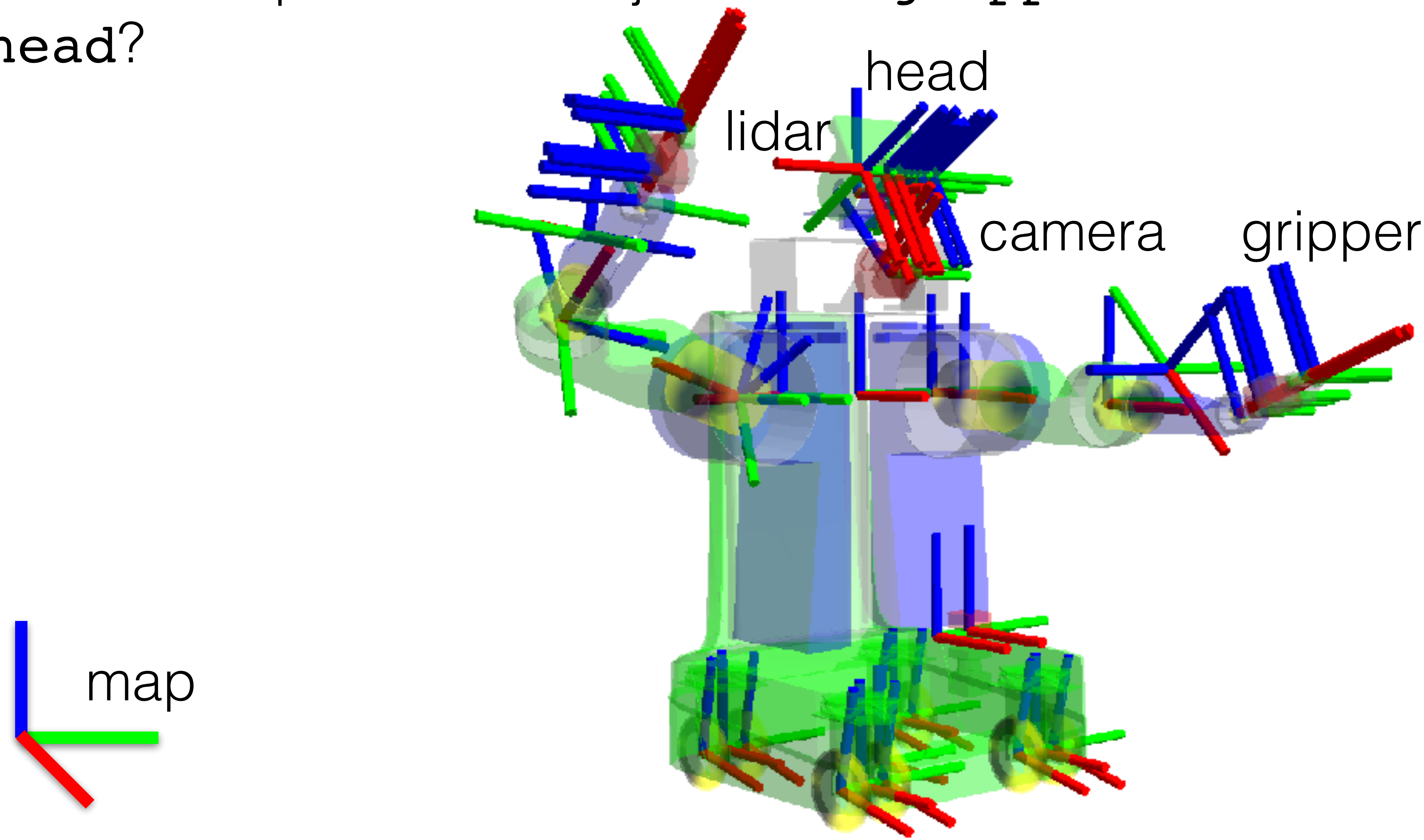
- Each coordinate frame define 3D Euclidean space.
- It is uniquely determined by its name.




Target Frame

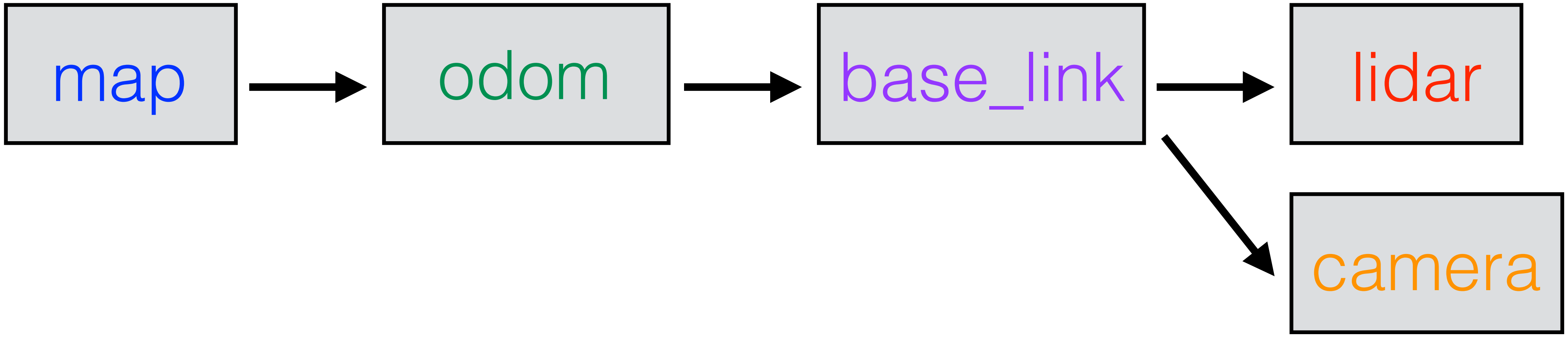
In arbitrary time we would like to answer questions like that:

- What is the pose of the **`head`** in the **`map`**?
- What color from **`camera`** corresponds to 3D points measured by **`lidar`**?
- What is the pose of the object in the **`gripper`** relative to **`head`**?



Coordinate frames & ROS

- Coordinate frames in ROS form a tree

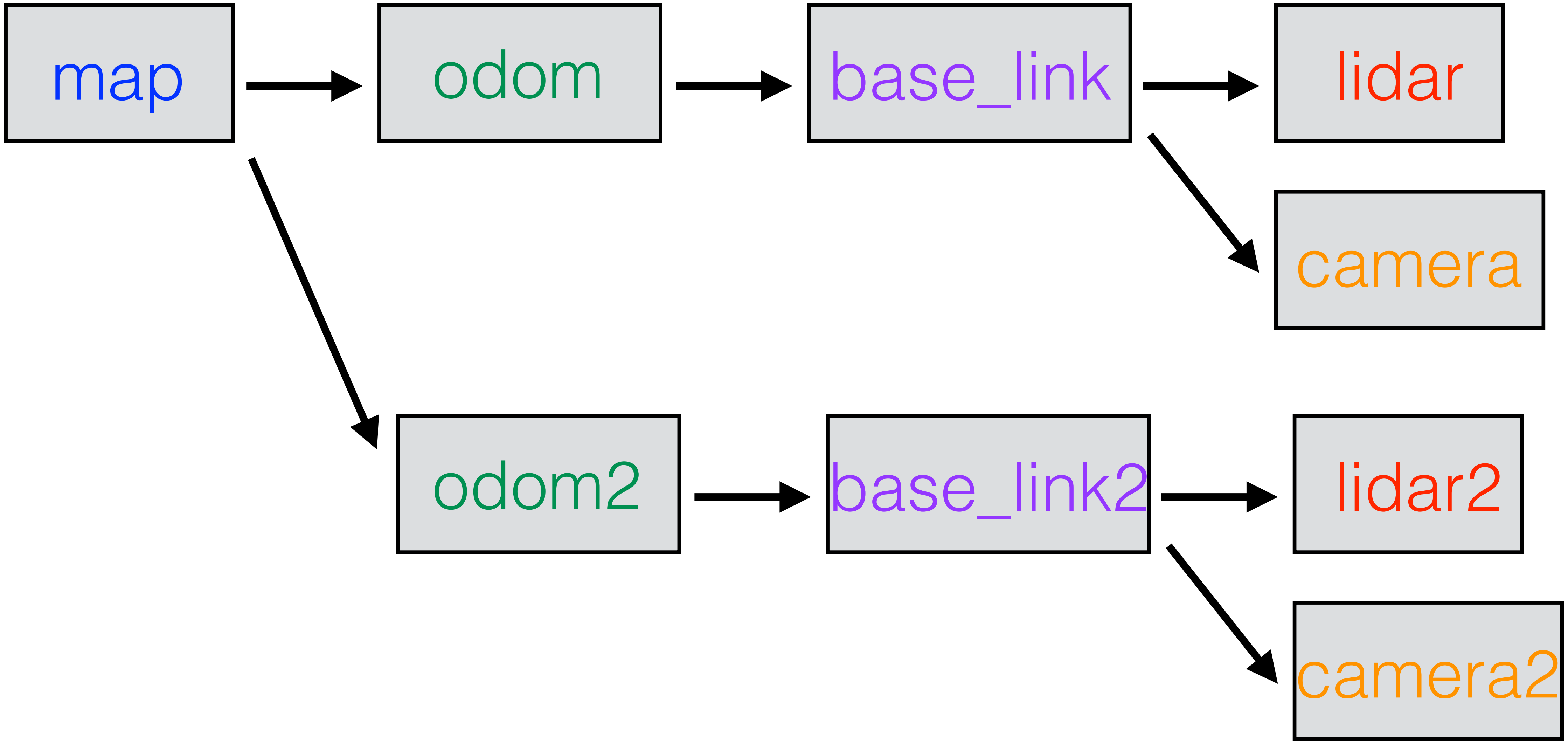


- If parent-child transformations are published by your nodes => you can access arbitrary transformation

```
$ rosrun tf tf_echo /map /lidar
At time 1263248513.809
- Translation: [2.398, 6.783, 0.000]
- Rotation: in Quaternion [0.000, 0.000, -0.707, 0.707]
```

Coordinate frames & ROS

- Transformation tree for two robots allow to estimate mutual position and use measurement from other robot.



Broadcasting static transformation between two c.f. in ROS

```
broadcaster = tf2_ros.StaticTransformBroadcaster()
transform = geometry_msgs.msg.TransformStamped()
# estimate R,t (e.g. measure or compute)
# … "TOPIC OF FOLLOWING TWO LECTURES" => R,t
# convert rotation matrix into quaternion
q = mat2quat(R)
# fill-in transform between coordinate frames (q,t)
transform.translation.x = t[0]
transform.translation.y = t[1]
transform.translation.z = t[2]
transform.rotation.x = q[0]
transform.rotation.y = q[1]
transform.rotation.z = q[2]
transform.rotation.w = q[3]
transform.header.stamp = rospy.Time.now()
transform.header.frame_id = "base_link"
transform.child_frame_id = "lidar"

# publish transform between coordinate frames (q,t)
broadcaster.sendTransform(transform)
```

Listening static transformation between two c.f. in ROS

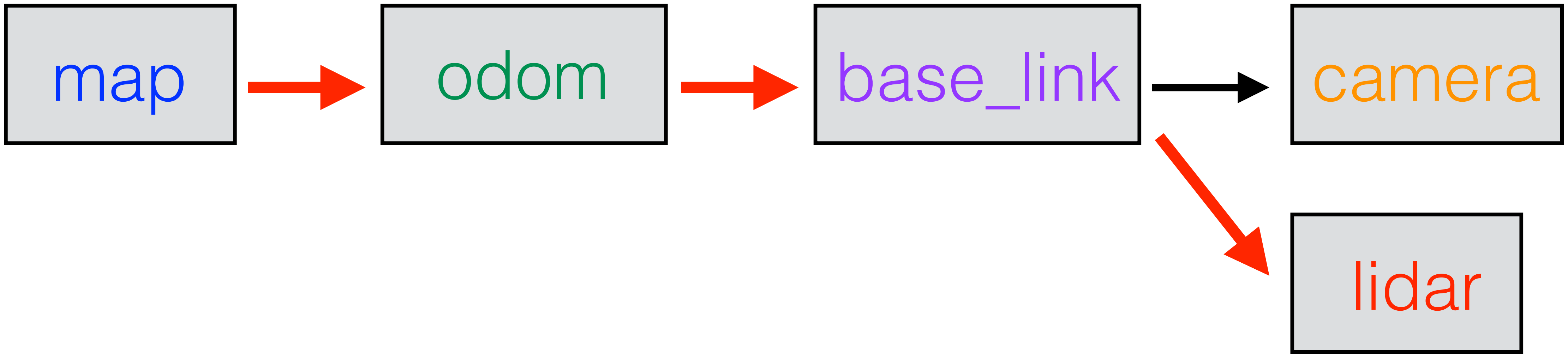
(1) in the your own node:

```
# initialize listener (10 sec buffer)
buffer = tf2_ros.Buffer()
listener = tf2_ros.TransformListener(buffer)

# estimate transformation from lidar to map
transform = buffer.lookup_transform('lidar','map',rospy.Time())
```

(2) In the terminal:

```
$ rosrun tf tf_echo /map /lidar
```



Outline

The topic of this lecture:

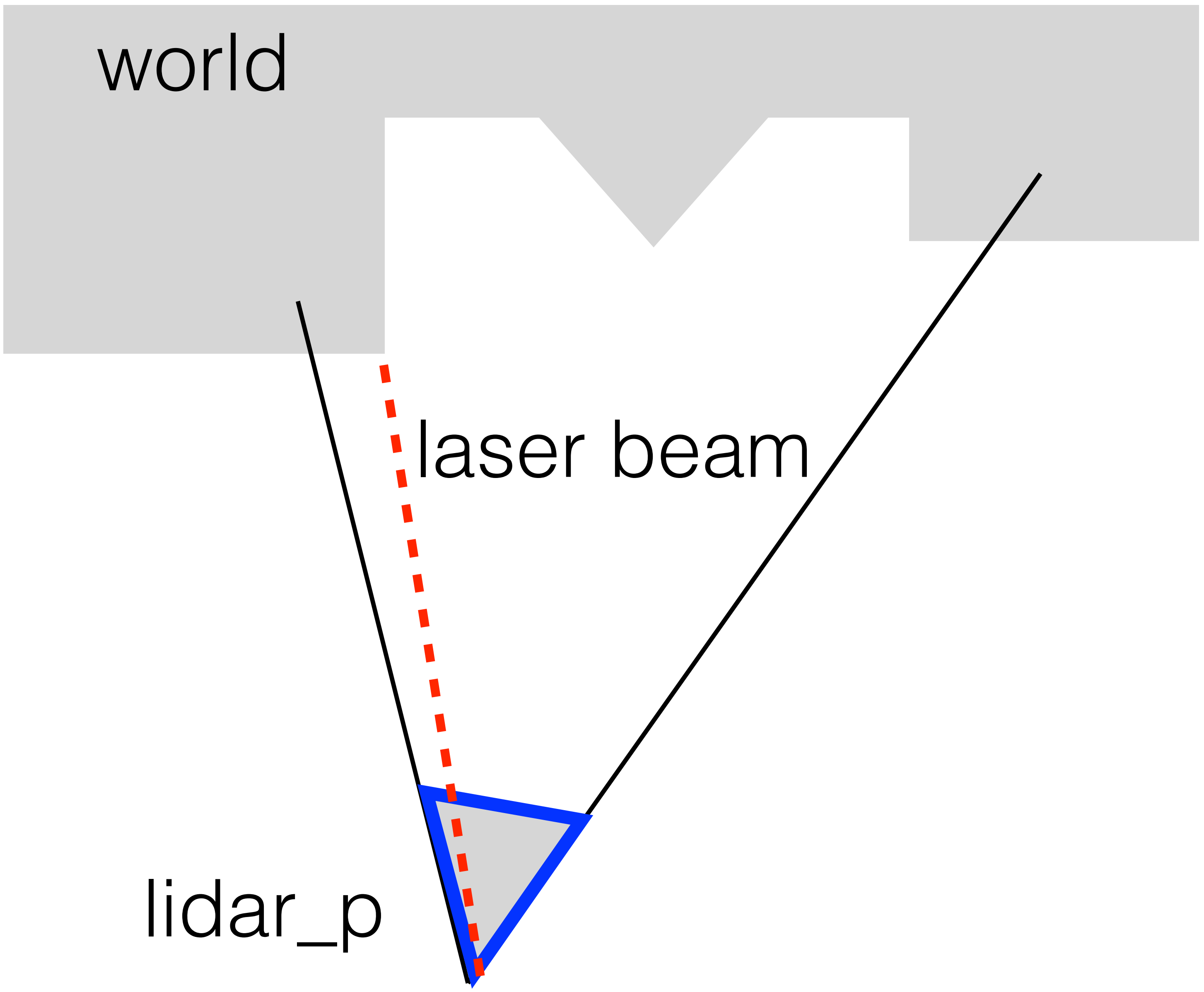
- estimating transformation between **static** coord. frames (sensor calibration)
- principle of lidar, camera, realsense, stereo …

The topic of next lecture:

- estimating transformation between **dynamic** coord. frames (robot/sensor localization - SLAM)

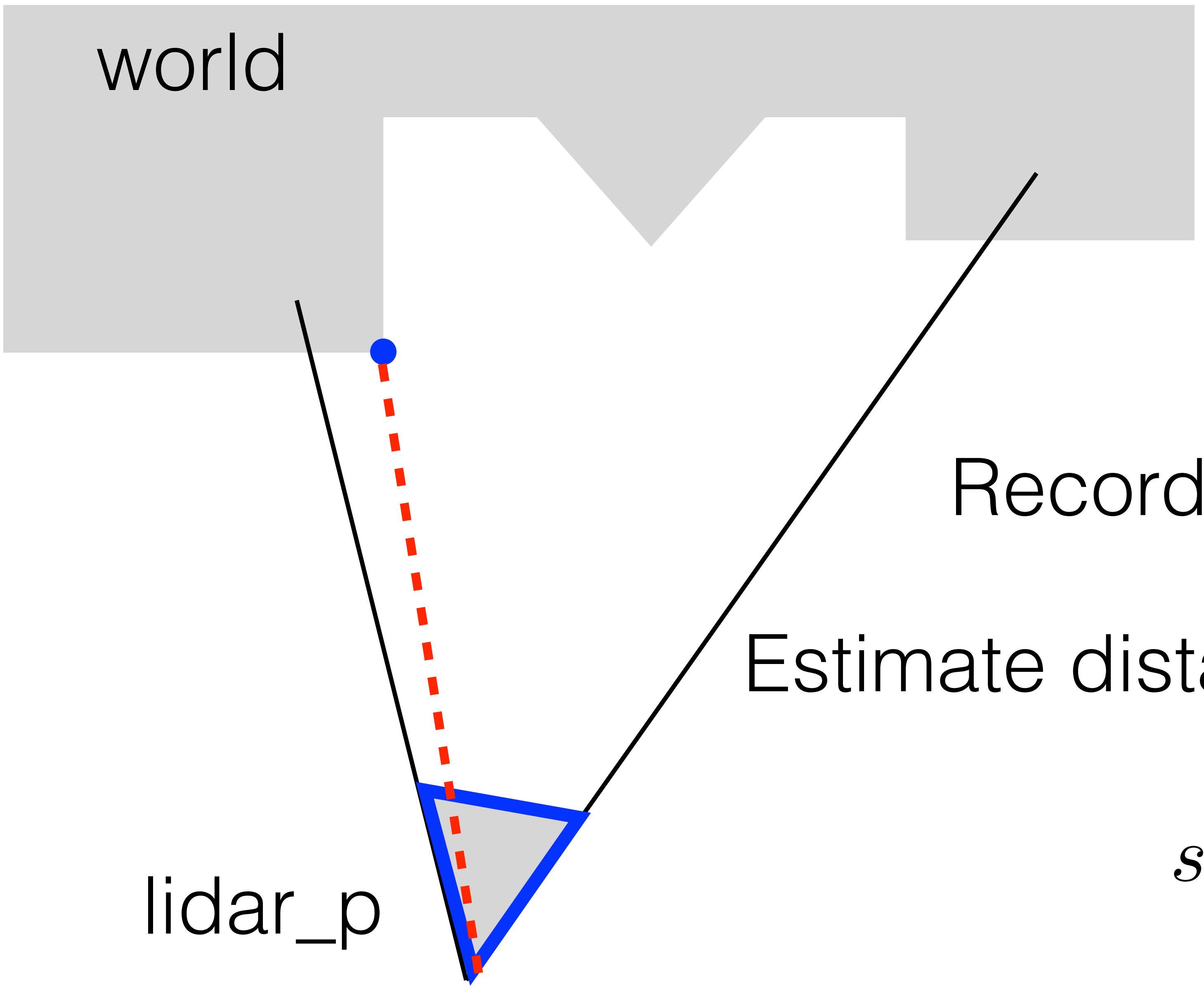
Lidar

Lidar is device which measure depth of some points in its field-of view by time-of-flight principle.



Lidar

Lidar is device which measure depth of some points in its field-of view by time-of-flight principle.



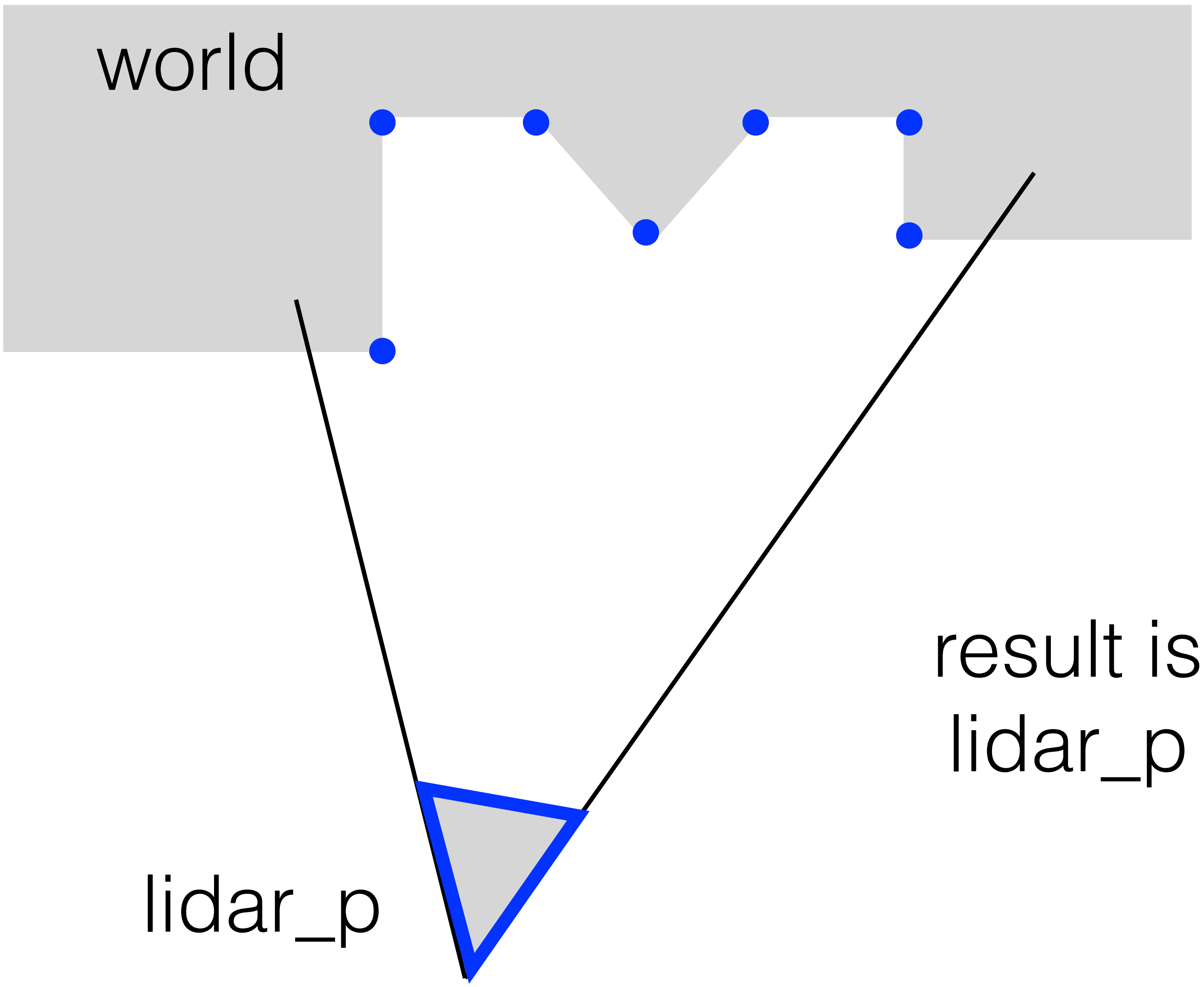
Record reflected beam

Estimate distance from time-of-flight.

$$s = c \cdot \frac{t}{2}$$

Lidar

Lidar is device which measure depth of some points in its field-of view by time-of-flight principle.

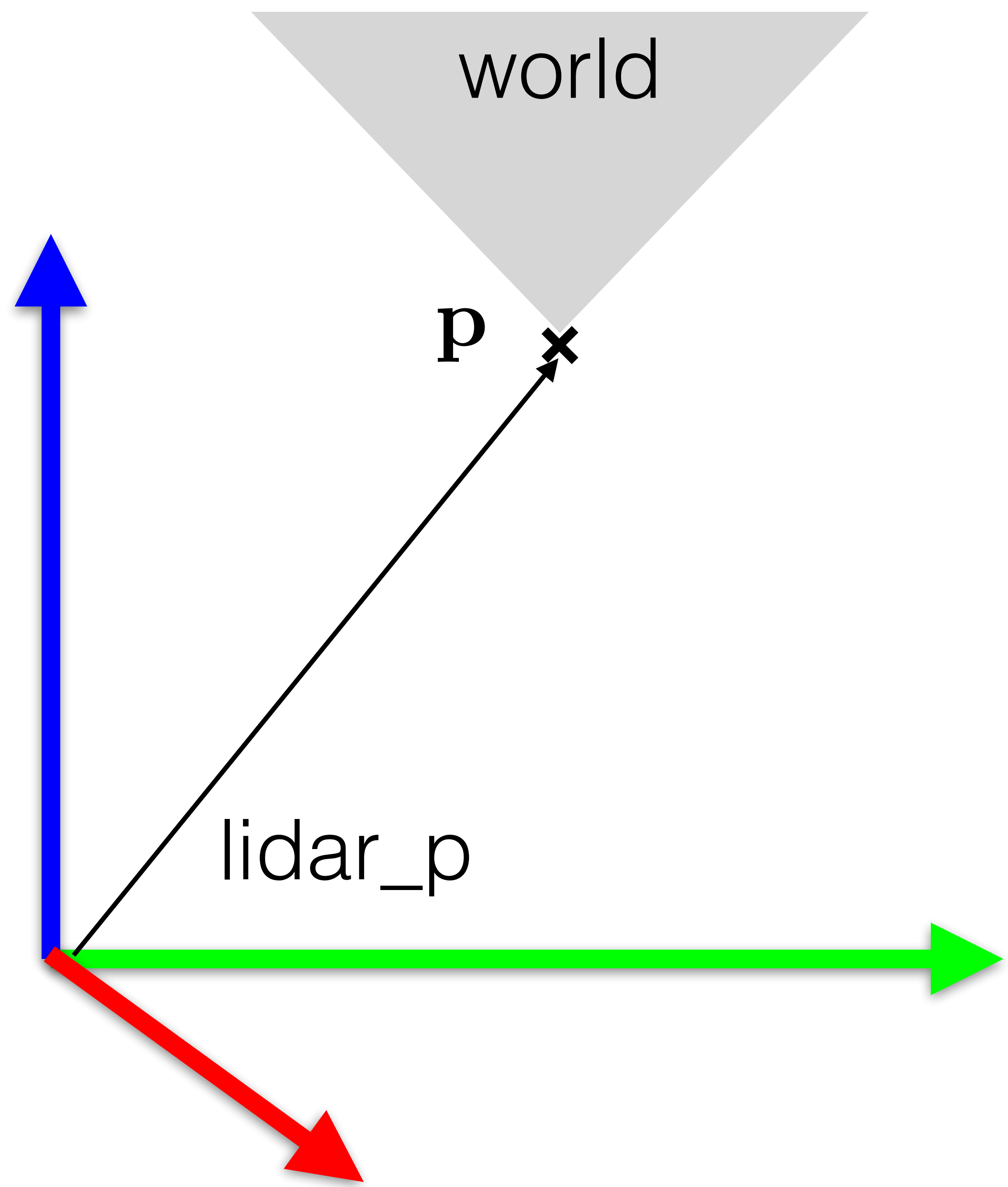


result is 3D point cloud in lidar_p coordinate frame

Euclidean transformation of a rigid body

Let us now work only with two coordinate frames:

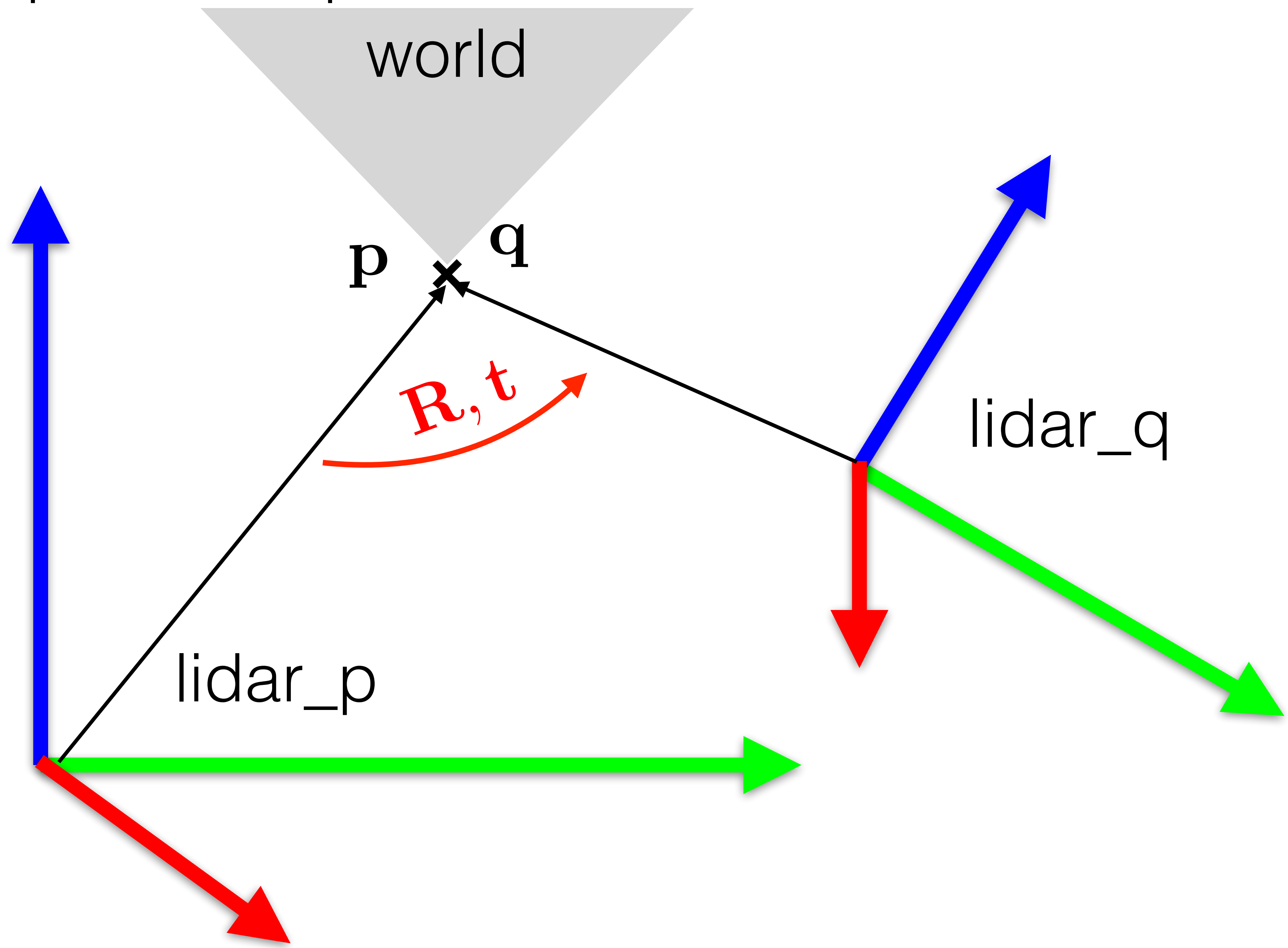
- lidar_p in which points are denoted as $\mathbf{p} \in \mathcal{R}^3$



Euclidean transformation of a rigid body

Let us now work only with two coordinate frames:

- lidar_p in which points are denoted as $\mathbf{p} \in \mathcal{R}^3$
- lidar_q in which points are denoted as $\mathbf{q} \in \mathcal{R}^3$



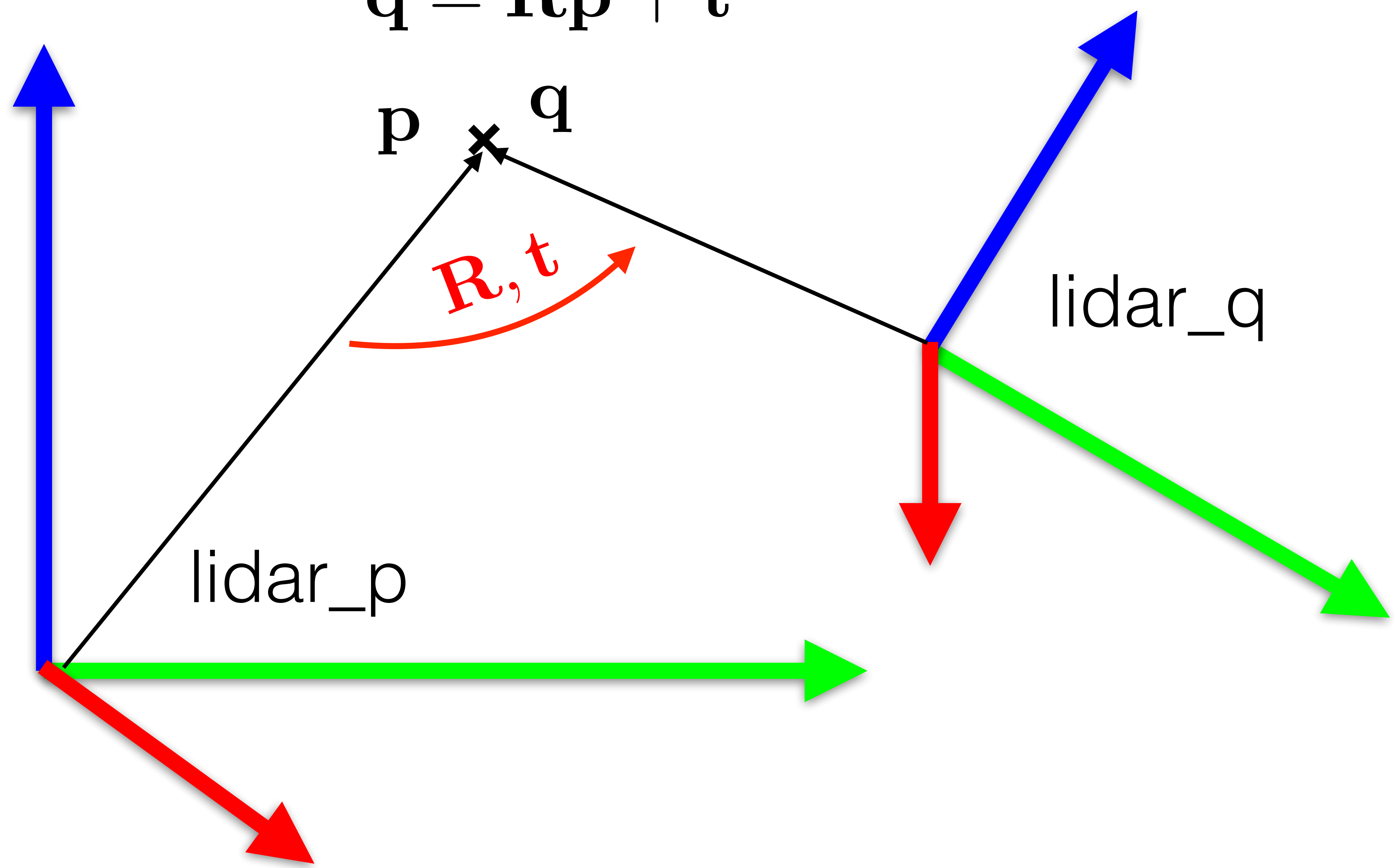
Euclidean transformation of a rigid body

Let us now work only with two coordinate frames:

- lidar_p in which points are denoted as $\mathbf{p} \in \mathcal{R}^3$
- lidar_q in which points are denoted as $\mathbf{q} \in \mathcal{R}^3$

Transformation between measurements uniquely determined:

$$\mathbf{q} = \mathbf{R}\mathbf{p} + \mathbf{t}$$



Euclidean transformation of a rigid body

- Assuming Euclidean motion (no squeezing)

$$\mathbf{q} = \mathbf{R}\mathbf{p} + \mathbf{t}$$

- where $\mathbf{R} \in \mathcal{SO}(3)$ is rotation and $\mathbf{t} \in \mathcal{R}^3$ is translation.

Euclidean transformation of a rigid body

- Assuming Euclidean motion (no squeezing)

$$\mathbf{q} = \mathbf{R}\mathbf{p} + \mathbf{t}$$

- where $\mathbf{R} \in \mathcal{SO}(3)$ is rotation and $\mathbf{t} \in \mathcal{R}^3$ is translation.
- Special orthogonal group

$$\mathcal{SO}(3)_{=}\{\mathbf{R} \in \mathcal{R}^{3\times 3} \mid \mathbf{R}^\top \mathbf{R} = \mathbf{I},\ \det(\mathbf{R}) = +1\}$$

Euclidean transformation of a rigid body

- Assuming Euclidean motion (no squeezing)

$$\mathbf{q} = \mathbf{R}\mathbf{p} + \mathbf{t}$$

- where $\mathbf{R} \in \mathcal{SO}(3)$ is rotation and $\mathbf{t} \in \mathcal{R}^3$ is translation.
- Special orthogonal group

$$\mathcal{SO}(3)_{=}\{\mathbf{R} \in \mathcal{R}^{3\times3} \mid \mathbf{R}^\top\mathbf{R} = \mathbf{I},\ \det(\mathbf{R}) = +1\}$$

- This is affine (not linear) transformation $\Rightarrow$ introduce homogeneous coordinates

$$\overline{\mathbf{p}} = \begin{bmatrix} \mathbf{p} \\ 1 \end{bmatrix}$$

- Euclidean transformation is given by matrix $\mathbf{g} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix}$

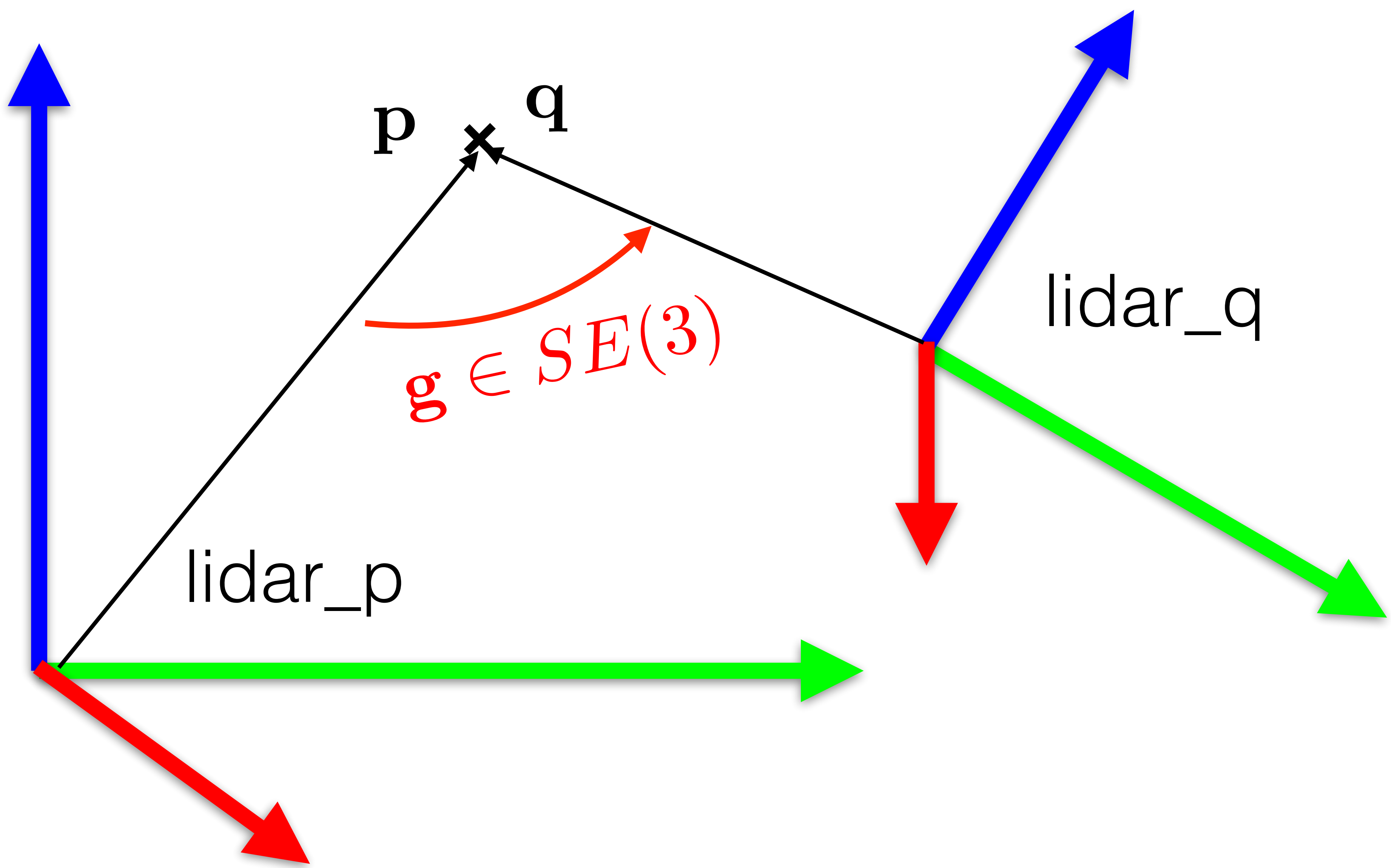
$$\overline{\mathbf{q}} = \mathbf{g}\,\overline{\mathbf{p}}$$

Euclidean transformation of a rigid body

- Set of all transformations forms Special Euclidean group

$$\mathcal{SE}(3) = \{\mathbf{g} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \mid \mathbf{R} \in \mathcal{SO}(3), \mathbf{t} \in \mathcal{R}^3\}$$

where $\mathcal{SO}(3) = \{\mathbf{R} \in \mathcal{R}^{3\times 3} \mid \mathbf{R}^\top \mathbf{R} = \mathbf{I},\ \det(\mathbf{R}) = +1\}$



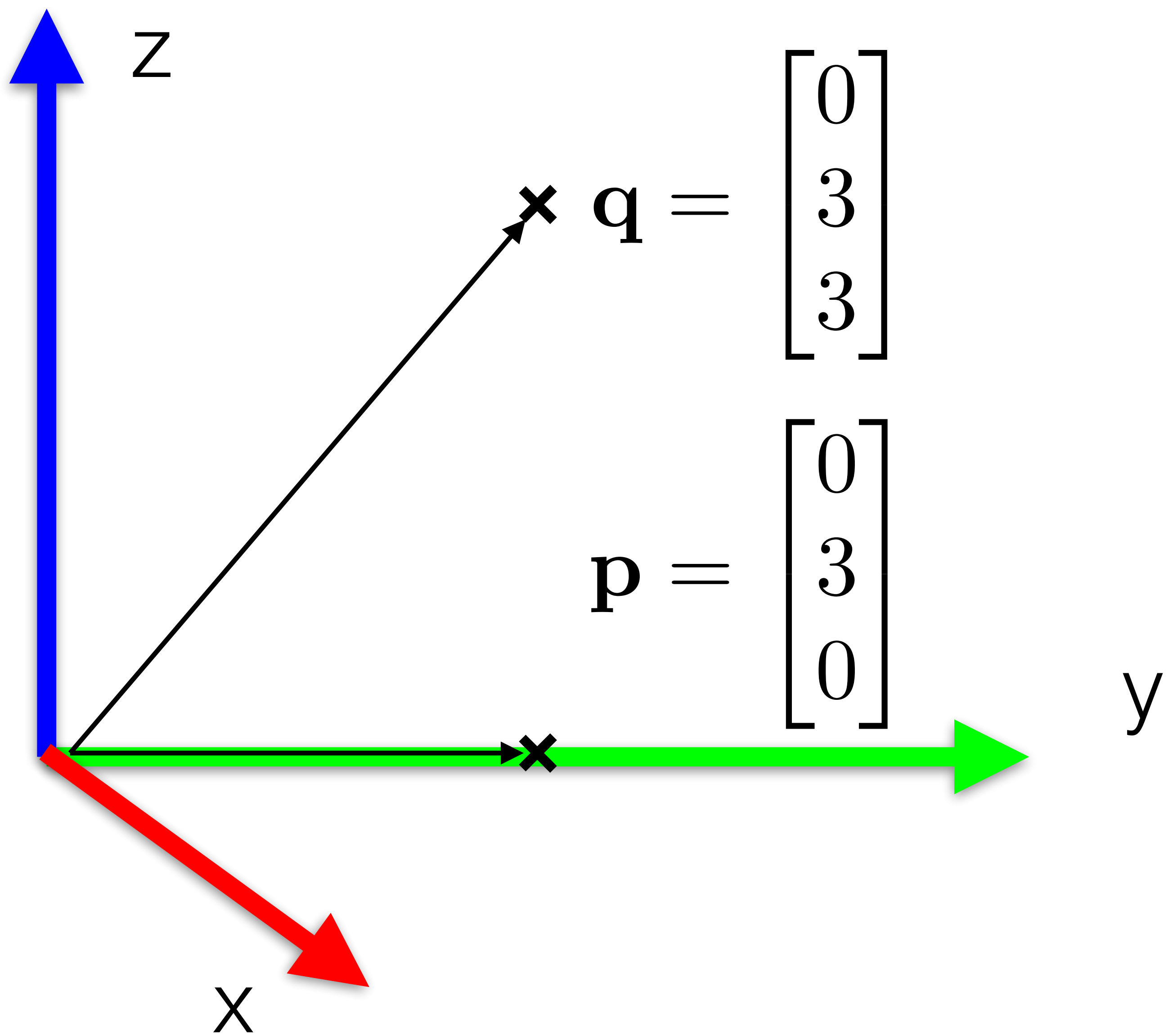
Euclidean transformation of a rigid body

Example 1:

What is rotation around axis y?

What does this rotation preserve?

?



$$\mathbf{q} = \begin{bmatrix} 0 \\ 3 \\ 3 \end{bmatrix}$$

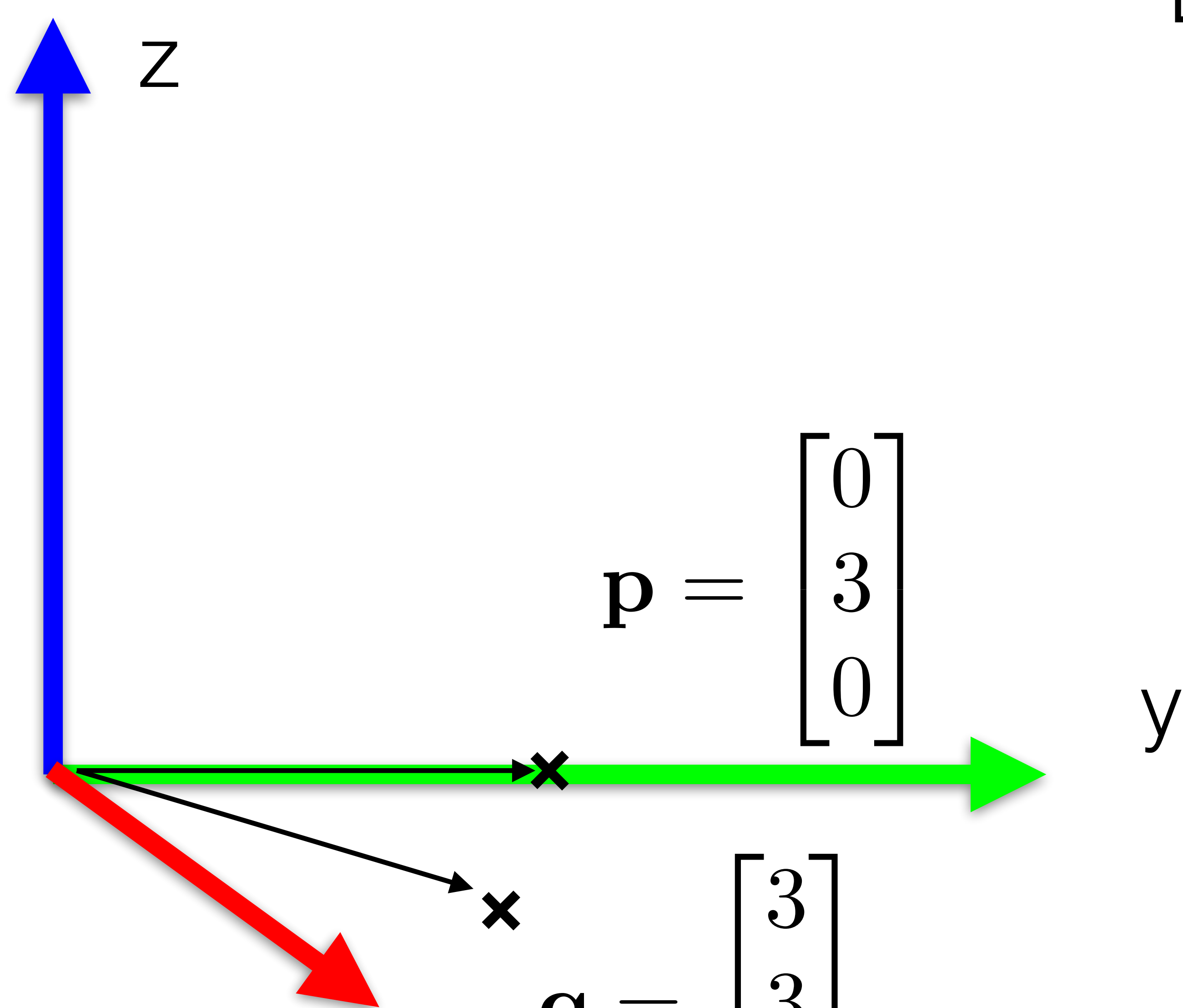
$$\mathbf{p} = \begin{bmatrix} 0 \\ 3 \\ 0 \end{bmatrix}$$

Euclidean transformation of a rigid body

Example 1:

What is rotation around axis y?
What does this rotation preserve?

$$R_y = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$$



$$\mathbf{p} = \begin{bmatrix} 0 \\ 3 \\ 0 \end{bmatrix}$$

$$\mathbf{q} = \begin{bmatrix} 3 \\ 3 \\ 0 \end{bmatrix}$$

Euclidean transformation of a rigid body

Example 2:

Given transformation from lidar1 to lidar2 $\mathbf{g}_{12}$, what is inverse transformation $\mathbf{g}_{21}$

$$\mathbf{g}_{12} = \begin{bmatrix} R_{12} & \mathbf{t}_{12} \\ 000 & 1 \end{bmatrix} \qquad \mathbf{g}_{21} = \quad ?$$

Euclidean transformation of a rigid body

Example 2:

Given transformation from lidar1 to lidar2 $\mathbf{g}_{12}$, what is inverse transformation $\mathbf{g}_{21}$

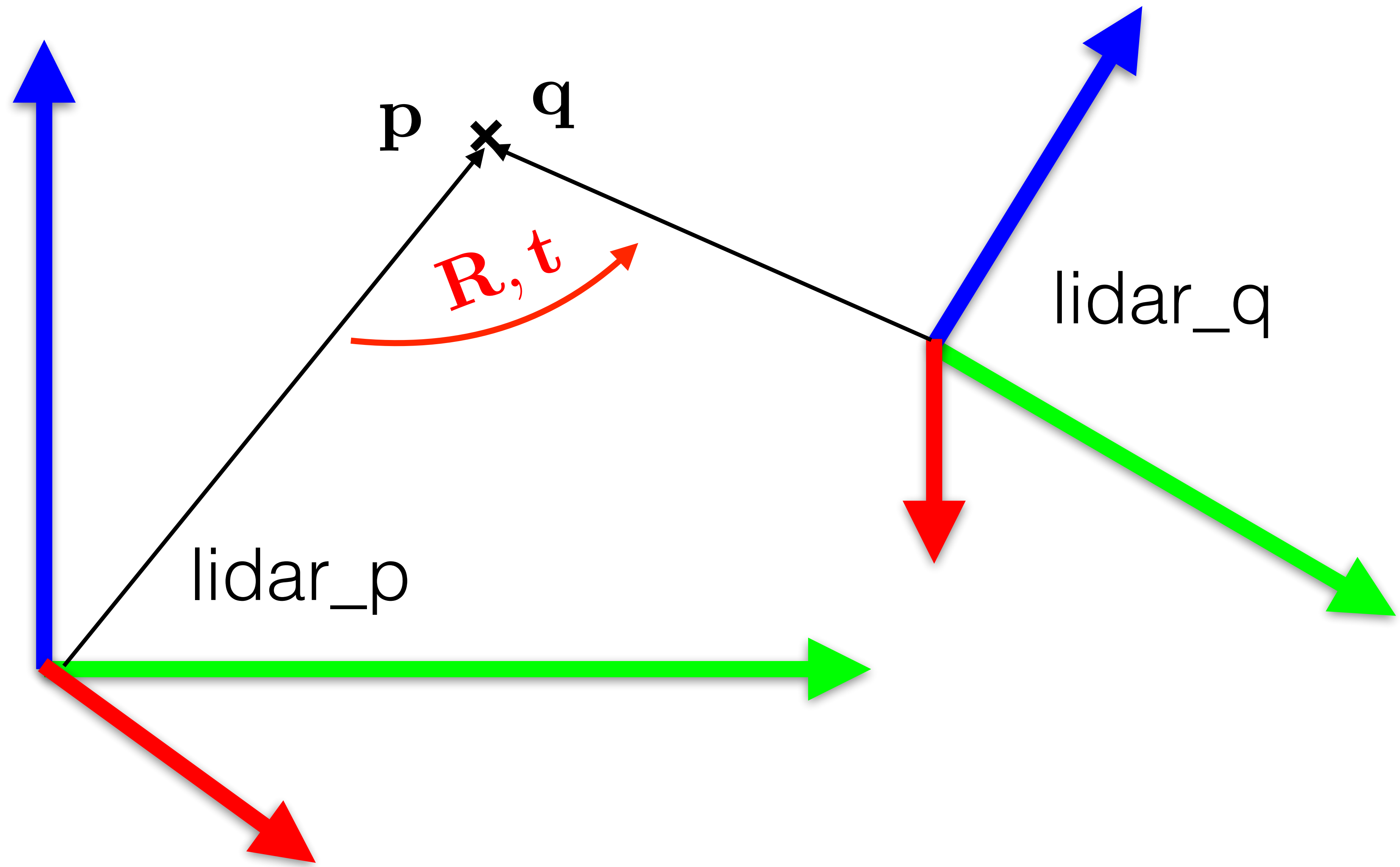
$$\mathbf{g}_{12} = \begin{bmatrix} R_{12} & \mathbf{t}_{12} \\ 000 & 1 \end{bmatrix} \qquad \mathbf{g}_{21} = \begin{bmatrix} R_{21}^{\top} & -R_{21}^{\top}\mathbf{t}_{21} \\ 000 & 1 \end{bmatrix}$$

Summary

$$\mathcal{SE}(3) = \{\mathbf{g} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \ \mid \ \mathbf{R} \in \mathcal{SO}(3), \mathbf{t} \in \mathcal{R}^3\}$$

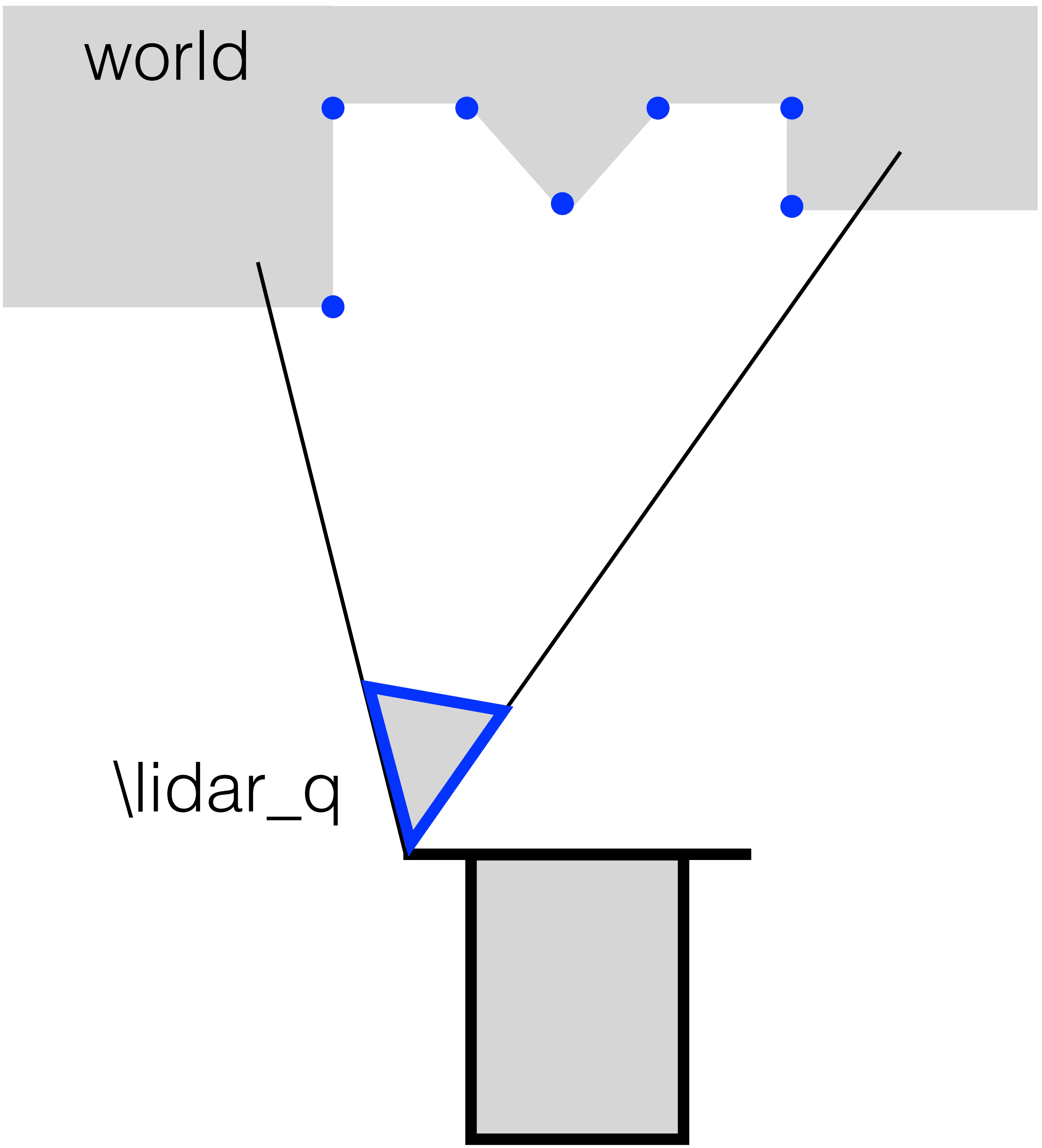
is set of transformations, which model spatio-temporal change of coordinate frames among sensors, actuators and world map.

Mutual calibration of two coordinate frames

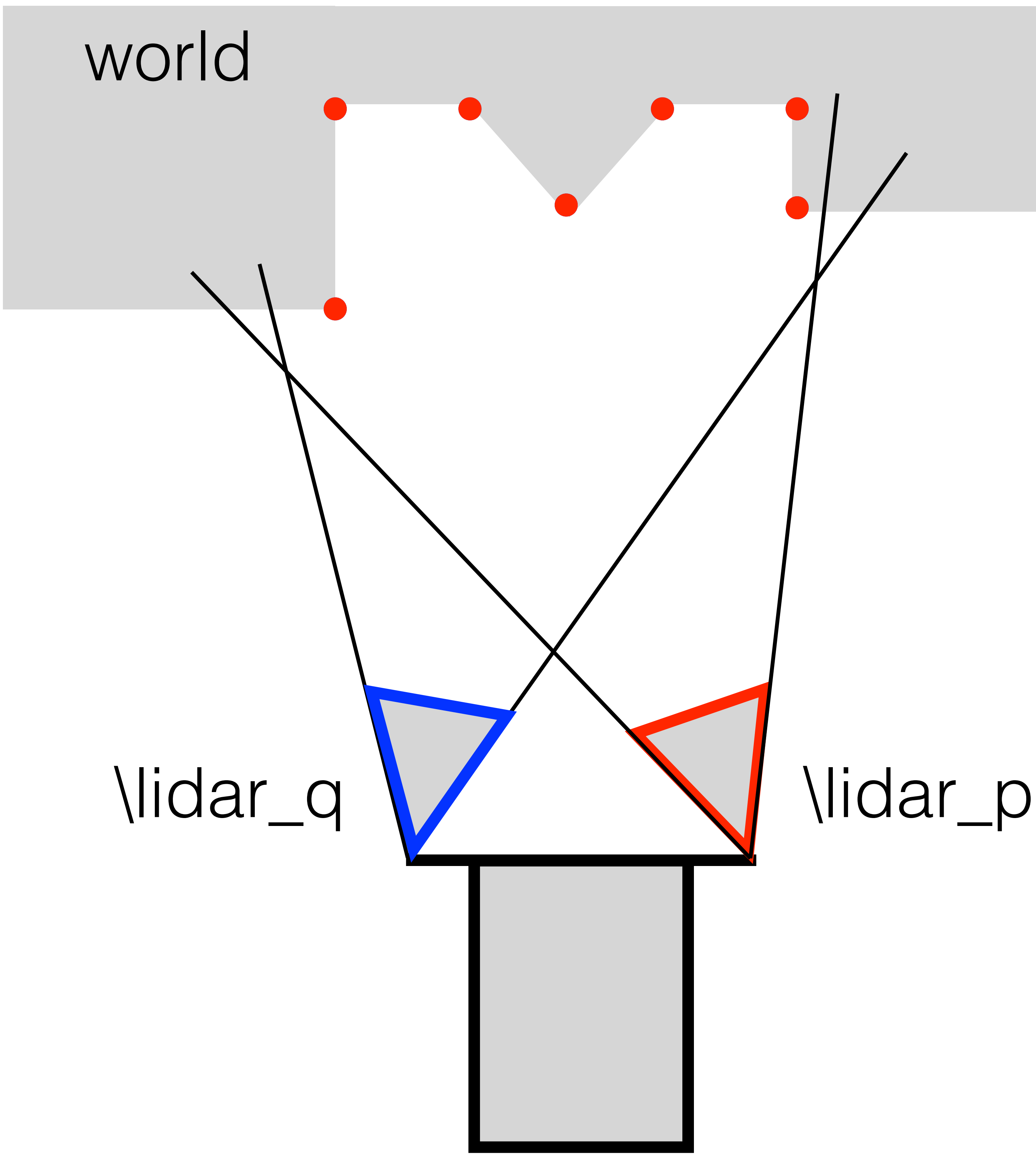


- Let us consider two lidars mounted on a static robot body.
- Each measures pointcloud in its own c.f.
- How can we estimate transformation between them?
- Measuring the mutual transformation by a ruler/protractor is often very inaccurate => can we estimate the R,t accurately?

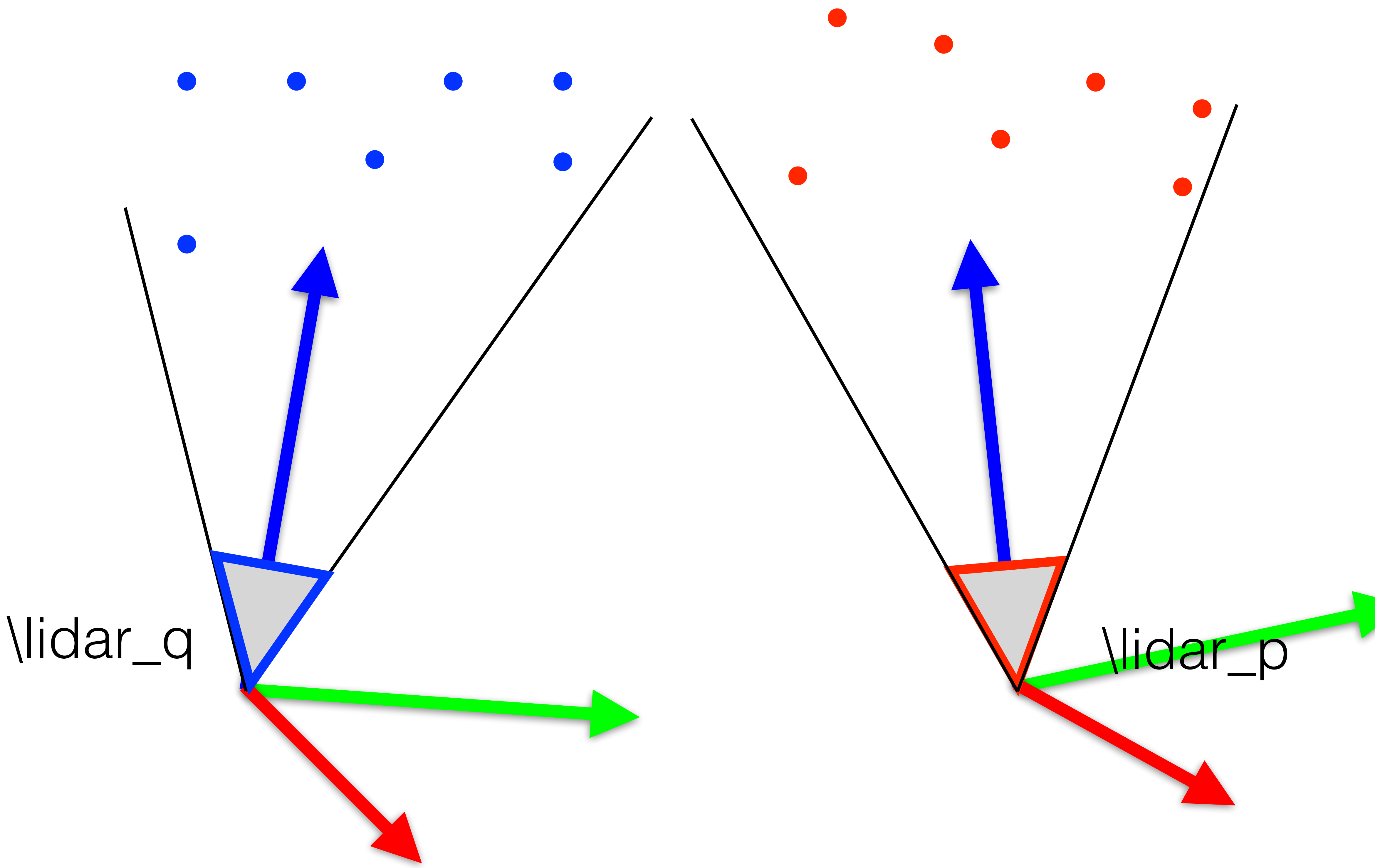
Mutual calibration of two coordinate frames



Mutual calibration of two coordinate frames

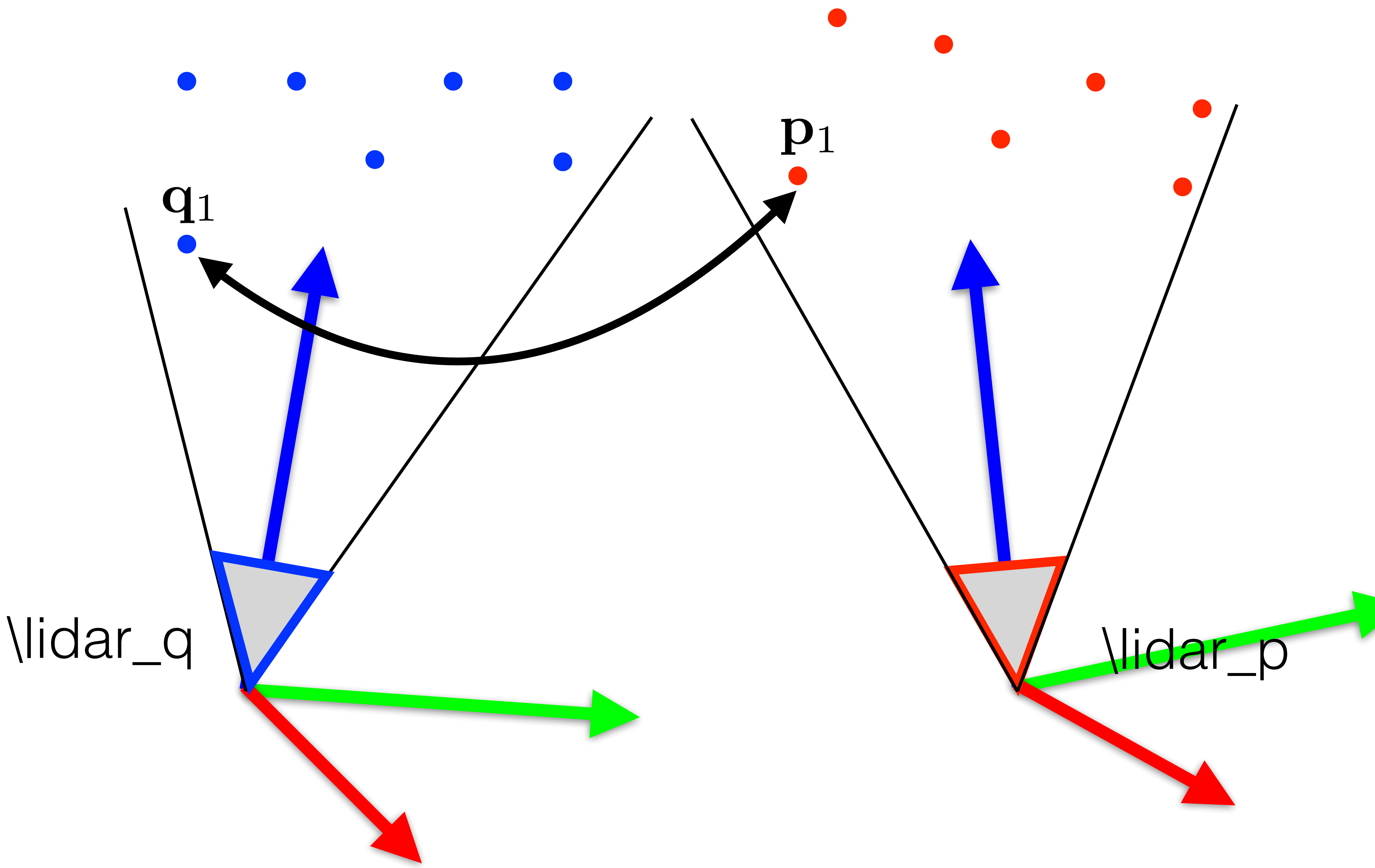


Mutual calibration of two coordinate frames



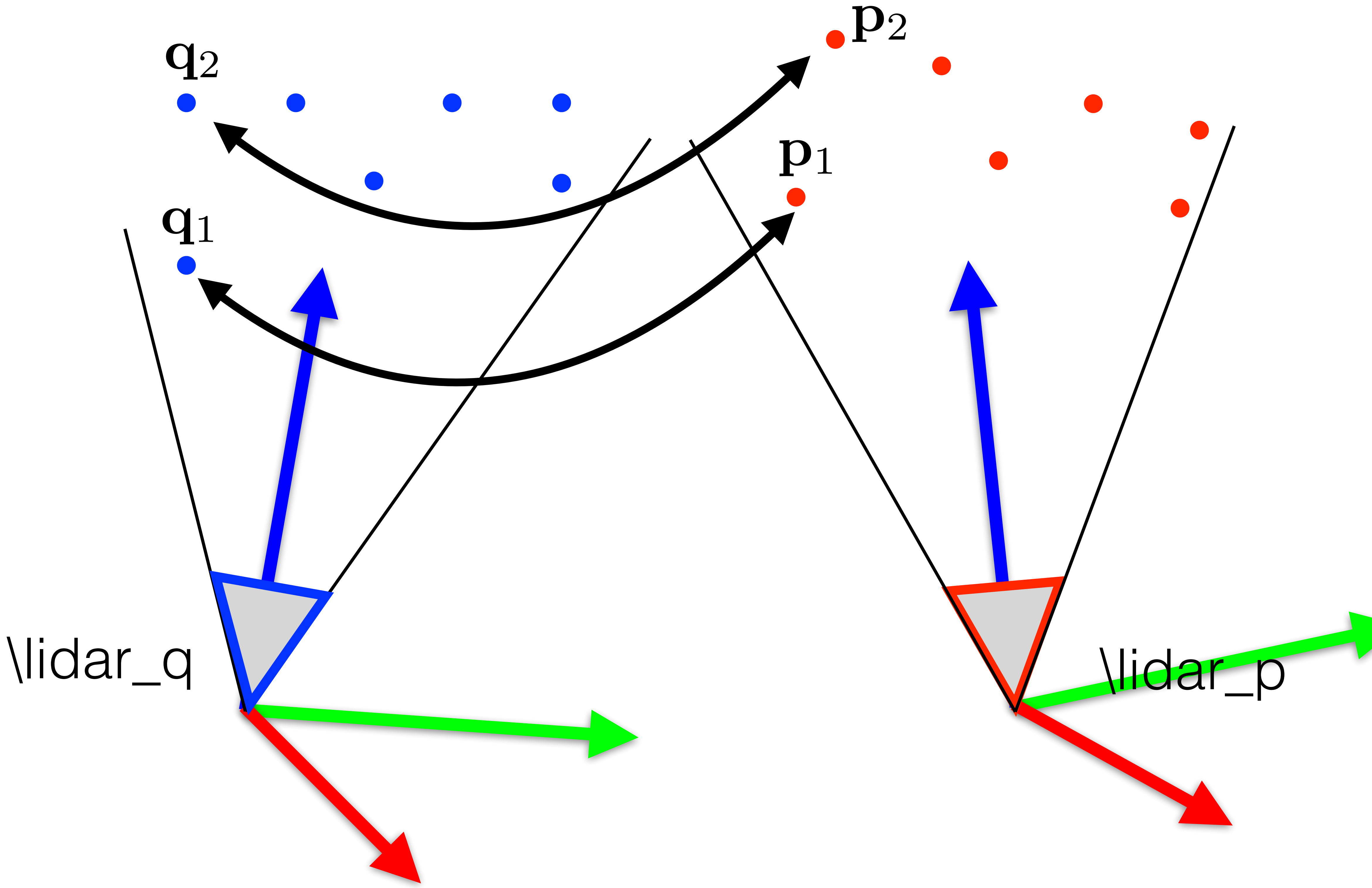
Mutual calibration of two coordinate frames

3D-3D correspondences

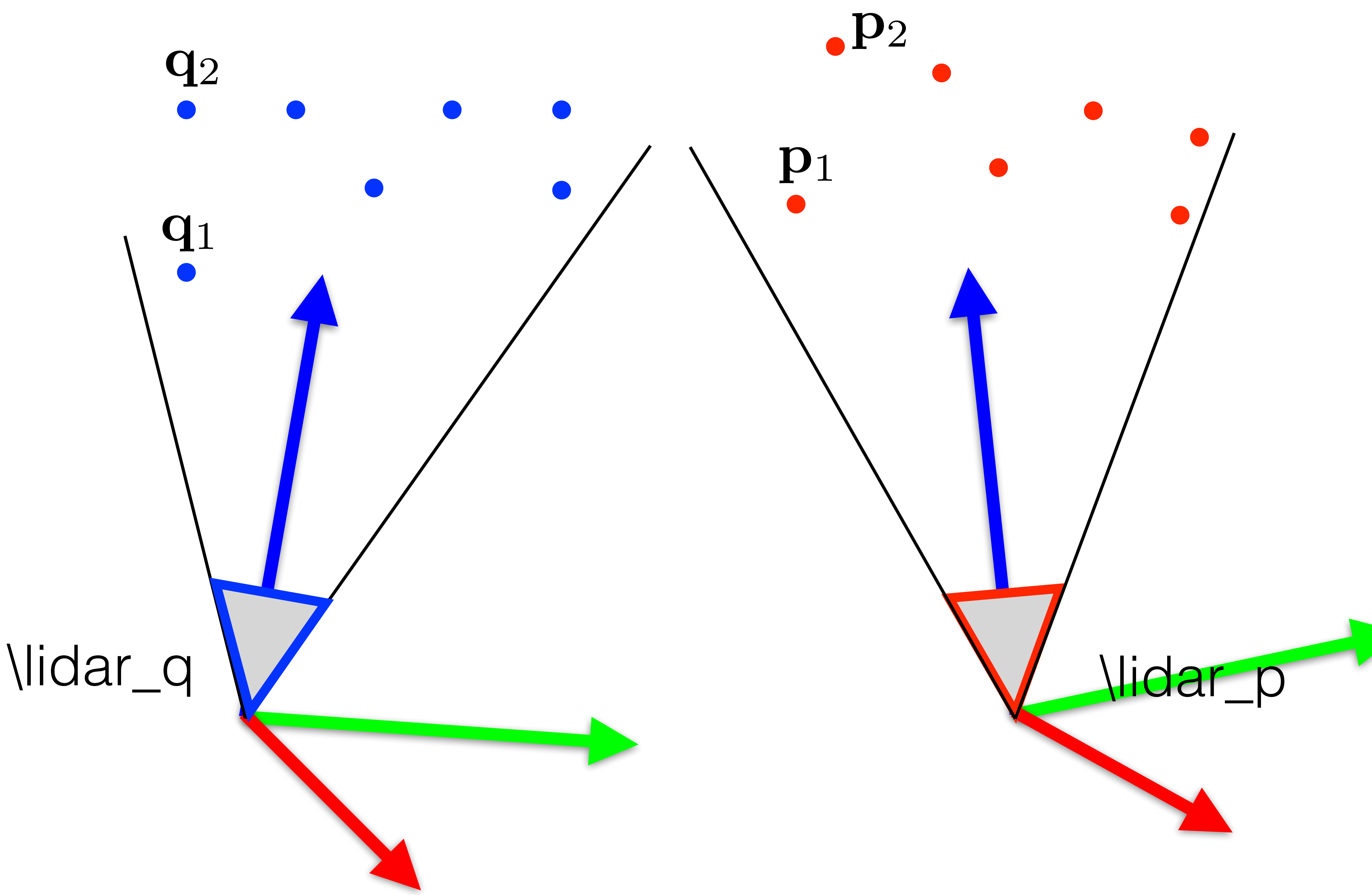


Mutual calibration of two coordinate frames

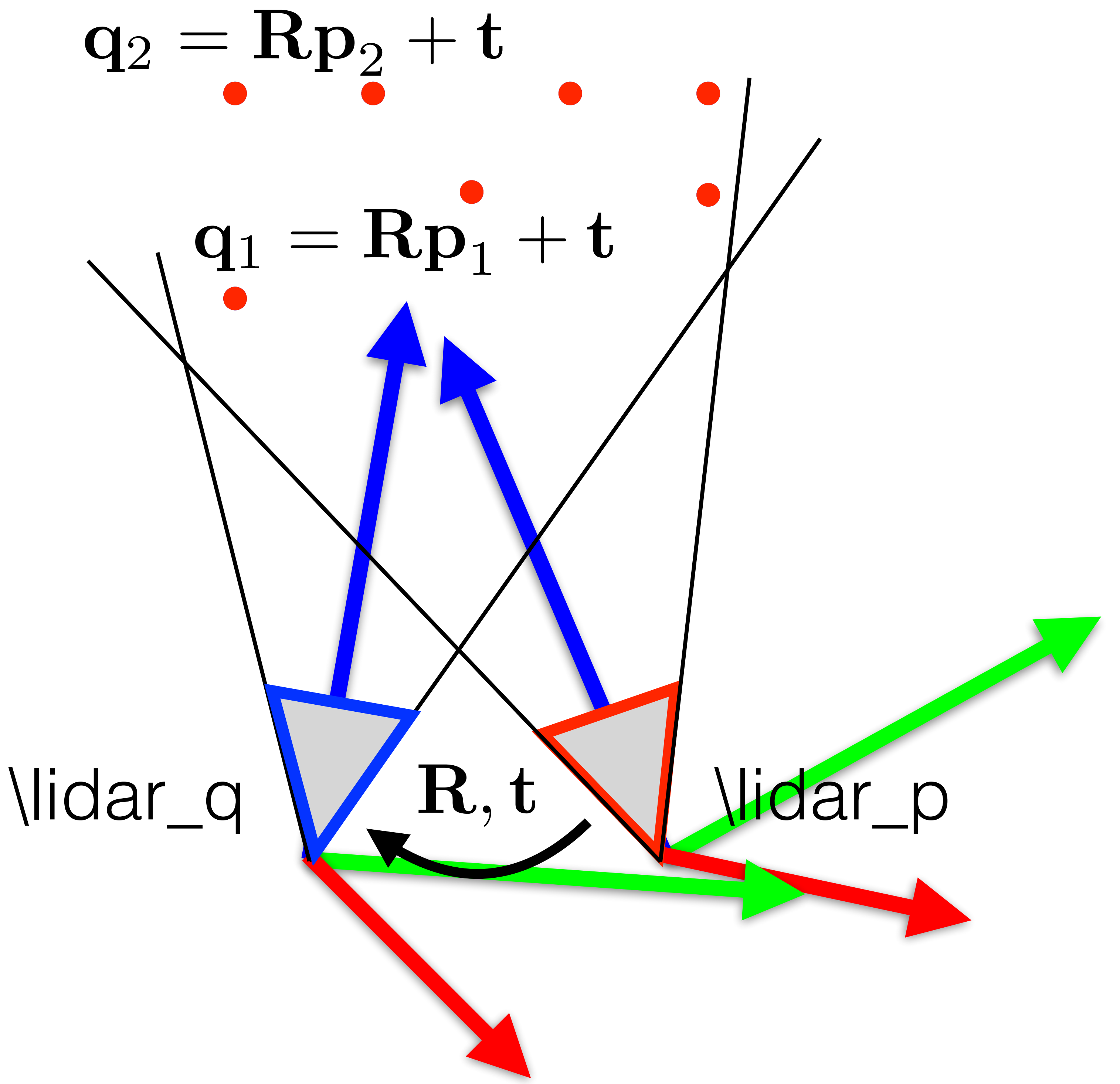
3D-3D correspondences



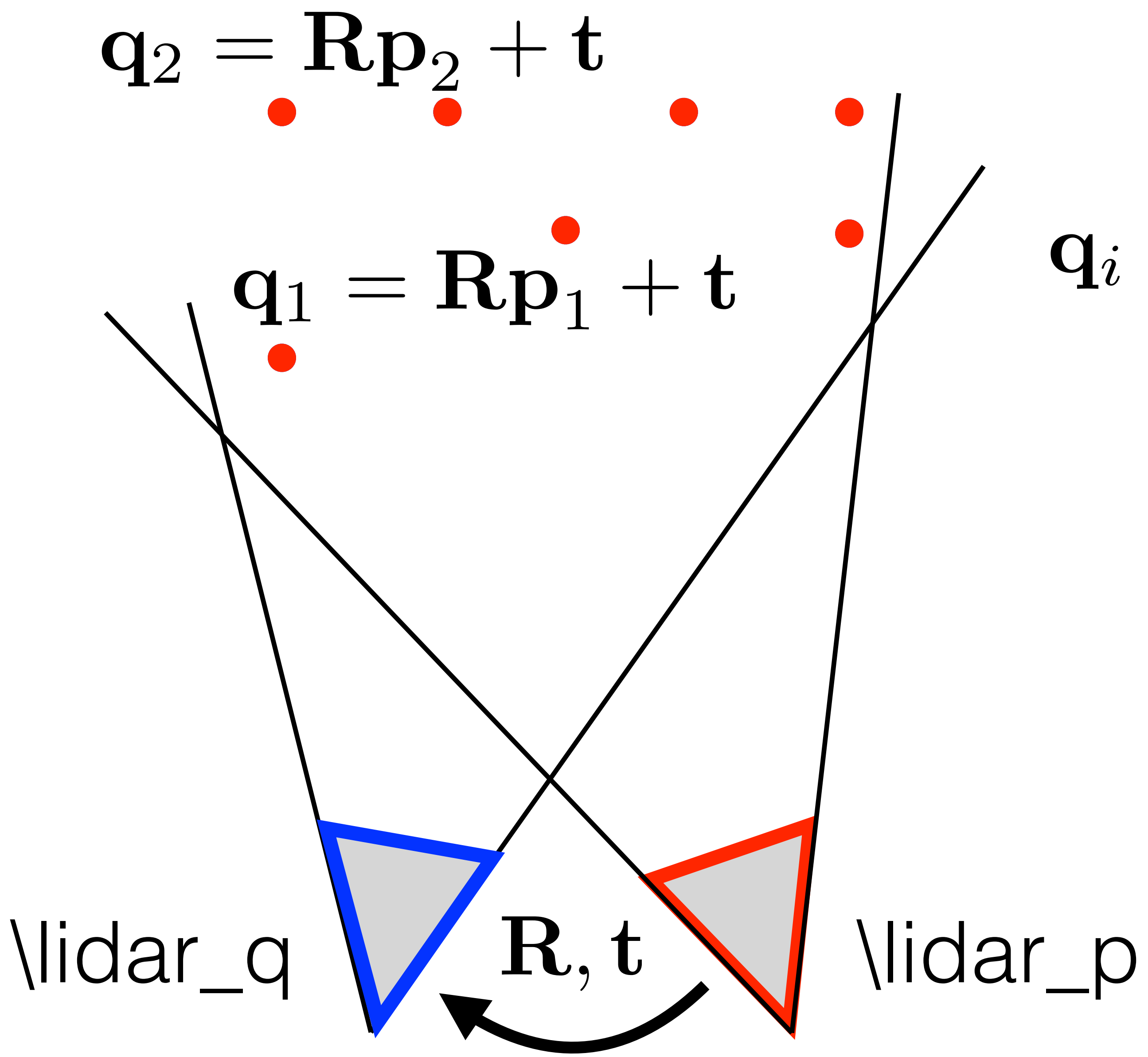
Mutual calibration of two coordinate frames



Mutual calibration of two coordinate frames



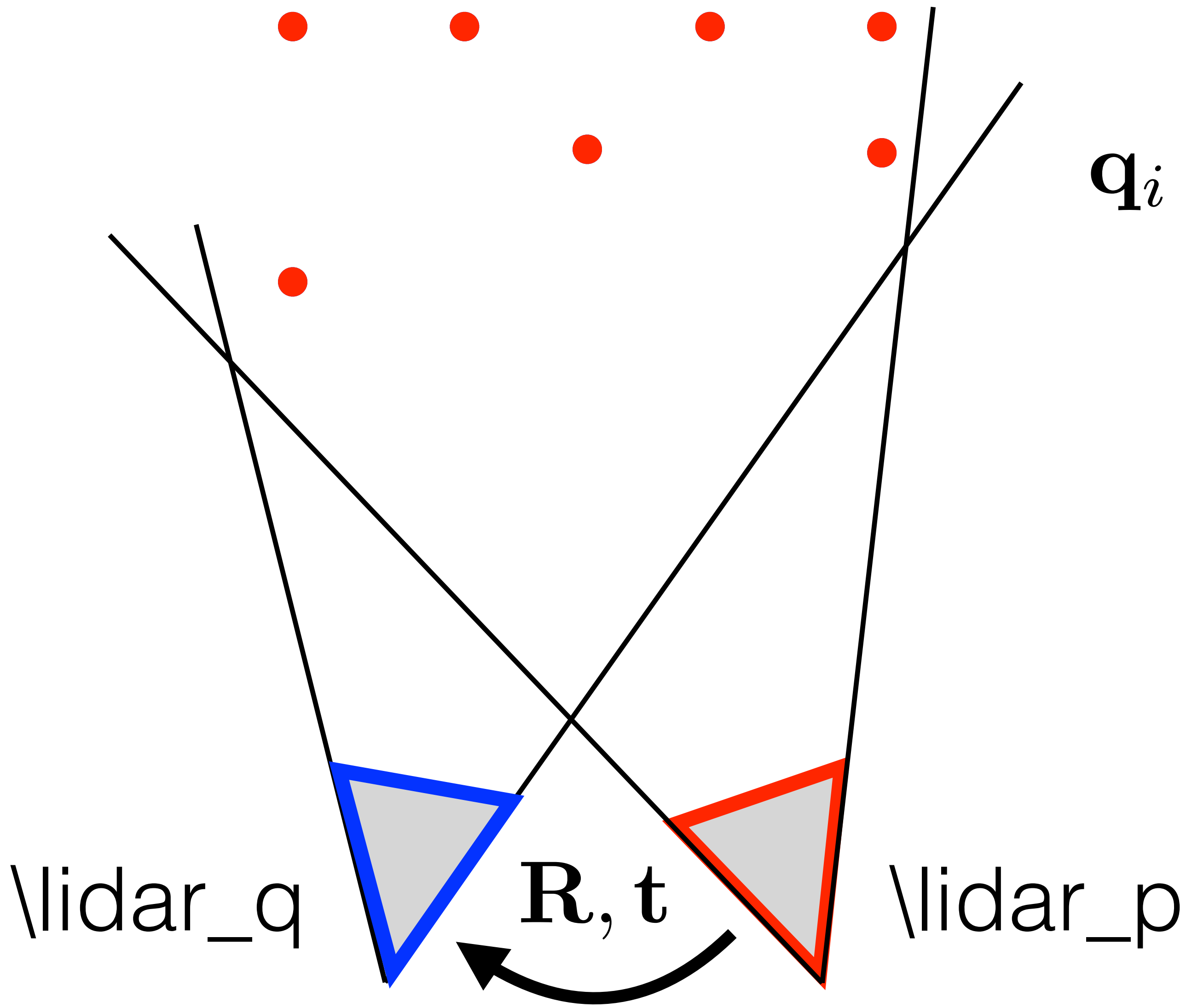
Mutual calibration of two coordinate frames



3N equations:

$$\mathbf{q}_i = \mathbf{R}\mathbf{p}_i + \mathbf{t} \quad \forall_{i=1\ldots N}$$

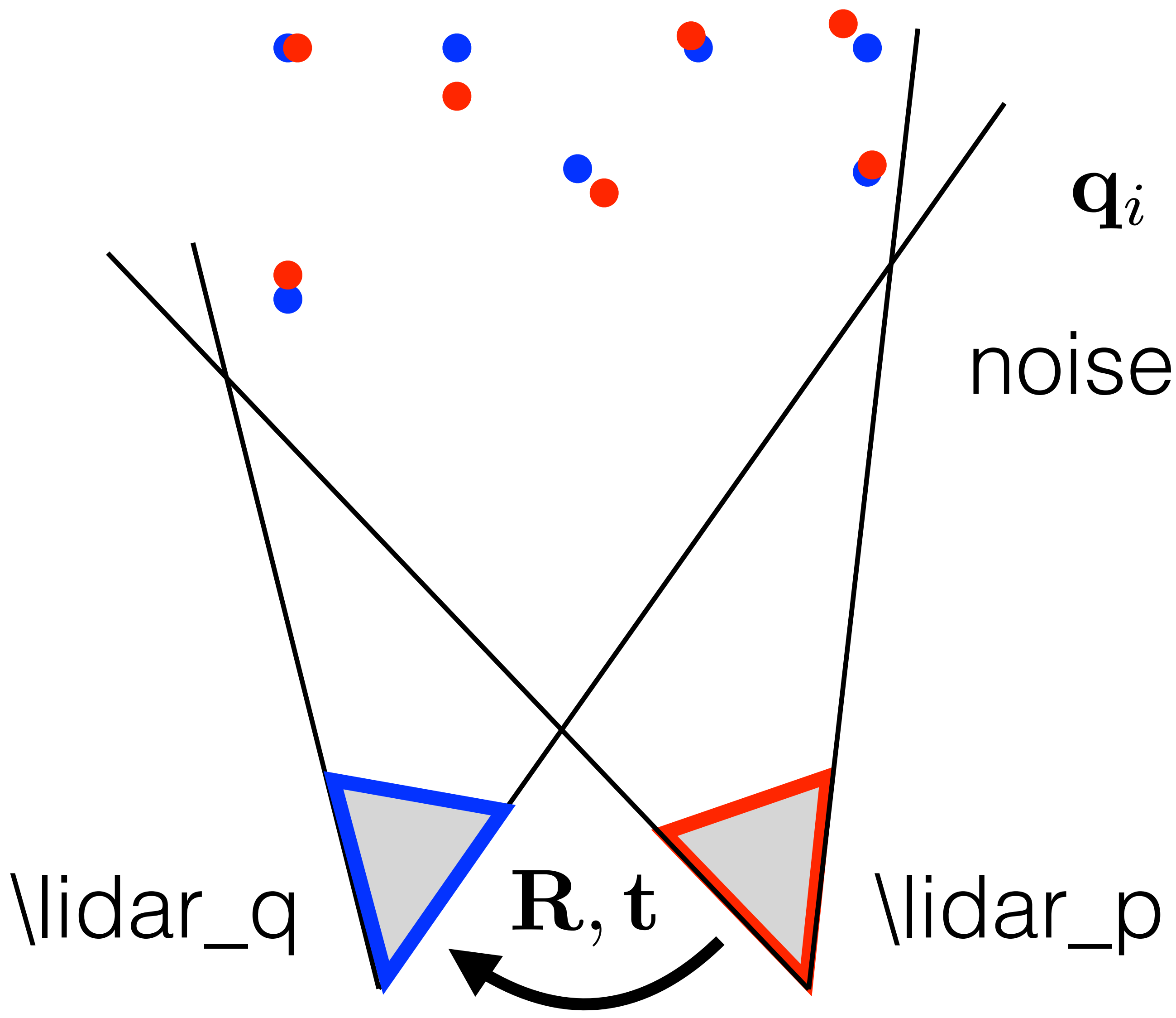
Mutual calibration of two coordinate frames



3N equations:

$$\mathbf{q}_i = \mathbf{R}\mathbf{p}_i + \mathbf{t} \quad \forall_{i=1 \ldots N}$$

Mutual calibration of two coordinate frames

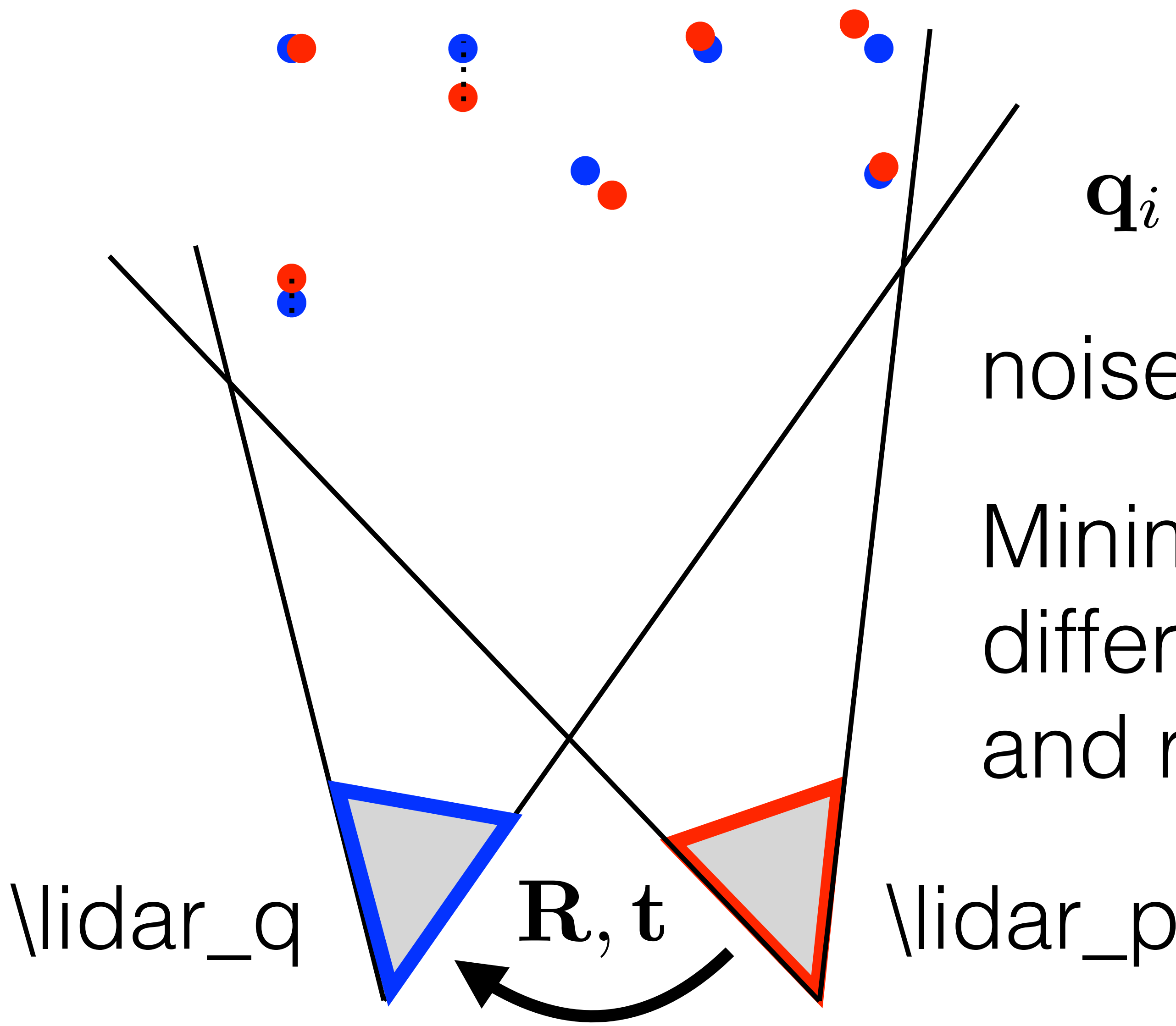


3N equations:

$$\mathbf{q}_i \neq \mathbf{R}\mathbf{p}_i + \mathbf{t} \quad \forall_{i=1 \ldots N}$$

noise => no exact solution

Mutual calibration of two coordinate frames



3N equations:

$$\mathbf{q}_i \neq \mathbf{R}\mathbf{p}_i + \mathbf{t} \quad \forall_{i=1\ldots N}$$

noise => no exact solution

Minimize sum of squared differences between left and right-hand side.

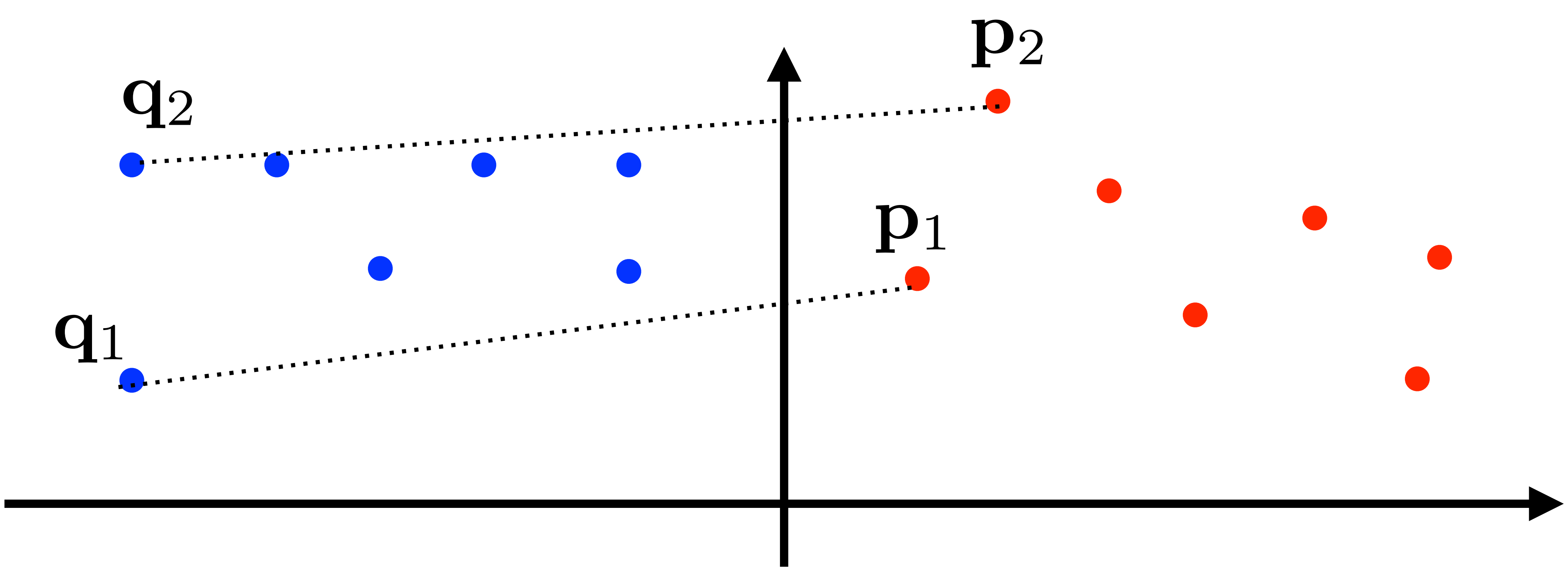
ML estimate wrt:
- **gaussian noise**
- **i.i.d measurements**

$$\mathbf{R}^*, \mathbf{t}^* = \underset{\mathbf{R}\in SO(3),\mathbf{t}\in\mathcal{R}^3}{\arg\min} \sum_i \|\mathbf{R}\mathbf{p}_i + \mathbf{t} - \mathbf{q}_i\|_2^2$$

Mutual calibration of two coordinate frames

$$\mathbf{R}^*, \mathbf{t}^* = \underset{\mathbf{R} \in SO(3), \mathbf{t} \in \mathcal{R}^3}{\arg\min} \sum_i \|\mathbf{R}\mathbf{p}_i + \mathbf{t} - \mathbf{q}_i\|_2^2$$

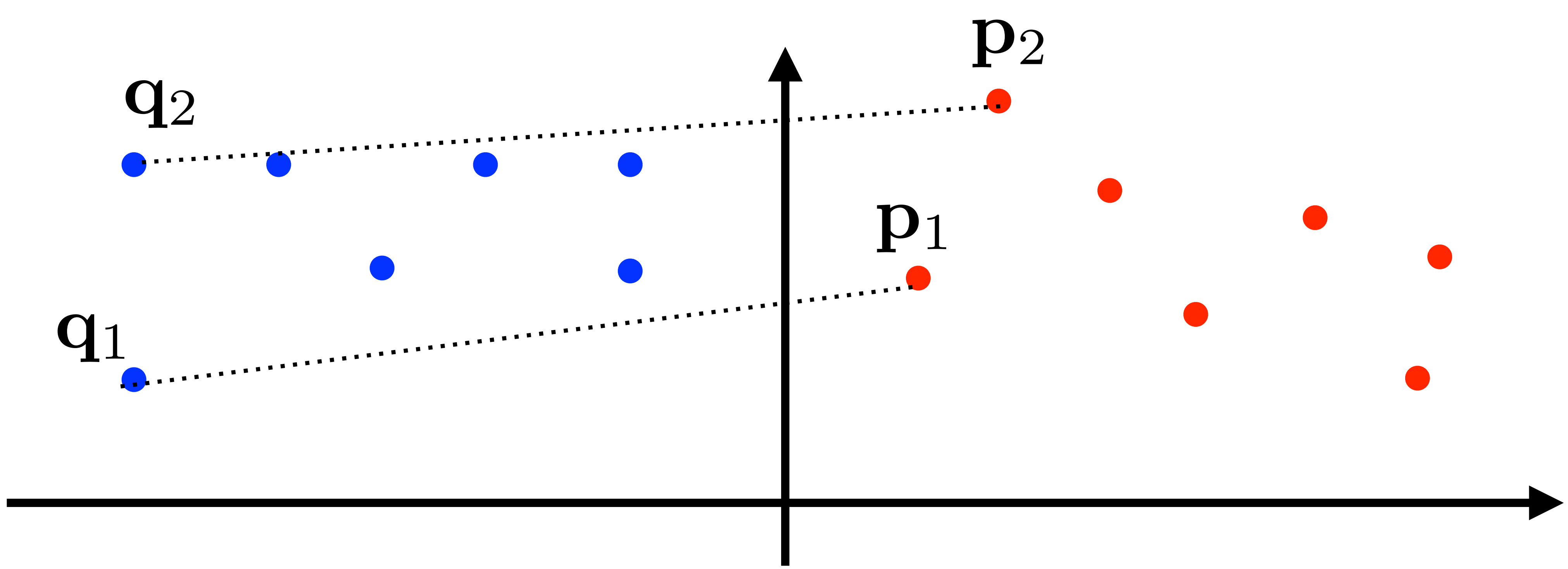
Solution:



Mutual calibration of two coordinate frames

$$\mathbf{R}^*, \mathbf{t}^* = \underset{\mathbf{R}\in SO(3), \mathbf{t}\in\mathcal{R}^3}{\arg\min} \sum_i \|\mathbf{R}\mathbf{p}_i + \mathbf{t} - \mathbf{q}_i\|_2^2$$

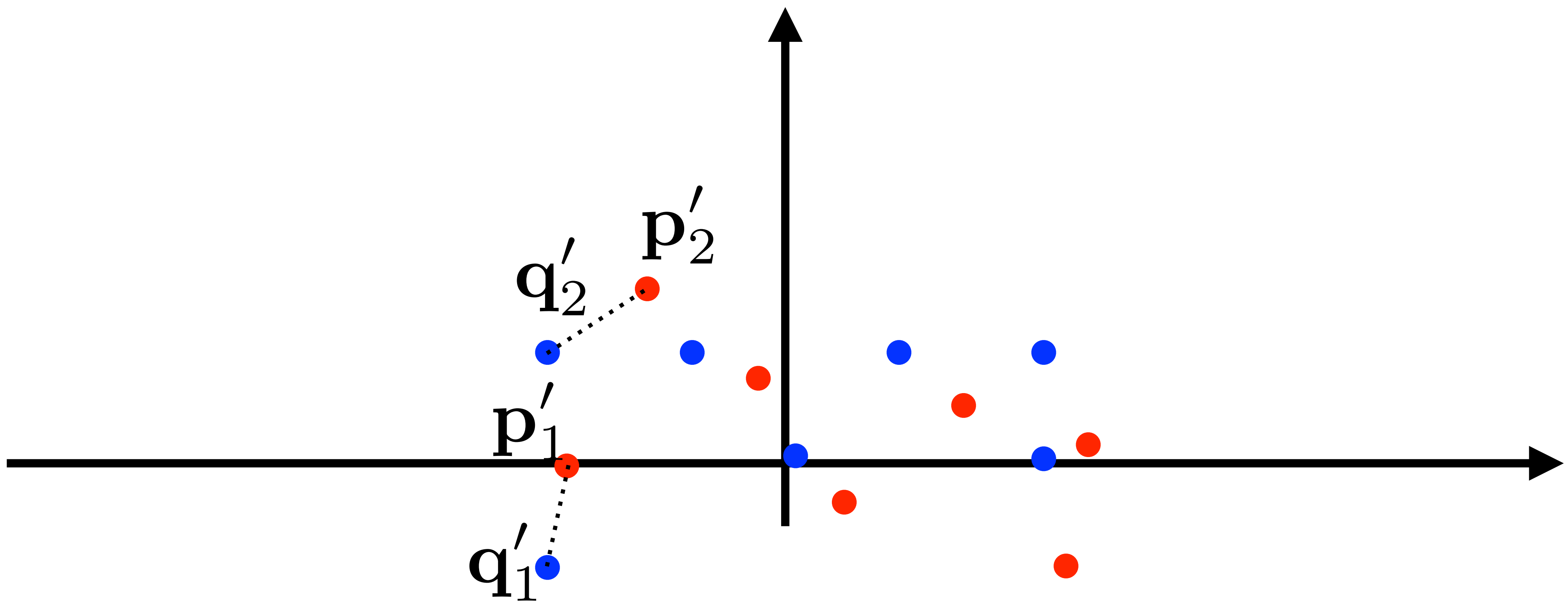
Solution: $$\mathbf{p}'_i = \mathbf{p}_i - \underbrace{\frac{1}{N}\sum_i \mathbf{p}_i}_{\widetilde{\mathbf{p}}}, \quad \mathbf{q}'_i = \mathbf{q}_i - \underbrace{\frac{1}{N}\sum_i \mathbf{q}_i}_{\widetilde{\mathbf{q}}}$$



Mutual calibration of two coordinate frames

$$\mathbf{R}^*, \mathbf{t}^* = \arg\min_{\mathbf{R} \in SO(3), \mathbf{t} \in \mathcal{R}^3} \sum_i \|\mathbf{R}\mathbf{p}_i + \mathbf{t} - \mathbf{q}_i\|_2^2$$

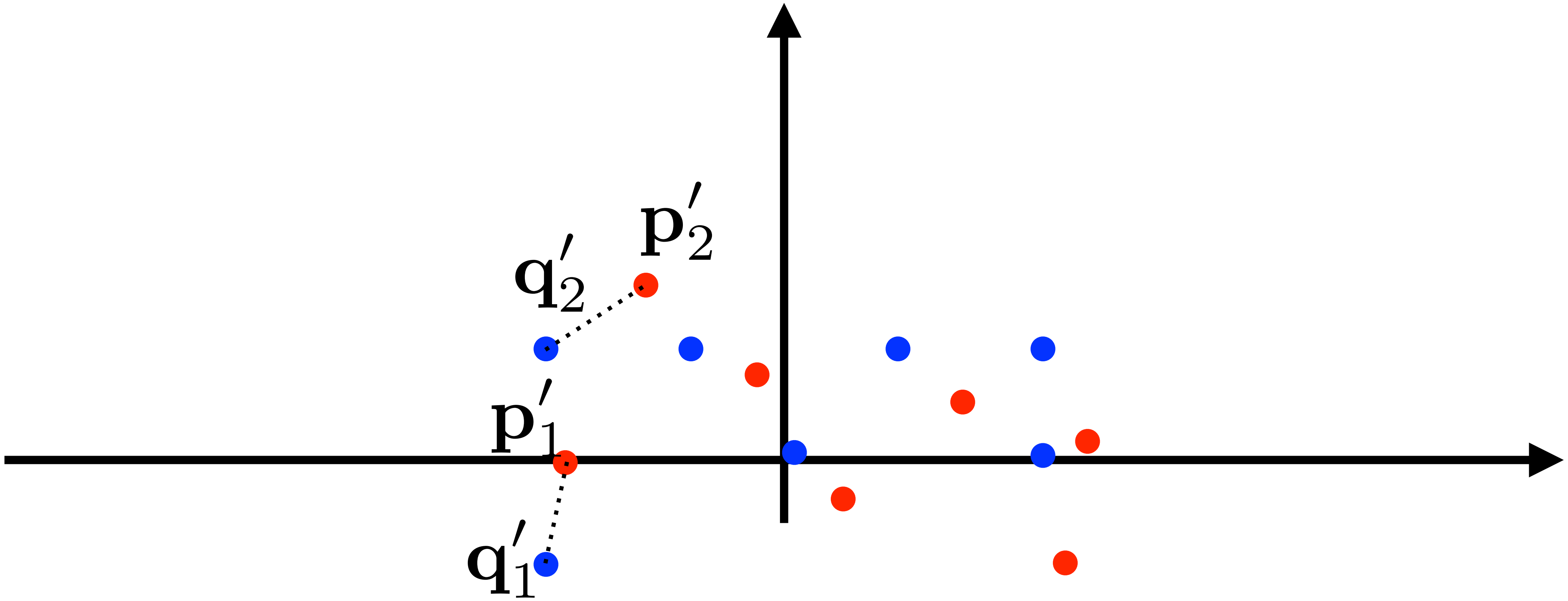
Solution: $\mathbf{p}'_i = \mathbf{p}_i - \underbrace{\frac{1}{N}\sum_i \mathbf{p}_i}_{\widetilde{\mathbf{p}}}, \quad \mathbf{q}'_i = \mathbf{q}_i - \underbrace{\frac{1}{N}\sum_i \mathbf{q}_i}_{\widetilde{\mathbf{q}}}$



Mutual calibration of two coordinate frames

$$\mathbf{R}^*, \mathbf{t}^* = \arg\min_{\mathbf{R}\in SO(3), \mathbf{t}\in\mathcal{R}^3} \sum_i \|\mathbf{R}\mathbf{p}_i + \mathbf{t} - \mathbf{q}_i\|_2^2$$

Solution: estimate covariante matrix: $\mathbf{H} = \sum_i \mathbf{p}_i' \mathbf{q}_i'^\top$

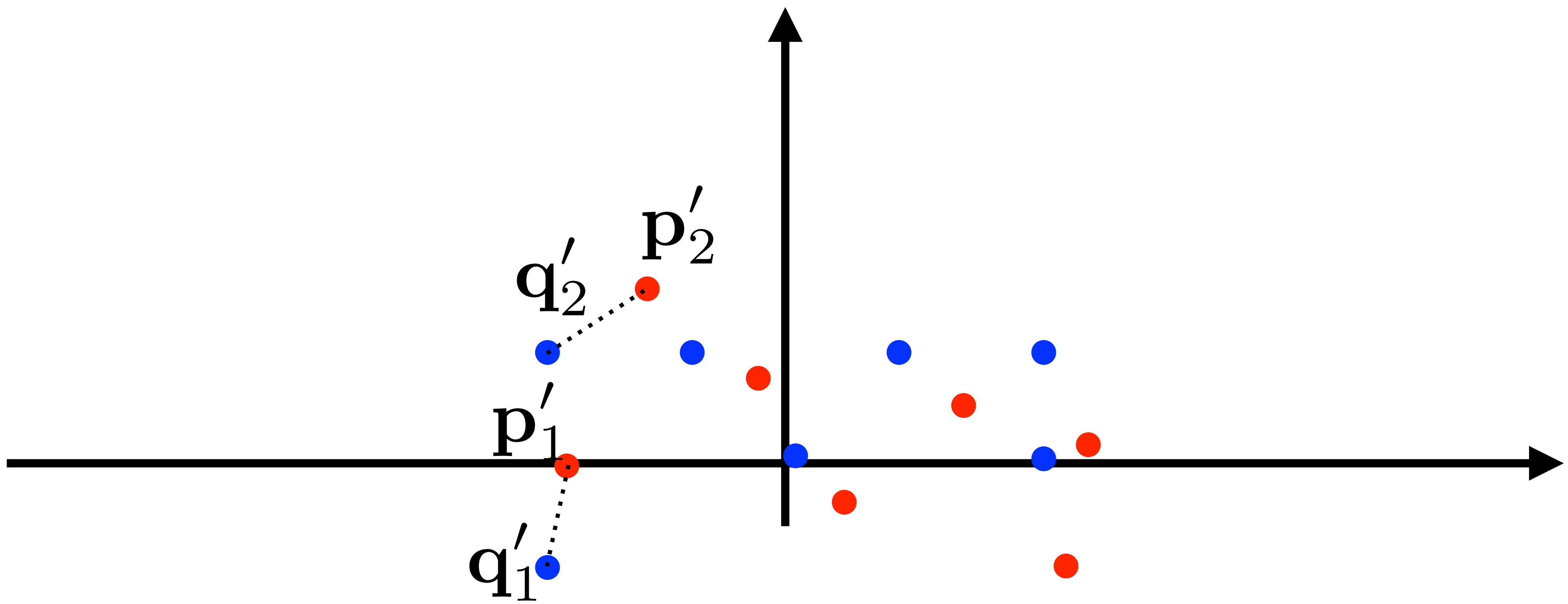


Mutual calibration of two coordinate frames

$$\mathbf{R}^*, \mathbf{t}^* = \underset{\mathbf{R}\in SO(3), \mathbf{t}\in\mathcal{R}^3}{\arg\min} \sum_i \|\mathbf{R}\mathbf{p}_i + \mathbf{t} - \mathbf{q}_i\|_2^2$$

Solution: estimate covariante matrix: $\mathbf{H} = \sum_i \mathbf{p}_i' \mathbf{q}_i'^\top$

find SVD decomposition: $\mathbf{H} = \mathbf{U}\mathbf{S}\mathbf{V}^\top$



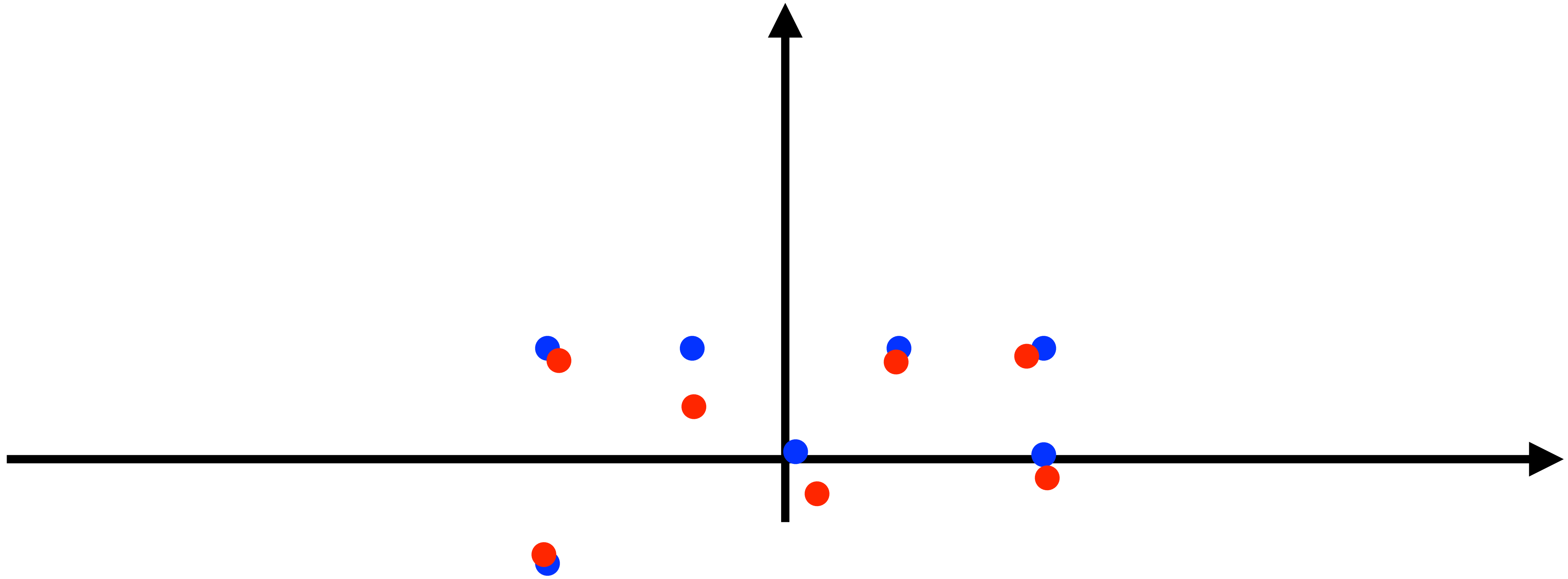
Mutual calibration of two coordinate frames

$$\mathbf{R}^*, \mathbf{t}^* = \underset{\mathbf{R}\in SO(3), \mathbf{t}\in\mathcal{R}^3}{\arg\min} \sum_i \|\mathbf{R}\mathbf{p}_i + \mathbf{t} - \mathbf{q}_i\|_2^2$$

Solution: estimate covariante matrix: $\mathbf{H} = \sum_i \mathbf{p}_i' \mathbf{q}_i'^\top$

find SVD decomposition: $\mathbf{H} = \mathbf{USV}^\top$

estimate optimal rotation: $\mathbf{R}^* = \mathbf{VU}^\top$



Mutual calibration of two coordinate frames

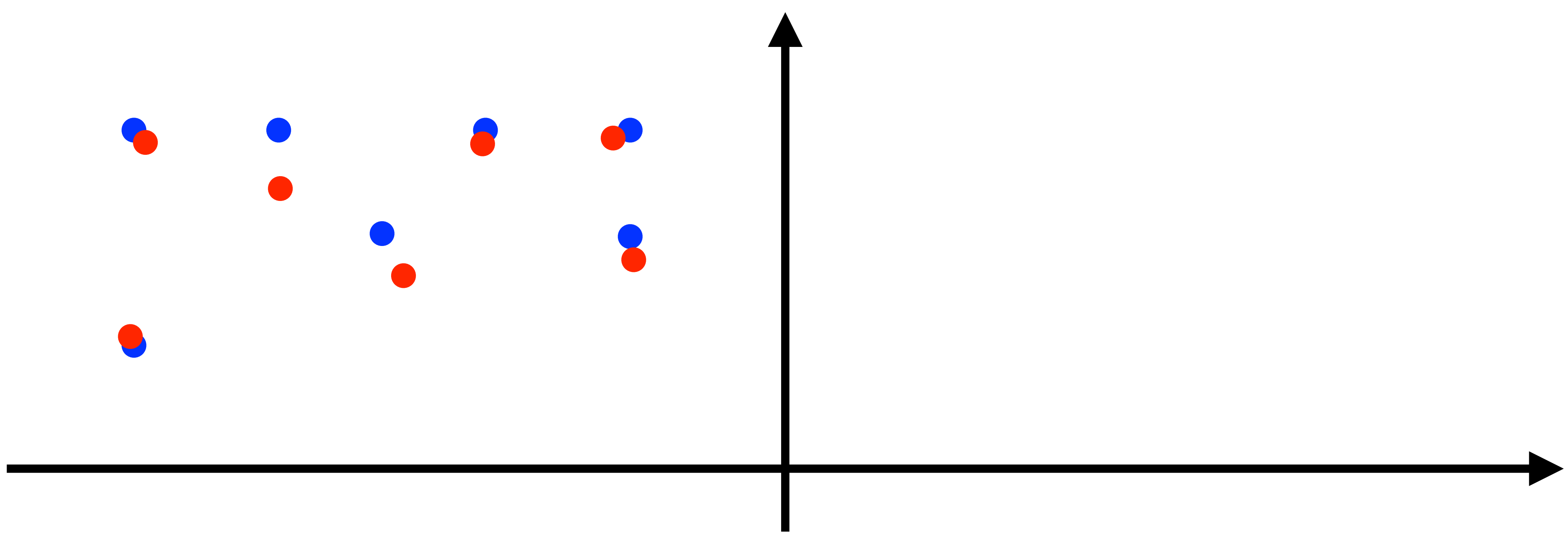
$$\mathbf{R}^*, \mathbf{t}^* = \arg\min_{\mathbf{R}\in SO(3), \mathbf{t}\in\mathcal{R}^3} \sum_i \|\mathbf{R}\mathbf{p}_i + \mathbf{t} - \mathbf{q}_i\|_2^2$$

Solution: estimate covariante matrix: $\mathbf{H} = \sum_i \mathbf{p}'_i \mathbf{q}'^{\top}_i$

find SVD decomposition: $\mathbf{H} = \mathbf{U}\mathbf{S}\mathbf{V}^\top$

estimate optimal rotation: $\mathbf{R}^* = \mathbf{V}\mathbf{U}^\top$

$$\mathbf{t}^* = \widetilde{\mathbf{q}} - \mathbf{R}^*\widetilde{\mathbf{p}}$$



Mutual calibration of two coordinate frames

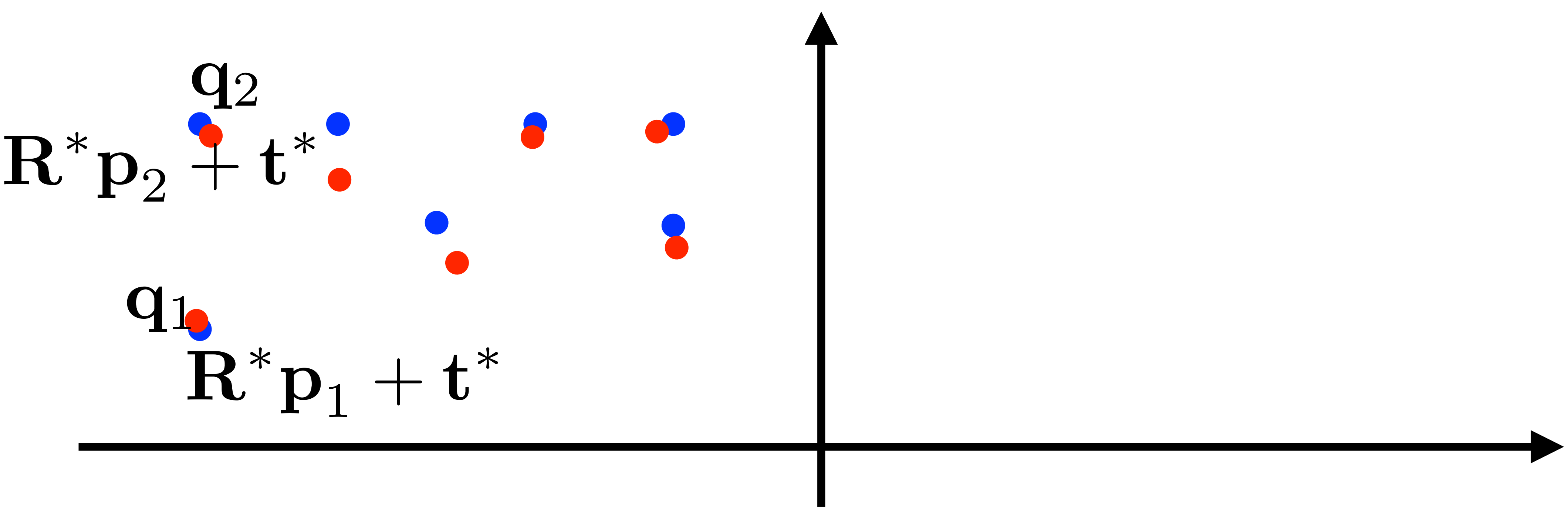
$$\mathbf{R}^*, \mathbf{t}^* = \arg\min_{\mathbf{R} \in SO(3), \mathbf{t} \in \mathcal{R}^3} \sum_i \|\mathbf{R}\mathbf{p}_i + \mathbf{t} - \mathbf{q}_i\|_2^2$$

Solution: estimate covariante matrix: $\mathbf{H} = \sum_i \mathbf{p}'_i \mathbf{q}'^{\top}_i = \mathbf{P}\mathbf{Q}^{\top}$

find SVD decomposition: $\mathbf{H} = \mathbf{U}\mathbf{S}\mathbf{V}^{\top}$

estimate optimal rotation: $\mathbf{R}^* = \mathbf{V}\mathbf{U}^{\top}$

$$\mathbf{t}^* = \widetilde{\mathbf{q}} - \mathbf{R}^* \widetilde{\mathbf{p}}$$



Mutual calibration of two coordinate frames

(1) Record pointclouds and manually estimate 3D-3D correspondences

(2) Solve: $$\mathbf{R}^*, \mathbf{t}^* = \operatorname*{arg\,min}_{\mathbf{R}\in SO(3), \mathbf{t}\in\mathcal{R}^3} \sum_i \|\mathbf{R}\mathbf{p}_i + \mathbf{t} - \mathbf{q}_i\|_2^2$$

Solution: $$\mathbf{R}^* = \mathbf{V}\mathbf{U}^\top$$
$$\mathbf{t}^* = \widetilde{\mathbf{q}} - \mathbf{R}^*\widetilde{\mathbf{p}}$$

In python:

```
H = P @ Q.T
U, S, V = np.linalg.svd(H, full_matrices=True)
```

Broadcasting static transformation between two c.f. in ROS:

```
broadcaster = tf2_ros.StaticTransformBroadcaster()
transform = geometry_msgs.msg.TransformStamped()
# compute transform from 3D-3D correspondences
broadcaster.sendTransform(transform)
```

Mutual calibration of two coordinate frames

- Application in Robotics for SLAM.
- Application in Computer graphics for alignment of 3D models

Proof [Arun-TPAMI-87]

$$\mathbf{R}^*, \mathbf{t}^* = \arg\min_{\mathbf{R}\in SO(3), \mathbf{t}\in\mathcal{R}^3} \sum_i \|\mathbf{R}\mathbf{p}_i + \mathbf{t} - \mathbf{q}_i\|_2^2 =$$

Proof [Arun-TPAMI-87]

$$\mathbf{R}^*, \mathbf{t}^* = \underset{\mathbf{R} \in SO(3), \mathbf{t} \in \mathcal{R}^3}{\arg\min} \sum_i \|\mathbf{R}\mathbf{p}_i + \mathbf{t} - \mathbf{q}_i\|_2^2 =$$

$$= \underset{\mathbf{R} \in SO(3), \mathbf{t} \in \mathcal{R}^3}{\arg\min} \sum_i \|\mathbf{R}(\mathbf{p}'_i + \widetilde{\mathbf{p}}) + \mathbf{t} - \mathbf{q}'_i - \widetilde{\mathbf{q}}\|_2^2 =$$

Proof [Arun-TPAMI-87]

$$\mathbf{R}^*, \mathbf{t}^* = \arg\min_{\mathbf{R}\in SO(3), \mathbf{t}\in\mathcal{R}^3} \sum_i \|\mathbf{R}\mathbf{p}_i + \mathbf{t} - \mathbf{q}_i\|_2^2 =$$

$$= \arg\min_{\mathbf{R}\in SO(3), \mathbf{t}\in\mathcal{R}^3} \sum_i \|\mathbf{R}(\mathbf{p}'_i + \widetilde{\mathbf{p}}) + \mathbf{t} - \mathbf{q}'_i - \widetilde{\mathbf{q}}\|_2^2 =$$

$$= \arg\min_{\mathbf{R}\in SO(3), \mathbf{t}\in\mathcal{R}^3} \sum_i \|\mathbf{R}\mathbf{p}'_i - \mathbf{q}'_i + \underbrace{\mathbf{R}\widetilde{\mathbf{p}} + \mathbf{t} - \widetilde{\mathbf{q}}}_{\mathbf{t}'}\|_2^2 =$$

Proof [Arun-TPAMI-87]

$$\mathbf{R}^*, \mathbf{t}^* = \underset{\mathbf{R}\in SO(3), \mathbf{t}\in\mathcal{R}^3}{\arg\min} \sum_i \|\mathbf{R}\mathbf{p}_i + \mathbf{t} - \mathbf{q}_i\|_2^2 =$$

$$= \underset{\mathbf{R}\in SO(3), \mathbf{t}\in\mathcal{R}^3}{\arg\min} \sum_i \|\mathbf{R}(\mathbf{p}'_i + \widetilde{\mathbf{p}}) + \mathbf{t} - \mathbf{q}'_i - \widetilde{\mathbf{q}}\|_2^2 =$$

$$= \underset{\mathbf{R}\in SO(3), \mathbf{t}\in\mathcal{R}^3}{\arg\min} \sum_i \|\mathbf{R}\mathbf{p}'_i - \mathbf{q}'_i + \underbrace{\mathbf{R}\widetilde{\mathbf{p}} + \mathbf{t} - \widetilde{\mathbf{q}}}_{\mathbf{t}'}\|_2^2 =$$

$$= \underset{\mathbf{R}\in SO(3), \mathbf{t}\in\mathcal{R}^3}{\arg\min} \sum_i (\mathbf{R}\mathbf{p}'_i - \mathbf{q}'_i + \mathbf{t}')^\top (\mathbf{R}\mathbf{p}'_i - \mathbf{q}'_i + \mathbf{t}') =$$

Proof [Arun-TPAMI-87]

$$\mathbf{R}^*, \mathbf{t}^* = \arg\min_{\mathbf{R}\in SO(3), \mathbf{t}\in\mathcal{R}^3} \sum_i \|\mathbf{R}\mathbf{p}_i + \mathbf{t} - \mathbf{q}_i\|_2^2 =$$

$$= \arg\min_{\mathbf{R}\in SO(3), \mathbf{t}\in\mathcal{R}^3} \sum_i \|\mathbf{R}(\mathbf{p}_i' + \widetilde{\mathbf{p}}) + \mathbf{t} - \mathbf{q}_i' - \widetilde{\mathbf{q}}\|_2^2 =$$

$$= \arg\min_{\mathbf{R}\in SO(3), \mathbf{t}\in\mathcal{R}^3} \sum_i \|\mathbf{R}\mathbf{p}_i' - \mathbf{q}_i' + \underbrace{\mathbf{R}\widetilde{\mathbf{p}} + \mathbf{t} - \widetilde{\mathbf{q}}}_{\mathbf{t}'}\|_2^2 =$$

$$= \arg\min_{\mathbf{R}\in SO(3), \mathbf{t}\in\mathcal{R}^3} \sum_i (\mathbf{R}\mathbf{p}_i' - \mathbf{q}_i' + \mathbf{t}')^\top (\mathbf{R}\mathbf{p}_i' - \mathbf{q}_i' + \mathbf{t}') =$$

$$= \arg\min_{\mathbf{R}\in SO(3), \mathbf{t}\in\mathcal{R}^3} \sum_i \|\mathbf{R}\mathbf{p}_i' - \mathbf{q}_i'\|_2^2 + \underbrace{\sum_i 2(\mathbf{R}\mathbf{p}_i' - \mathbf{q}_i')\mathbf{t}'}_{=0} + \|\mathbf{t}'\|_2^2 =$$

Proof [Arun-TPAMI-87]

$$\mathbf{R}^*, \mathbf{t}^* = \underset{\mathbf{R}\in SO(3), \mathbf{t}\in\mathcal{R}^3}{\arg\min} \sum_i \|\mathbf{R}\mathbf{p}_i + \mathbf{t} - \mathbf{q}_i\|_2^2 =$$

$$= \underset{\mathbf{R}\in SO(3), \mathbf{t}\in\mathcal{R}^3}{\arg\min} \sum_i \|\mathbf{R}(\mathbf{p}'_i + \widetilde{\mathbf{p}}) + \mathbf{t} - \mathbf{q}'_i - \widetilde{\mathbf{q}}\|_2^2 =$$

$$= \underset{\mathbf{R}\in SO(3), \mathbf{t}\in\mathcal{R}^3}{\arg\min} \sum_i \|\mathbf{R}\mathbf{p}'_i - \mathbf{q}'_i + \underbrace{\mathbf{R}\widetilde{\mathbf{p}} + \mathbf{t} - \widetilde{\mathbf{q}}}_{\mathbf{t}'}\|_2^2 =$$

$$= \underset{\mathbf{R}\in SO(3), \mathbf{t}\in\mathcal{R}^3}{\arg\min} \sum_i (\mathbf{R}\mathbf{p}'_i - \mathbf{q}'_i + \mathbf{t}')^\top (\mathbf{R}\mathbf{p}'_i - \mathbf{q}'_i + \mathbf{t}') =$$

$$= \underset{\mathbf{R}\in SO(3), \mathbf{t}\in\mathcal{R}^3}{\arg\min} \sum_i \|\mathbf{R}\mathbf{p}'_i - \mathbf{q}'_i\|_2^2 + \underbrace{\sum_i 2(\mathbf{R}\mathbf{p}'_i - \mathbf{q}'_i)\mathbf{t}'}_{=0} + \|\mathbf{t}'\|_2^2 =$$

$$= \underset{\mathbf{R}\in SO(3), \mathbf{t}\in\mathcal{R}^3}{\arg\min} \sum_i \|\mathbf{R}\mathbf{p}'_i - \mathbf{q}'_i\|_2^2 + \|\mathbf{t}'\|_2^2$$

we can reach second term zero by $\mathbf{t} = \widetilde{\mathbf{q}} - \mathbf{R}\widetilde{\mathbf{p}} = \mathbf{t}^*$

Proof [Arun-TPAMI-87]

$$= \arg\min_{\mathbf{R}\in SO(3),\mathbf{t}\in\mathcal{R}^3} \sum_i \|\mathbf{R}\mathbf{p}'_i - \mathbf{q}'_i\|_2^2 + \|\mathbf{t}'\|_2^2$$

we can reach second term zero by $\mathbf{t} = \tilde{\mathbf{q}} - \mathbf{R}\tilde{\mathbf{p}} = \mathbf{t}^*$

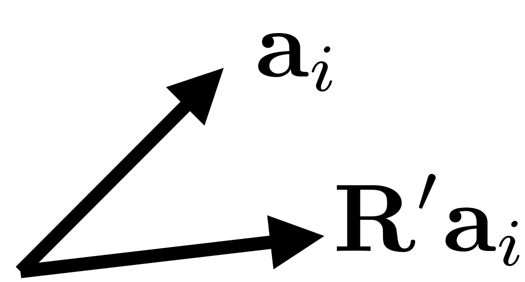
$$\arg\min_{\mathbf{R}\in SO(3)} \sum_i \|\mathbf{R}\mathbf{p}'_i - \mathbf{q}'_i\|_2^2 = \arg\max_{\mathbf{R}\in SO(3)} \sum_i \mathbf{q}'^{\top}_i \mathbf{R}\mathbf{p}'_i =$$

$$= \arg\max_{\mathbf{R}\in SO(3)} \sum_i \underbrace{\mathbf{q}'^{\top}_i}_{\mathbf{a}_i} \underbrace{\mathbf{R}\mathbf{p}'_i}_{\mathbf{b}_i} = \arg\max_{\mathbf{R}\in SO(3)} \ \text{trace}\, \mathbf{R}\, \underbrace{\mathbf{P}\mathbf{Q}^\top}_{\mathbf{H}} = \mathbf{V}\mathbf{U}^\top$$

$$\arg\max_{\mathbf{R}',\mathbf{R}^*\in SO(3)} \ \text{trace}\, \mathbf{R}'\mathbf{R}^*\mathbf{U}\mathbf{S}\mathbf{V}^\top$$ … expand into two rotations

$$\arg\max_{\mathbf{R}'\in SO(3)} \text{trace}\, \mathbf{R}' \underbrace{\mathbf{V}\mathbf{U}^\top}_{\mathbf{R}^*} \underbrace{\mathbf{U}\mathbf{S}\mathbf{V}^\top}_{\mathbf{H}} = \arg\max_{\mathbf{R}'\in SO(3)} \text{trace}\, \mathbf{R}' \underbrace{(\mathbf{V}\sqrt{\mathbf{S}})}_{\mathbf{A}} \underbrace{(\sqrt{\mathbf{S}}\mathbf{V})^\top}_{\mathbf{A}^\top} =$$

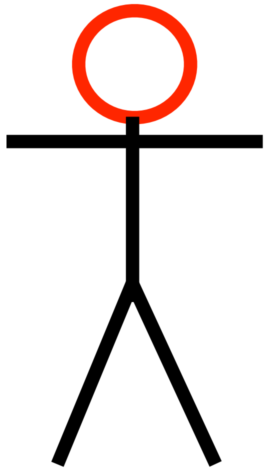
$$= \arg\max_{\mathbf{R}'\in SO(3)} \sum_i \mathbf{a}_i^\top \mathbf{R}'\mathbf{a}_i = \mathbf{E}$$

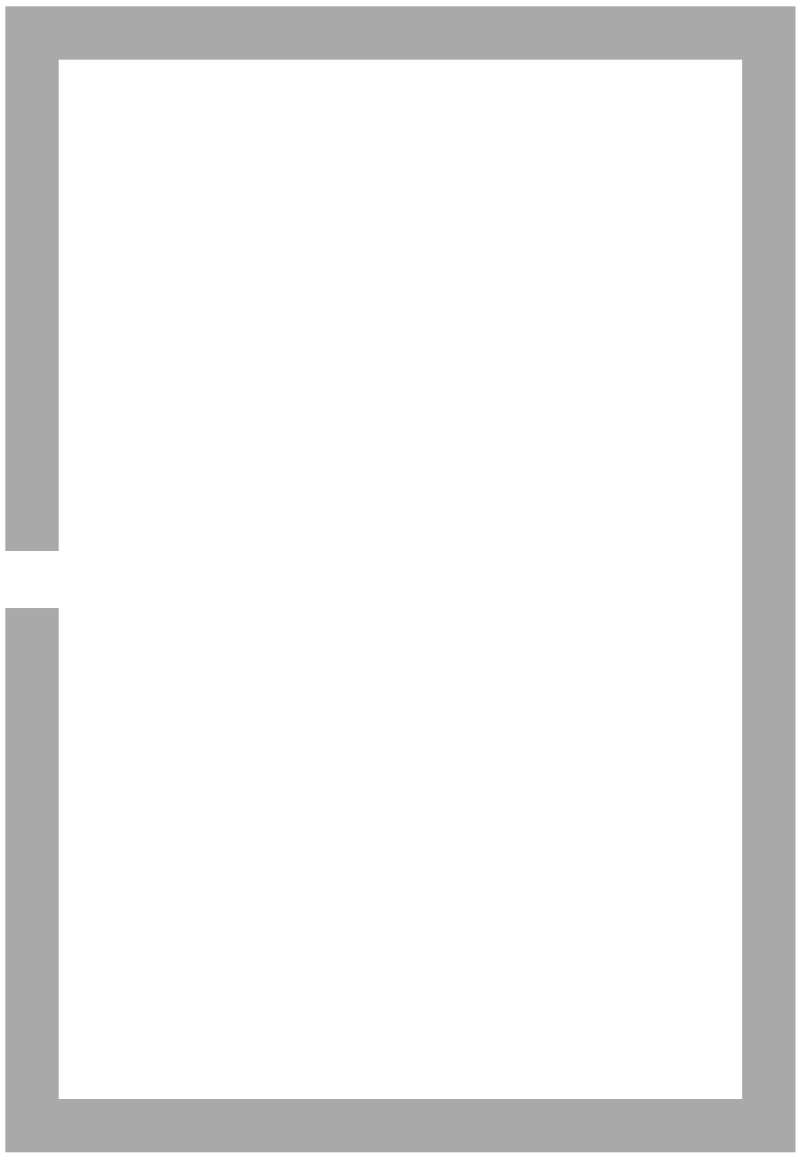


$$\text{trace}\, \mathbf{B}\mathbf{A}^\top = \sum_i \mathbf{a}_i^\top \mathbf{b}_i$$

Camera

Camera model

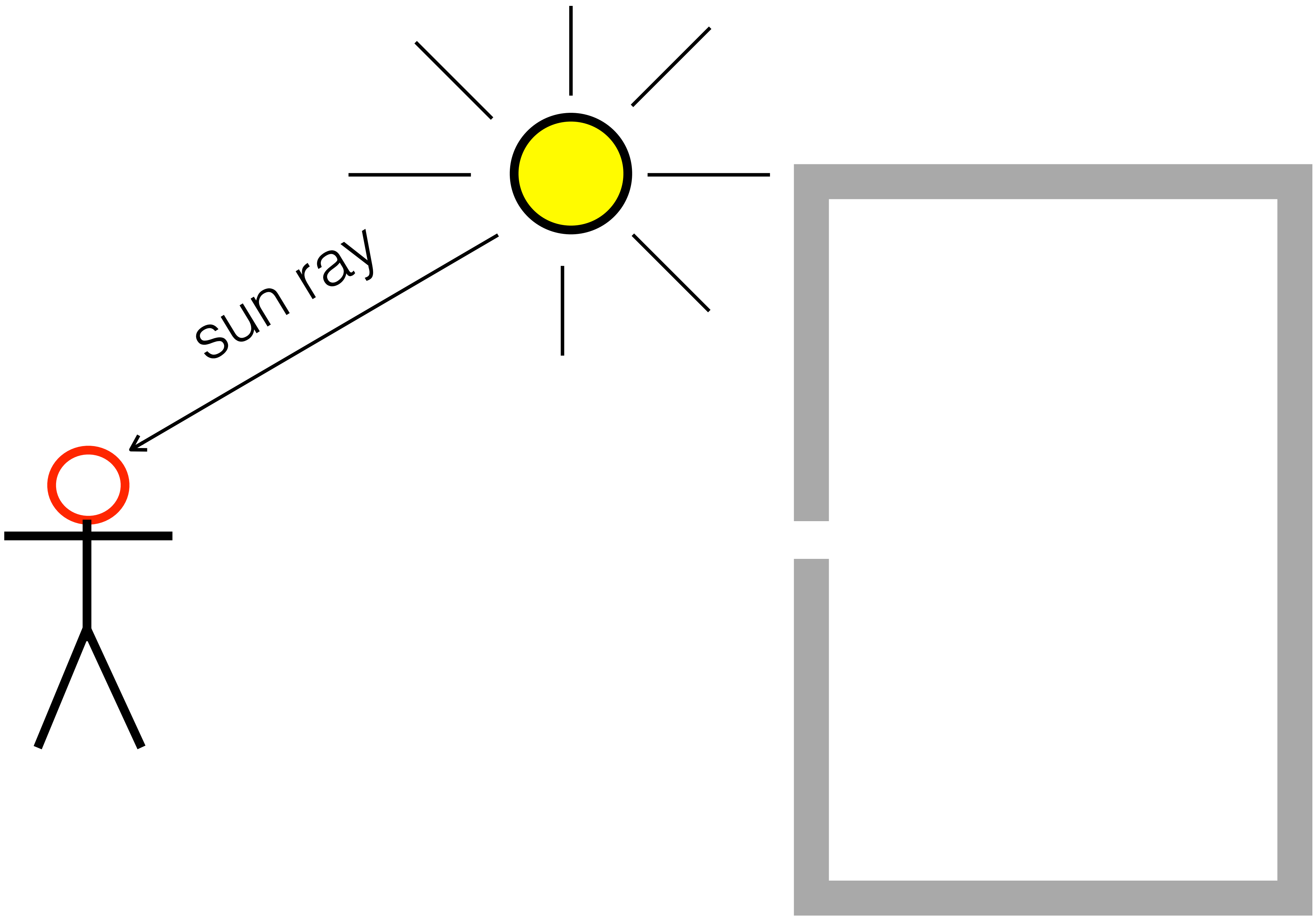




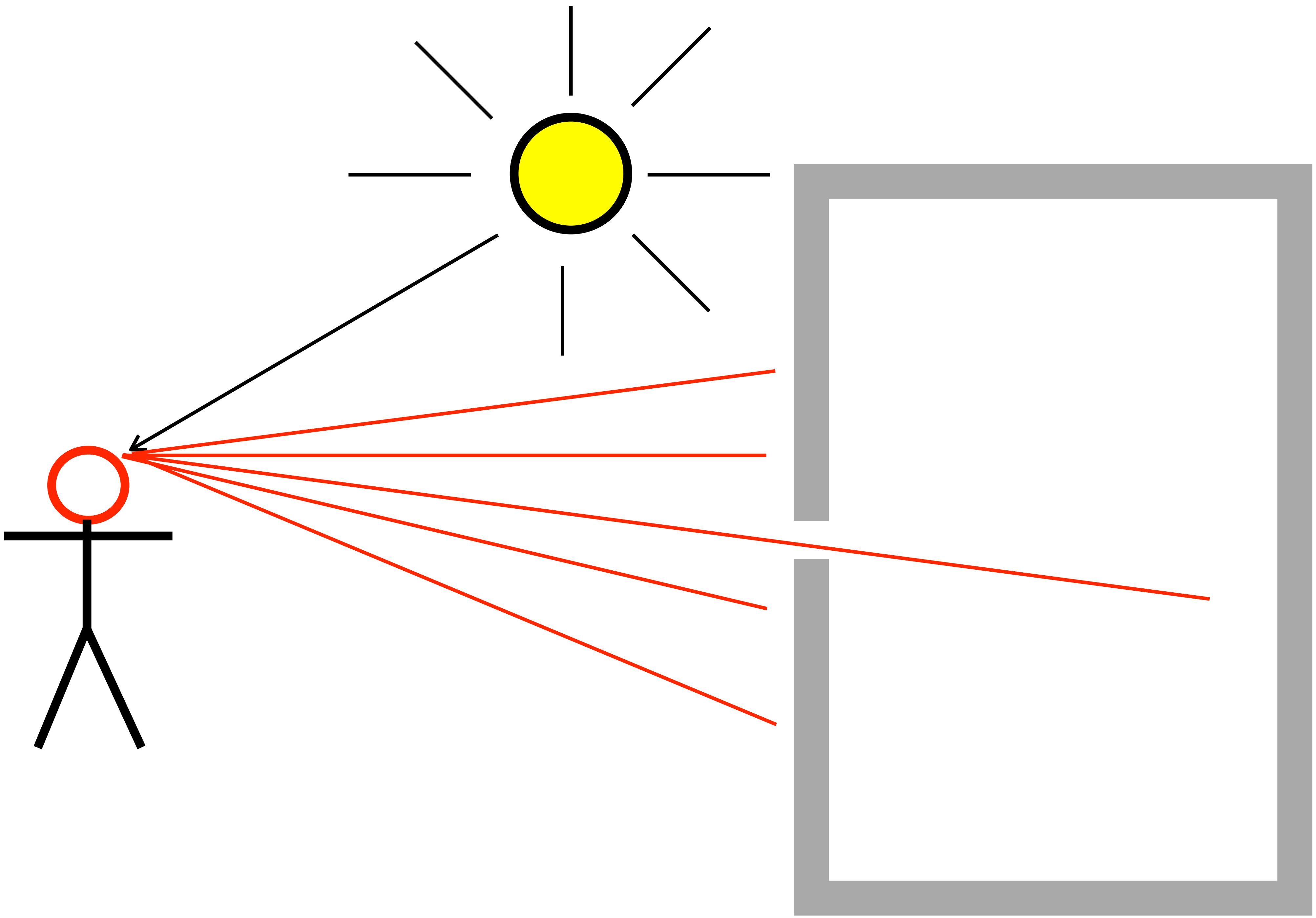
Object in front of camera

Pinhole camera model

Camera model

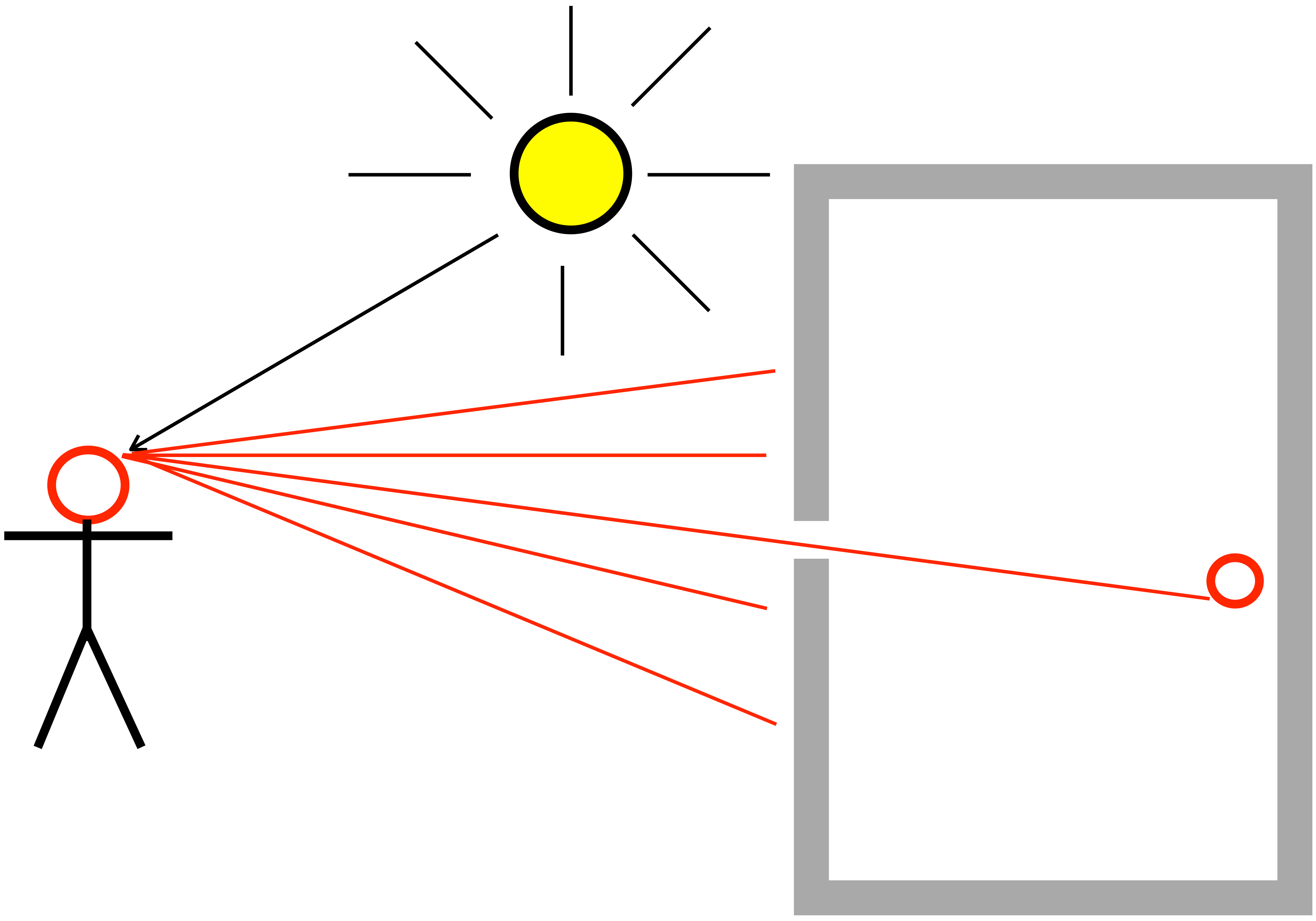


Camera model



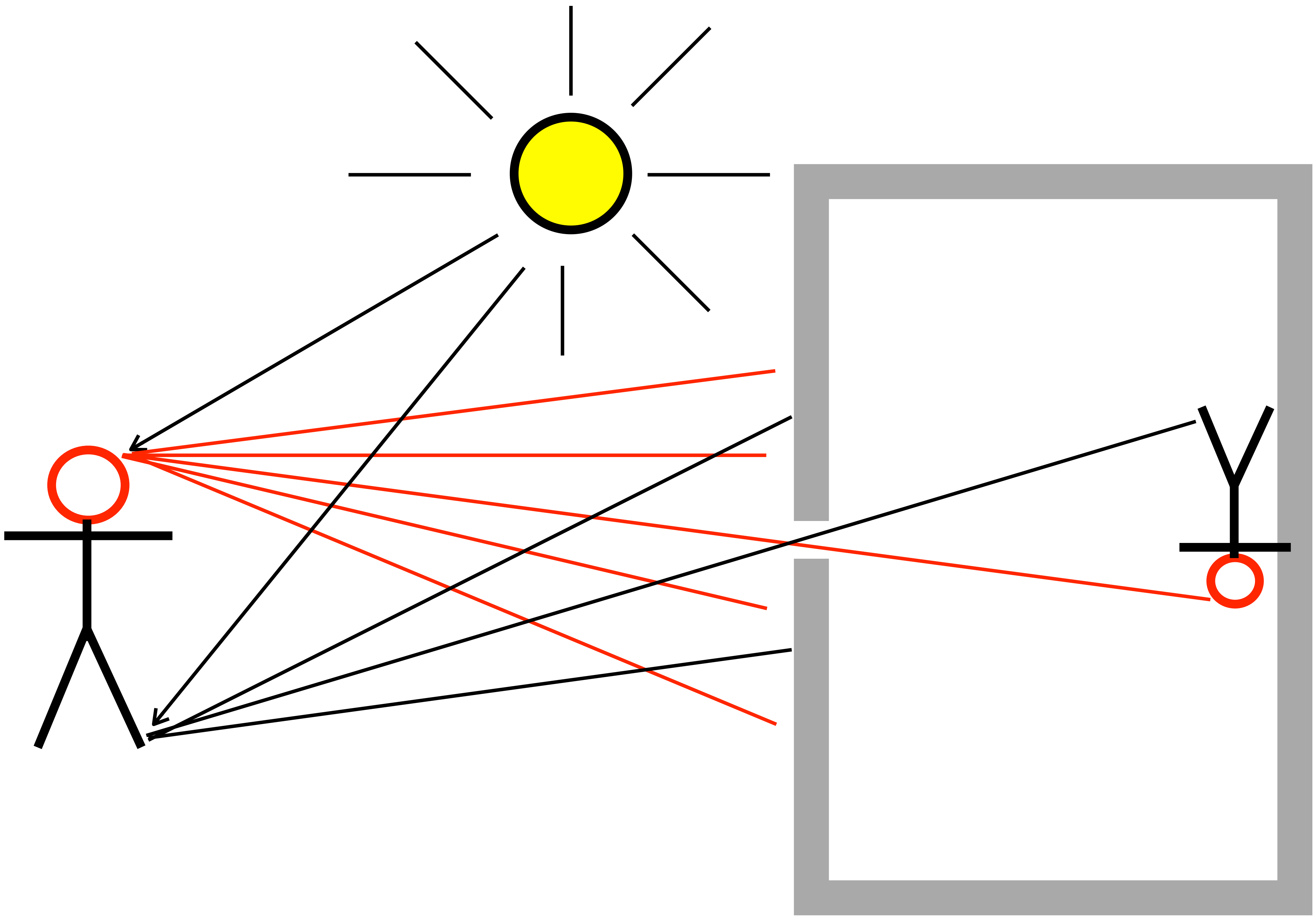
Sun ray is reflected from Lambertian surface in hemisphere

Camera model

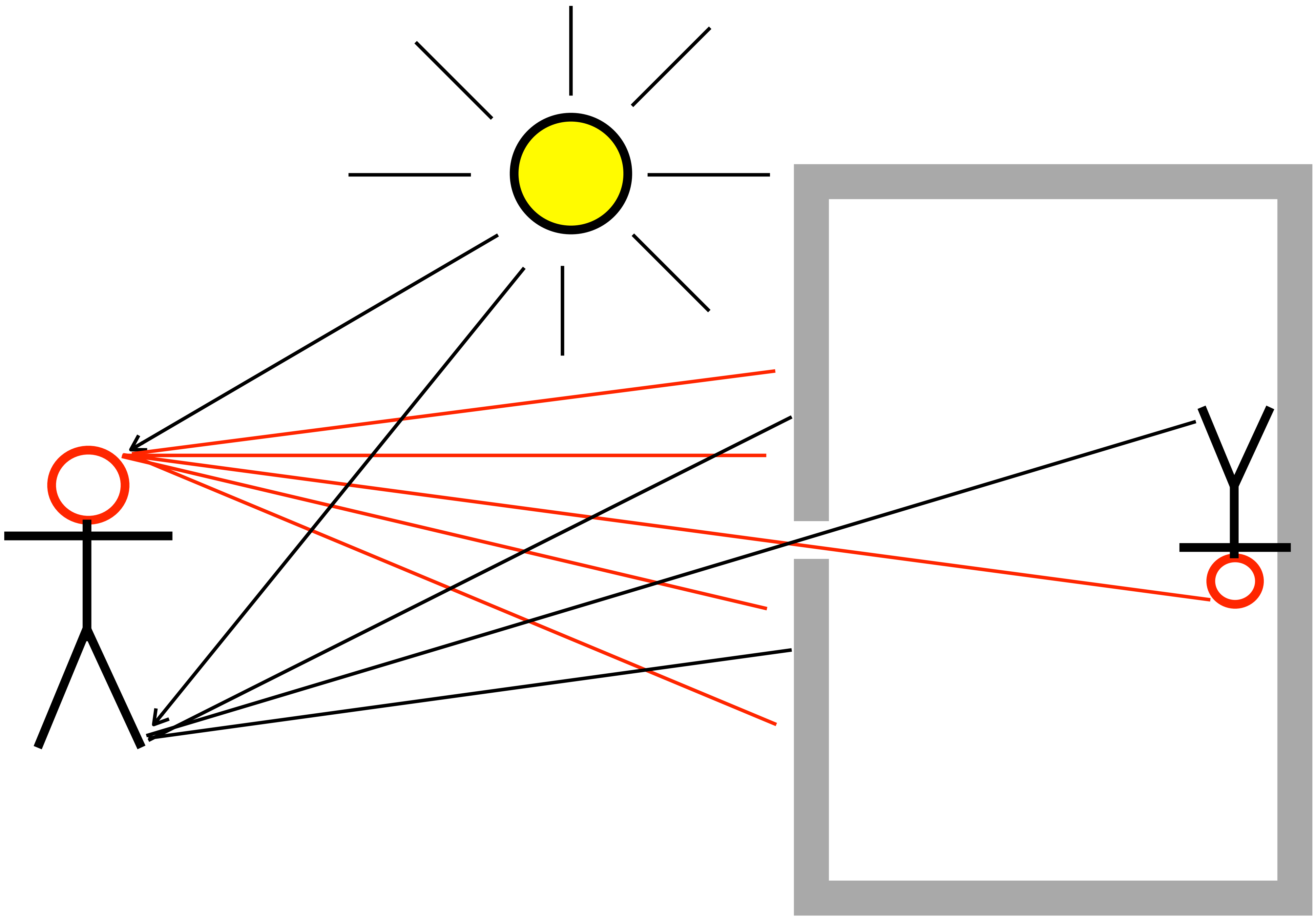


Reflected ray (red) forms inverted image of the object

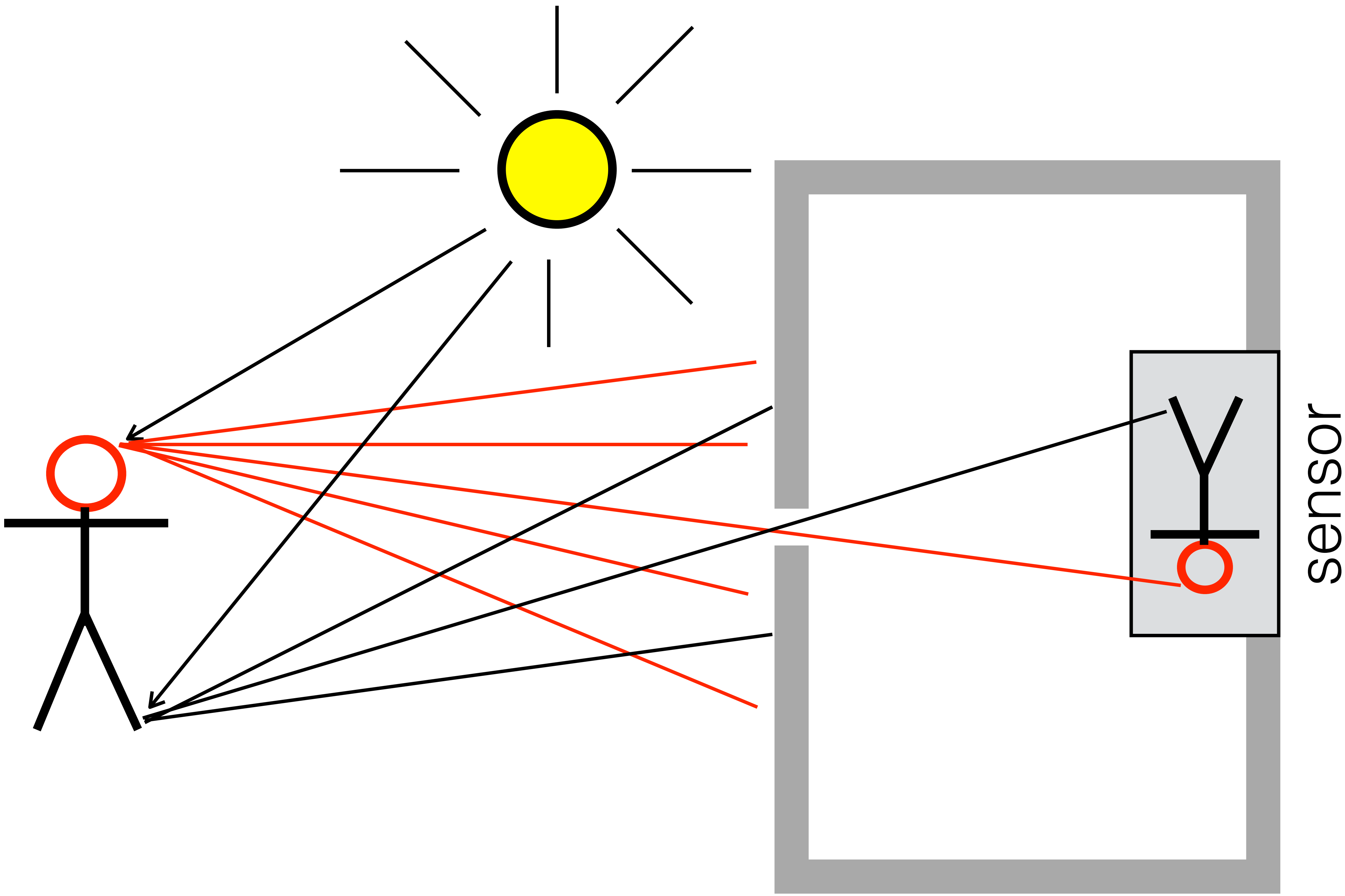
Camera model



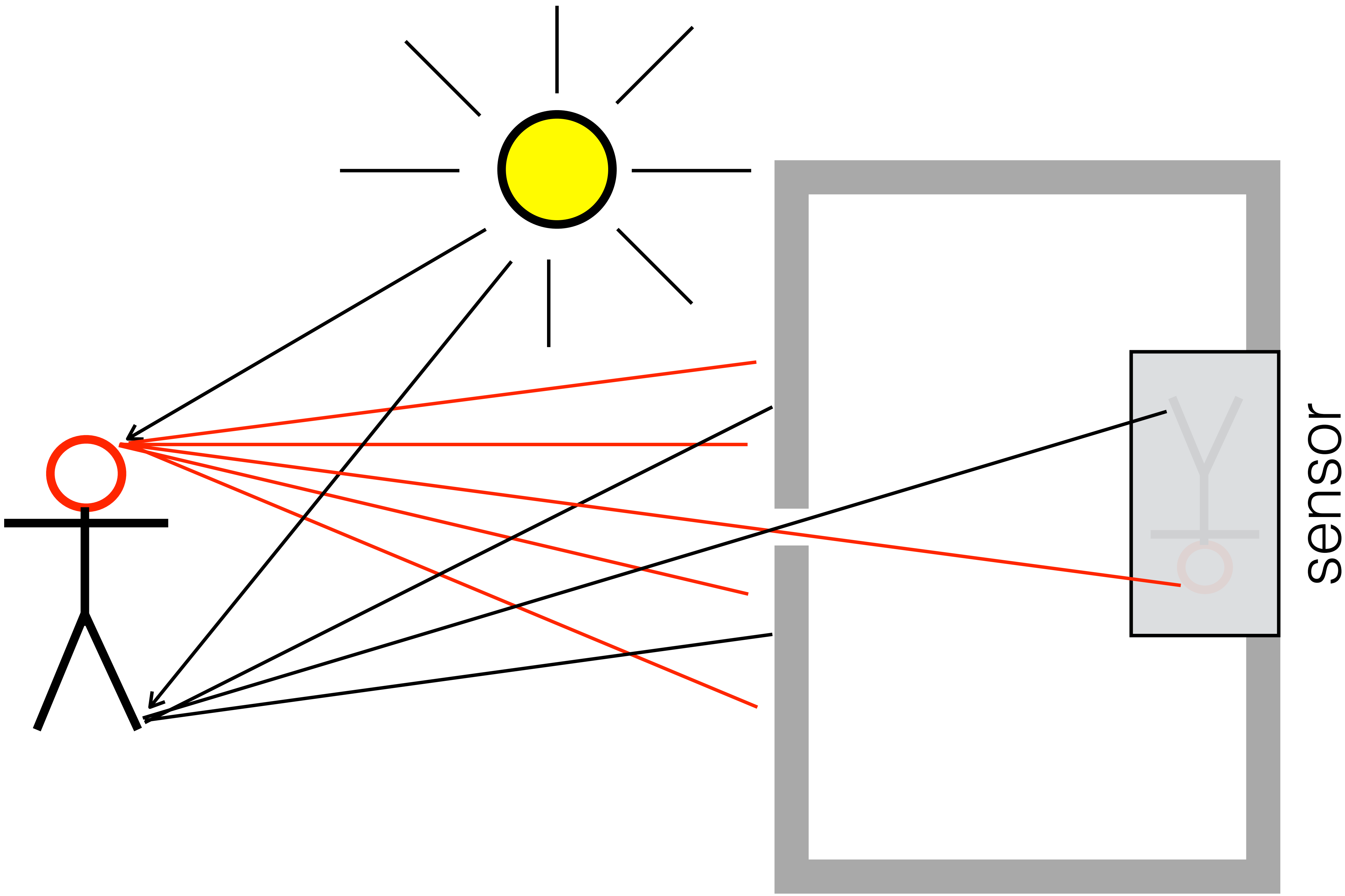
Camera model



Camera model

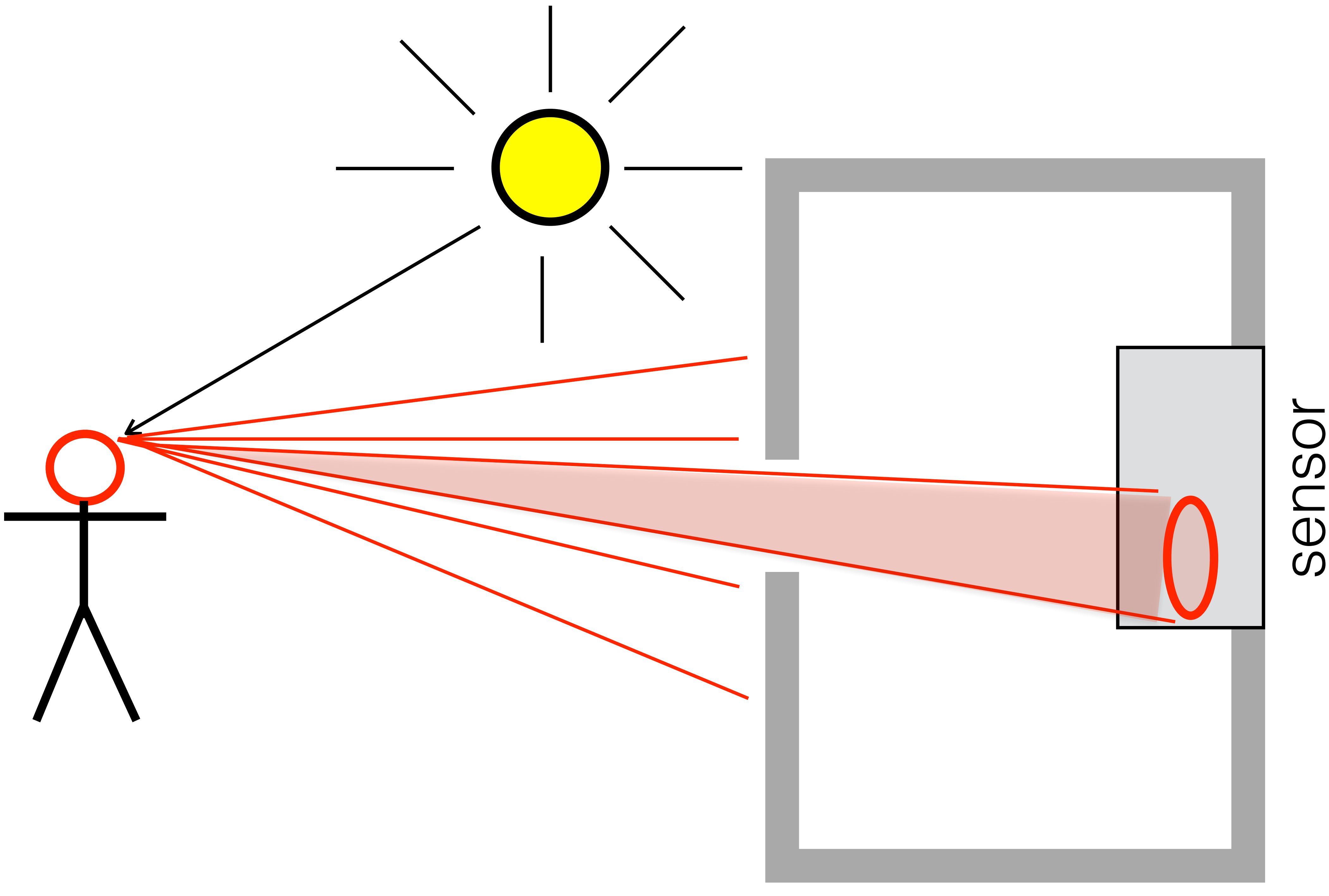


Camera model



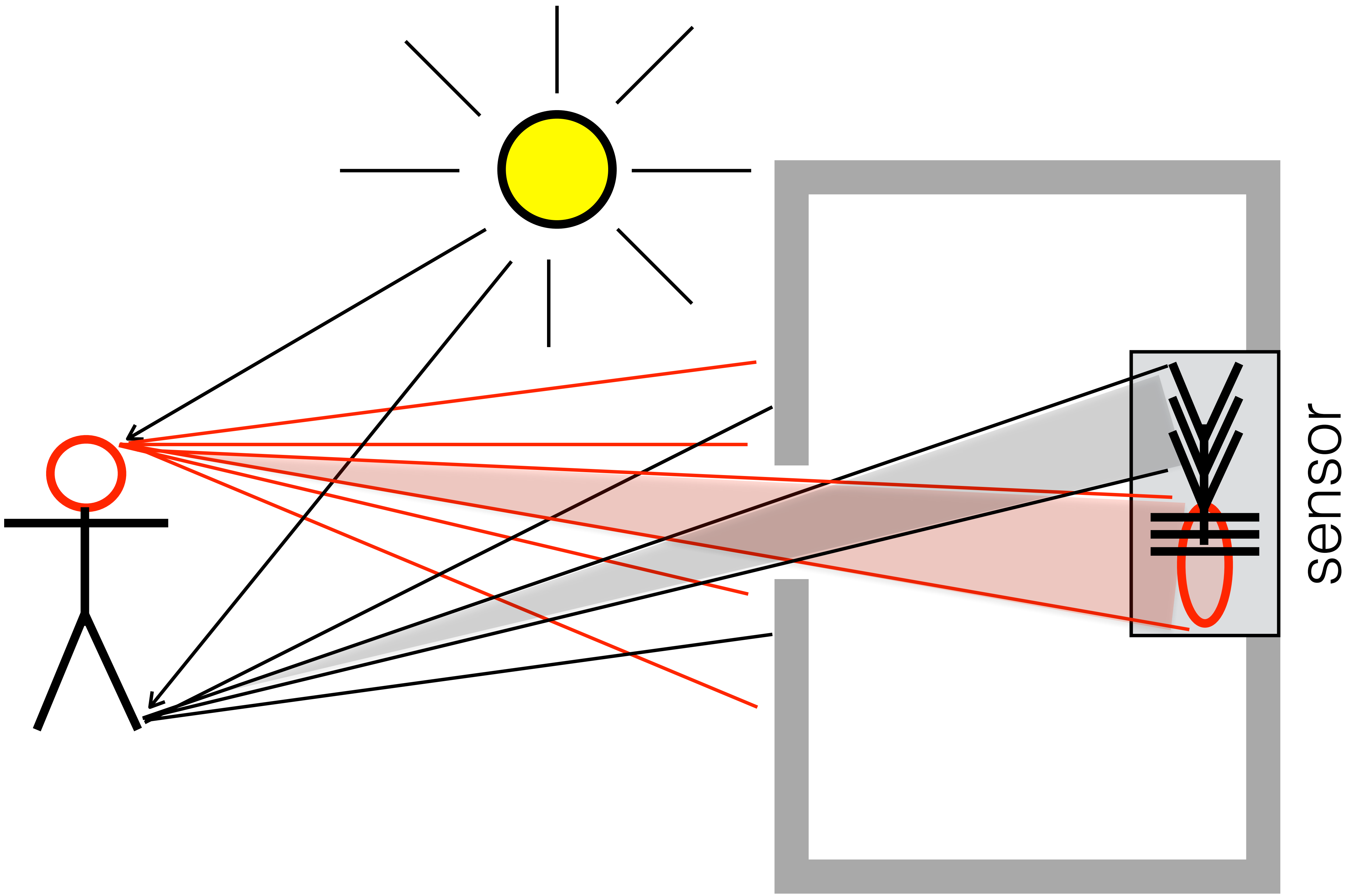
Light energy from rays traversed through pinhole is small

Camera model



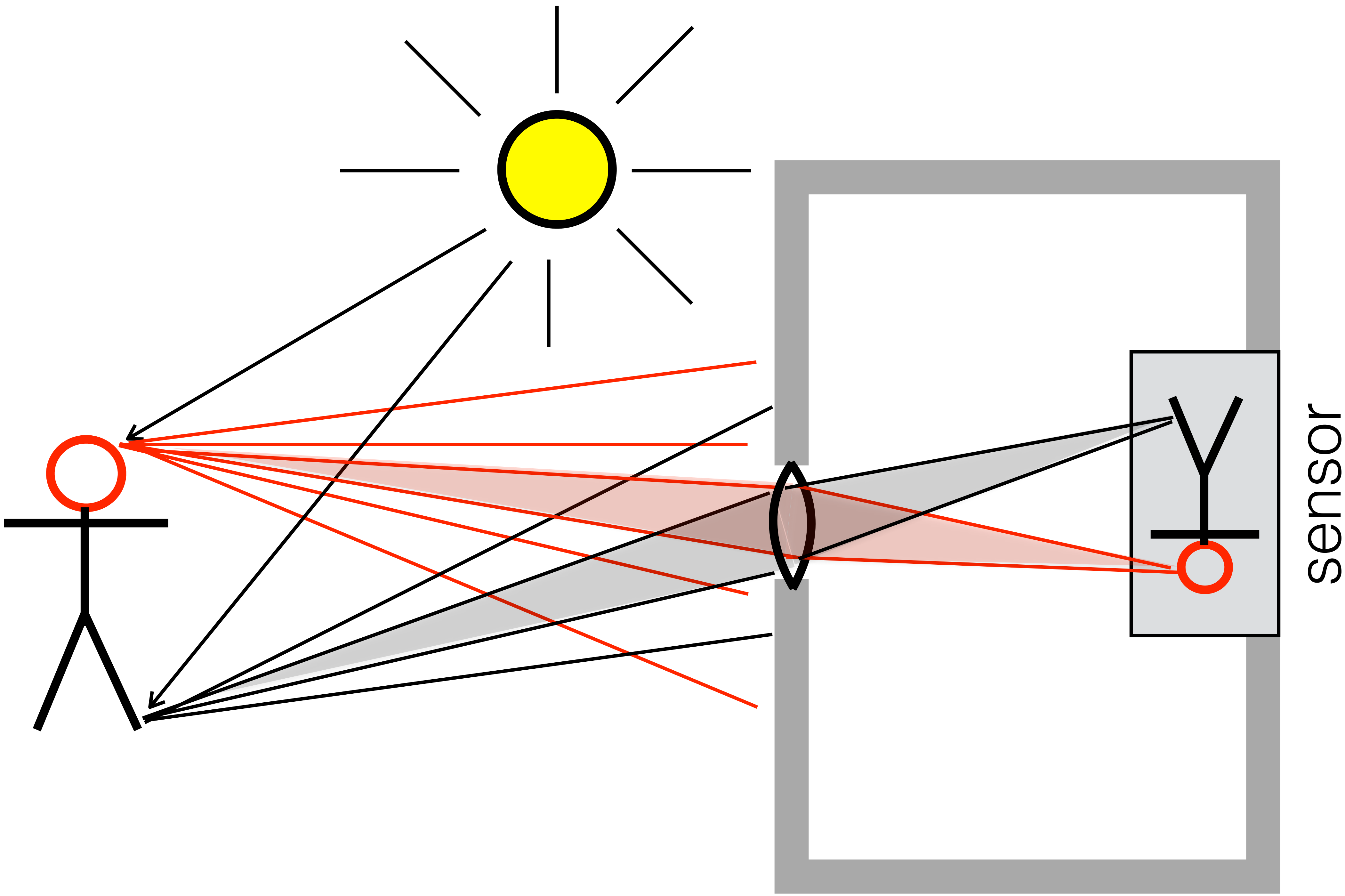
Increasing hole size yields more energy but blurs image

Camera model



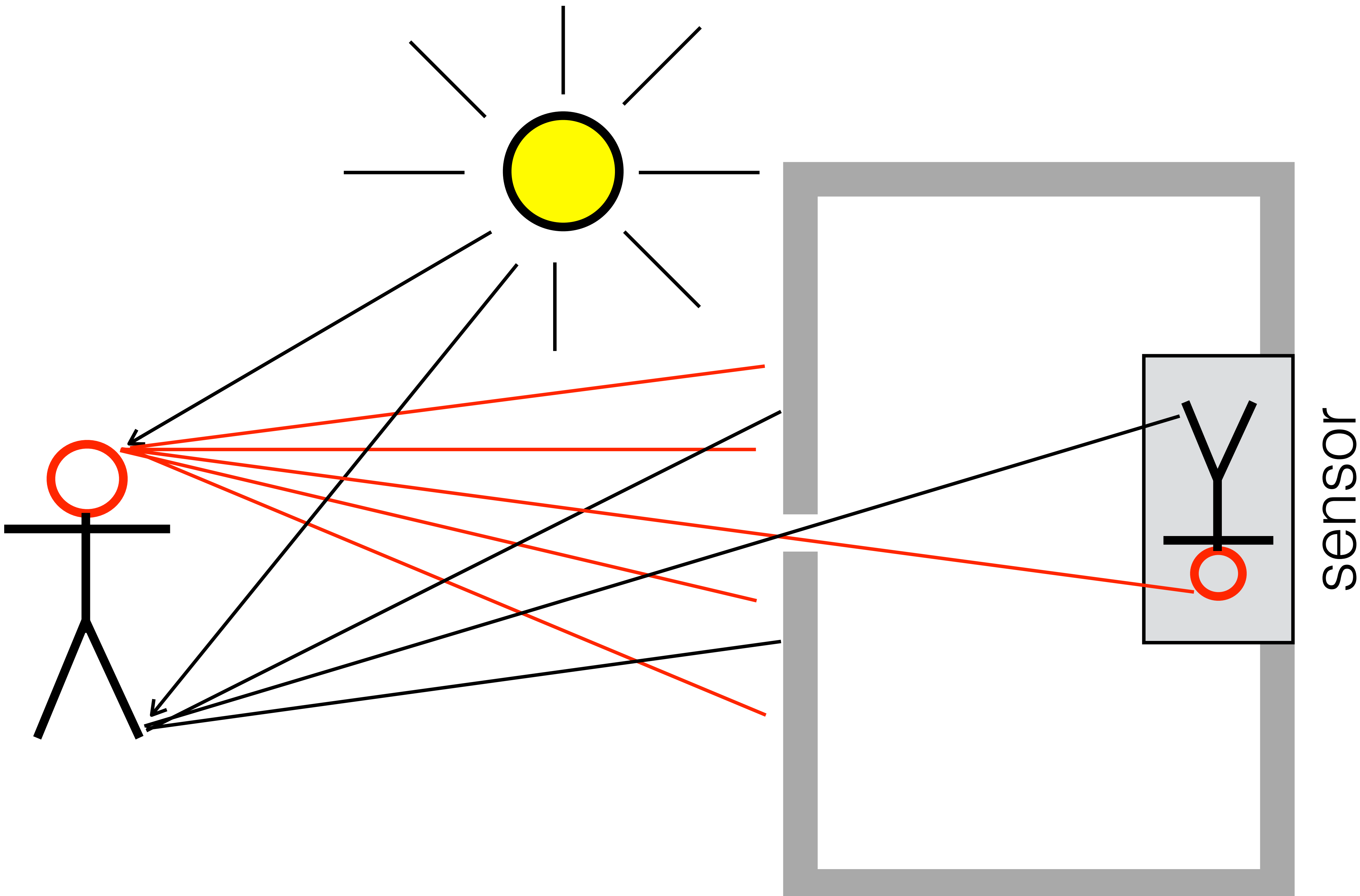
Increasing hole size yields more energy but blurs image

Camera model



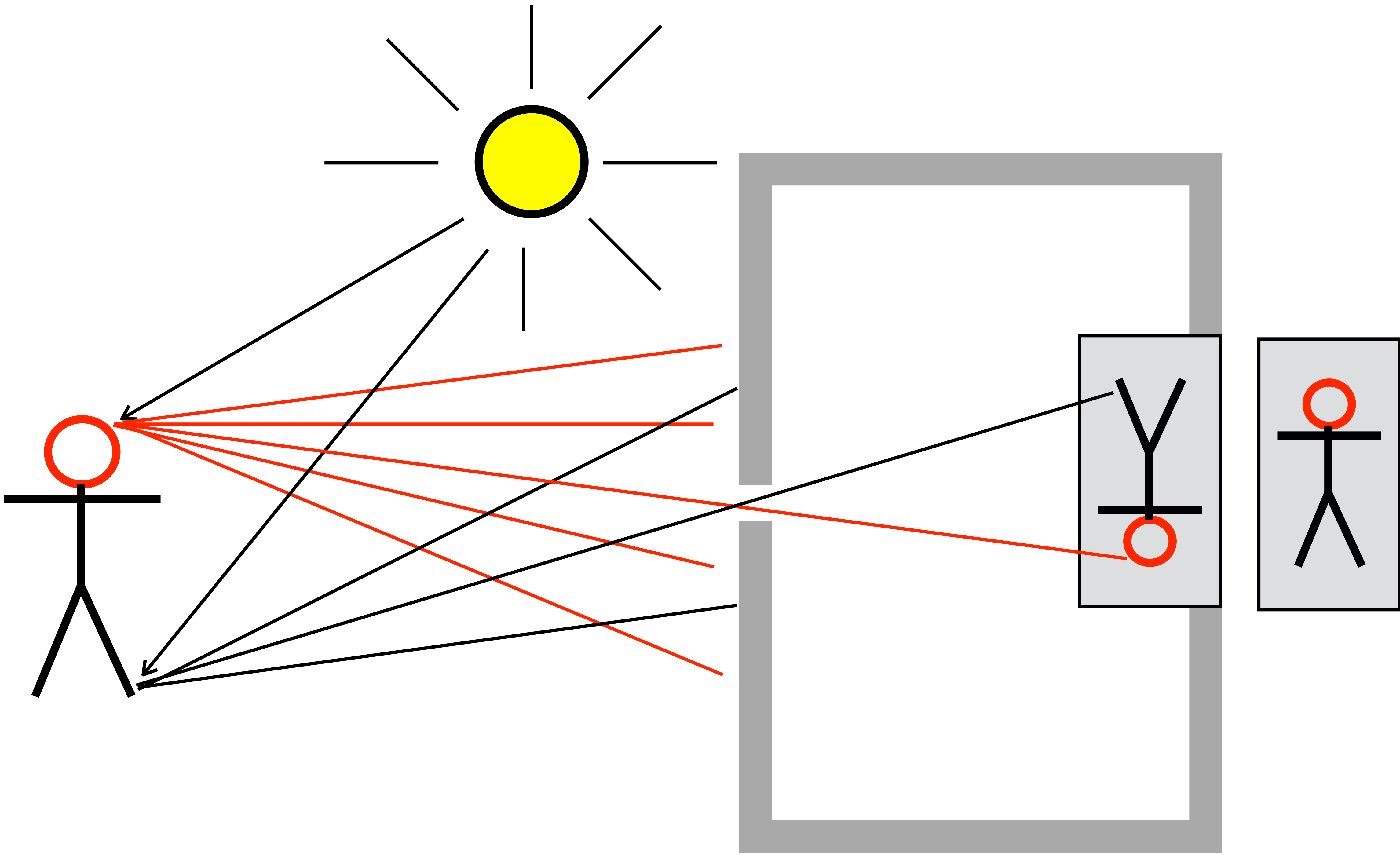
Lens focus cone of light rays in a single point

Camera model



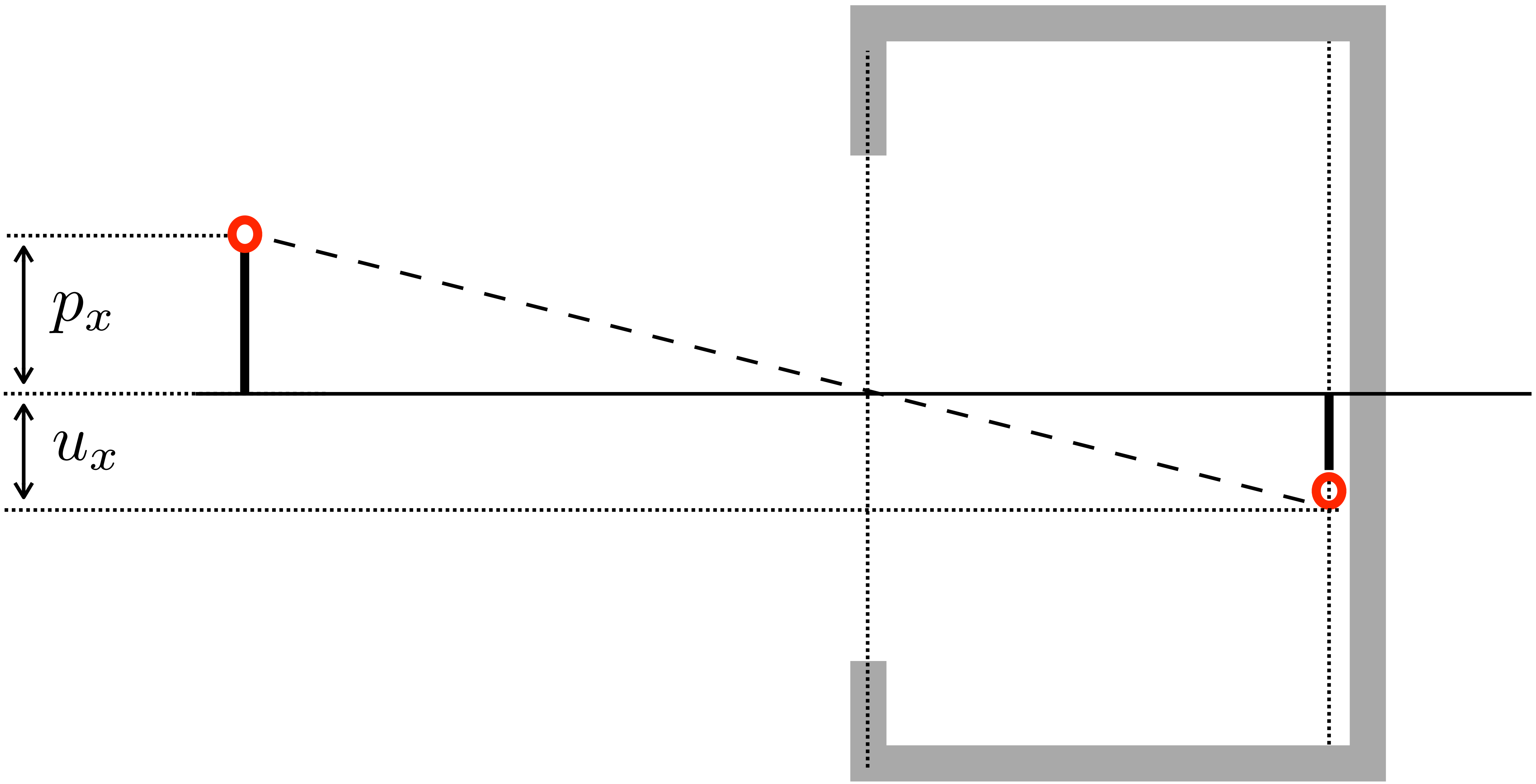
Since source and target of these cones are points, we omit the lens geometry.

Camera model



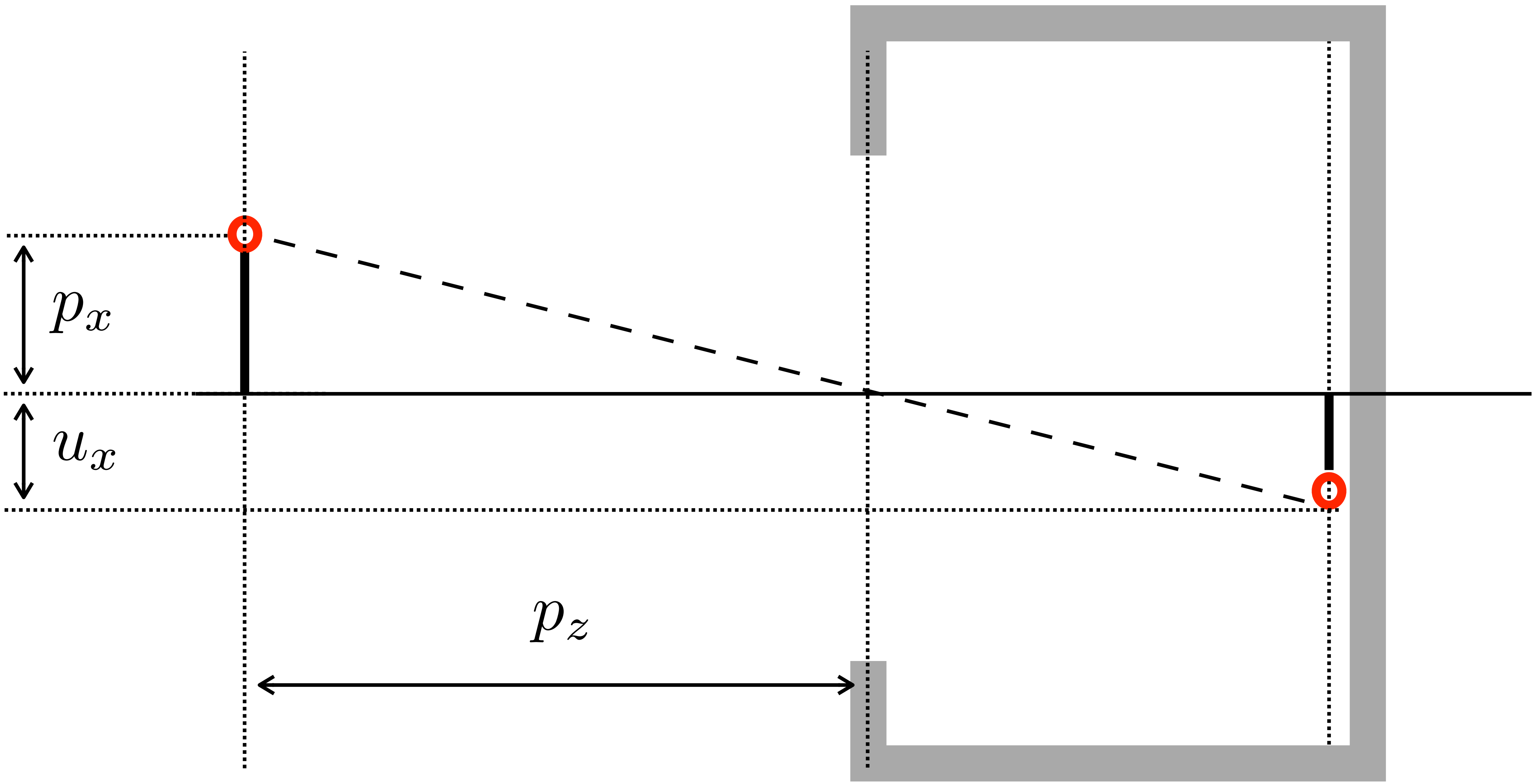
Recorded inverted image is finally flipped on the chip

Projection of 3D point in \camera on image plane

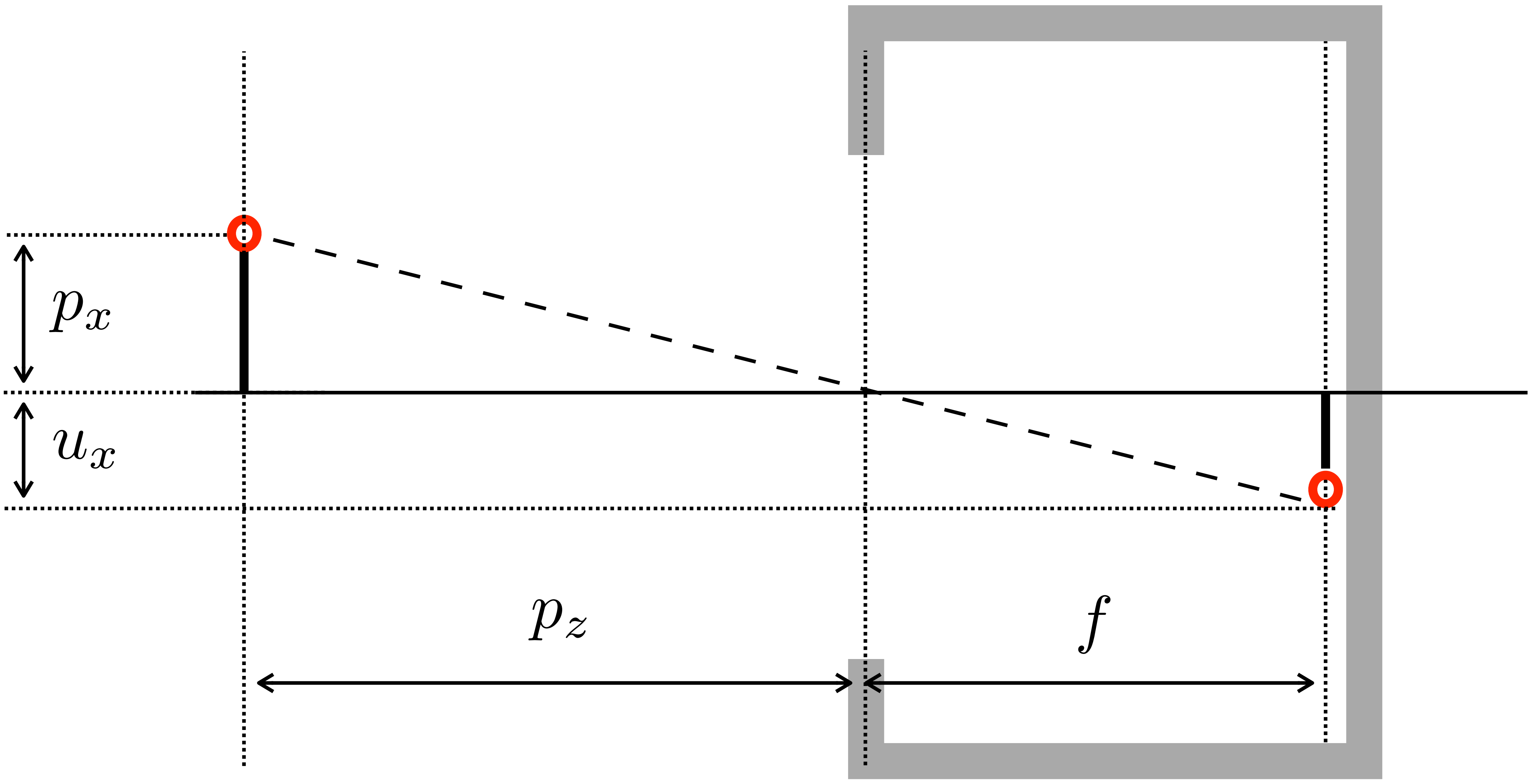


Simplified geometry

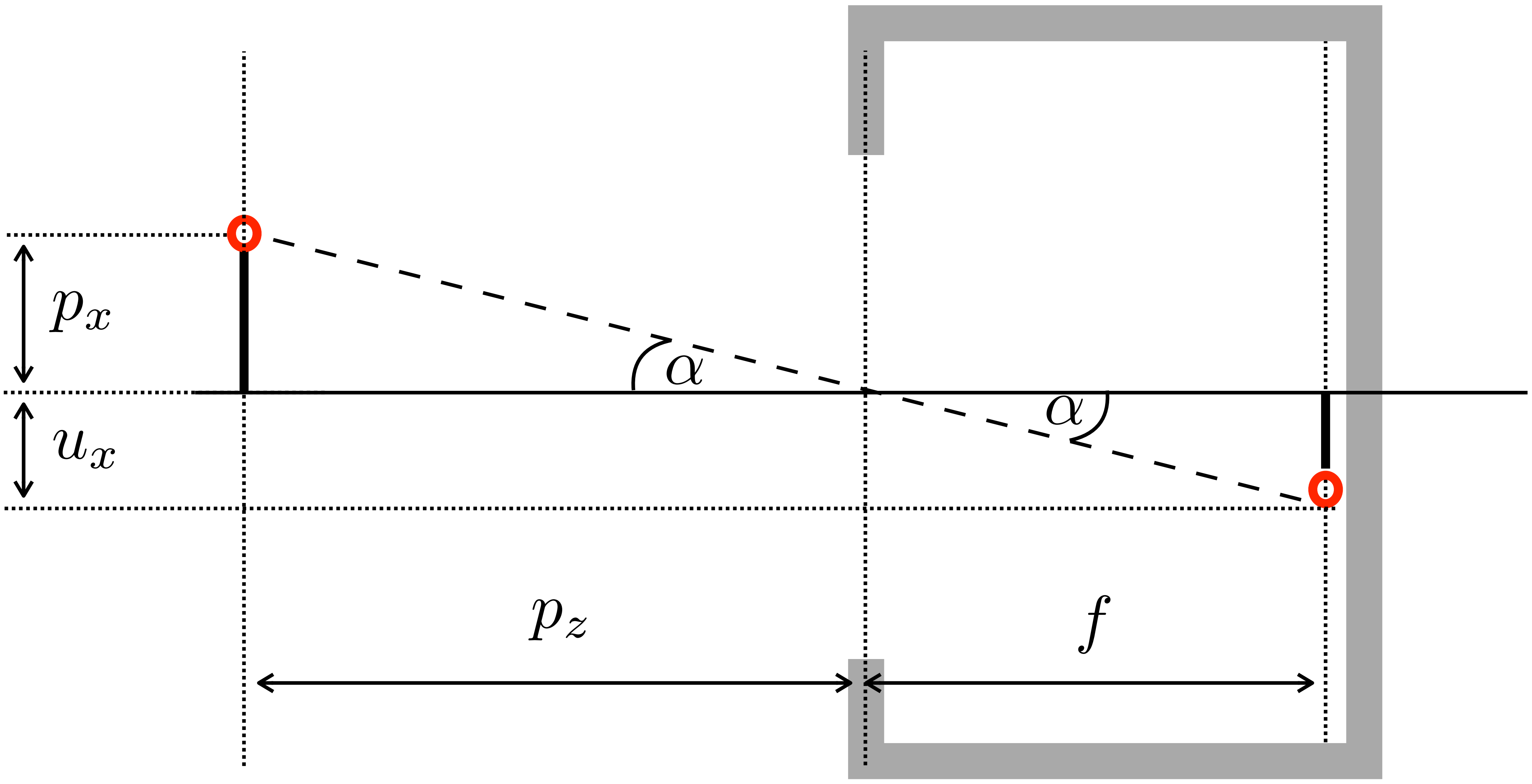
Projection of 3D point in \camera on image plane

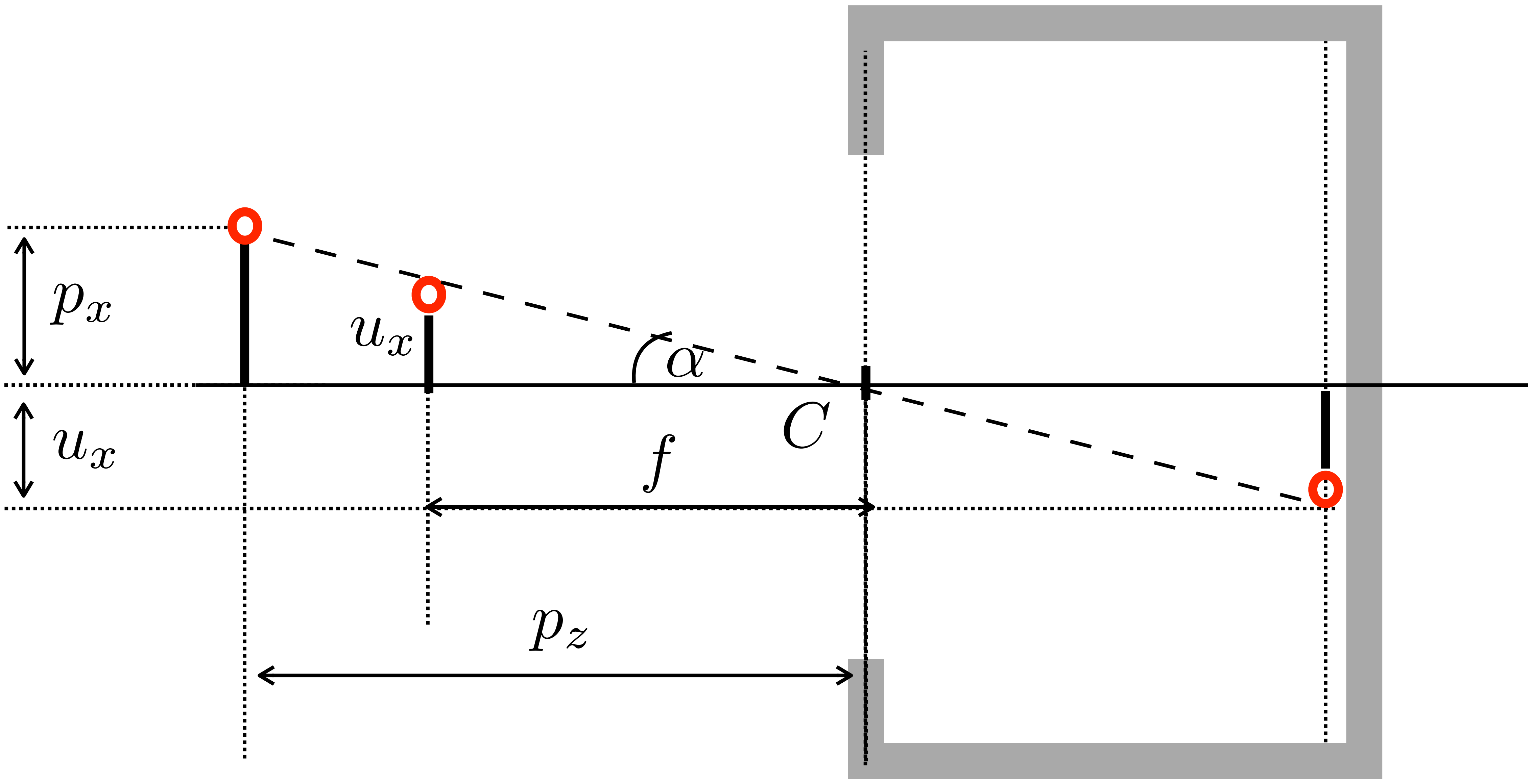


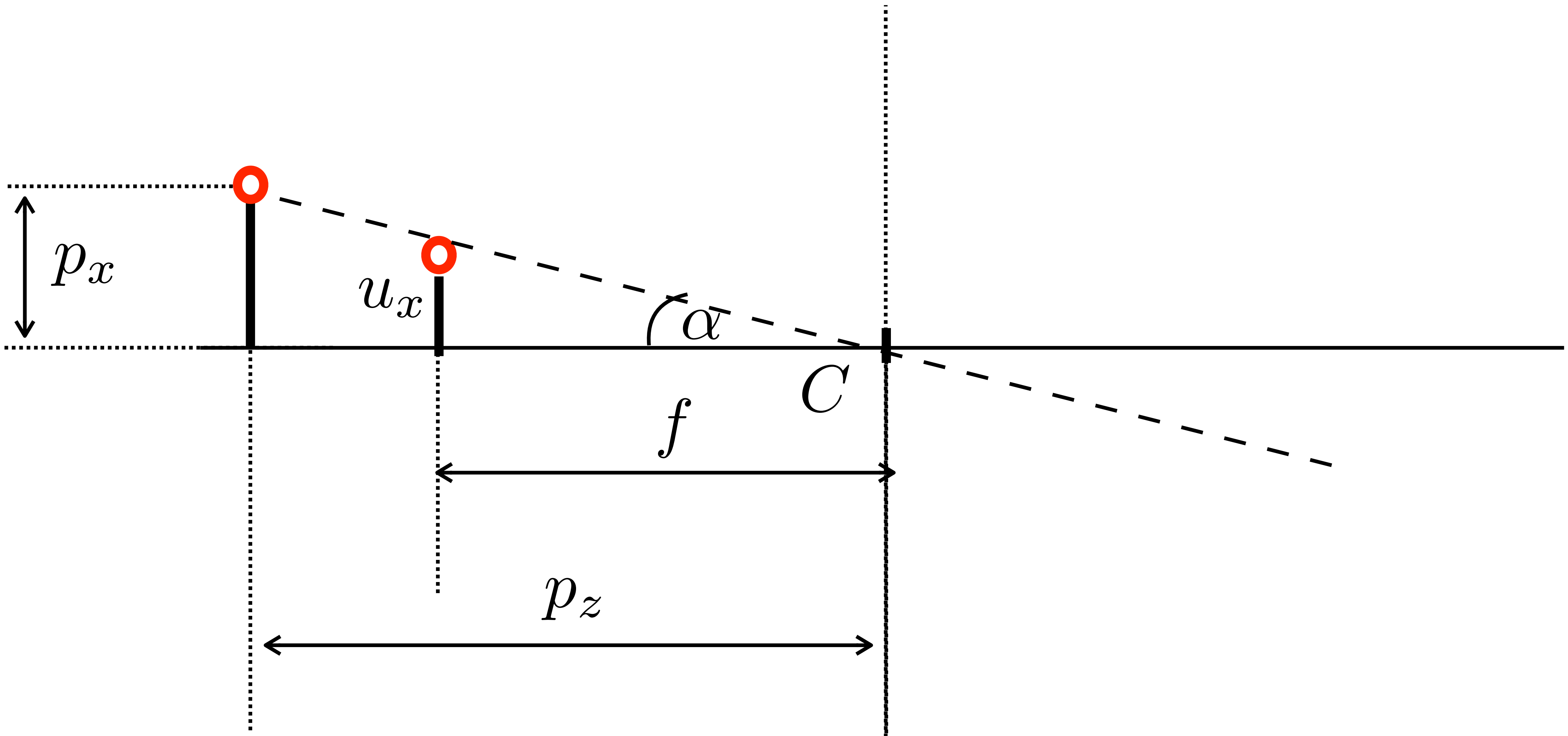
Projection of 3D point in \camera on image plane



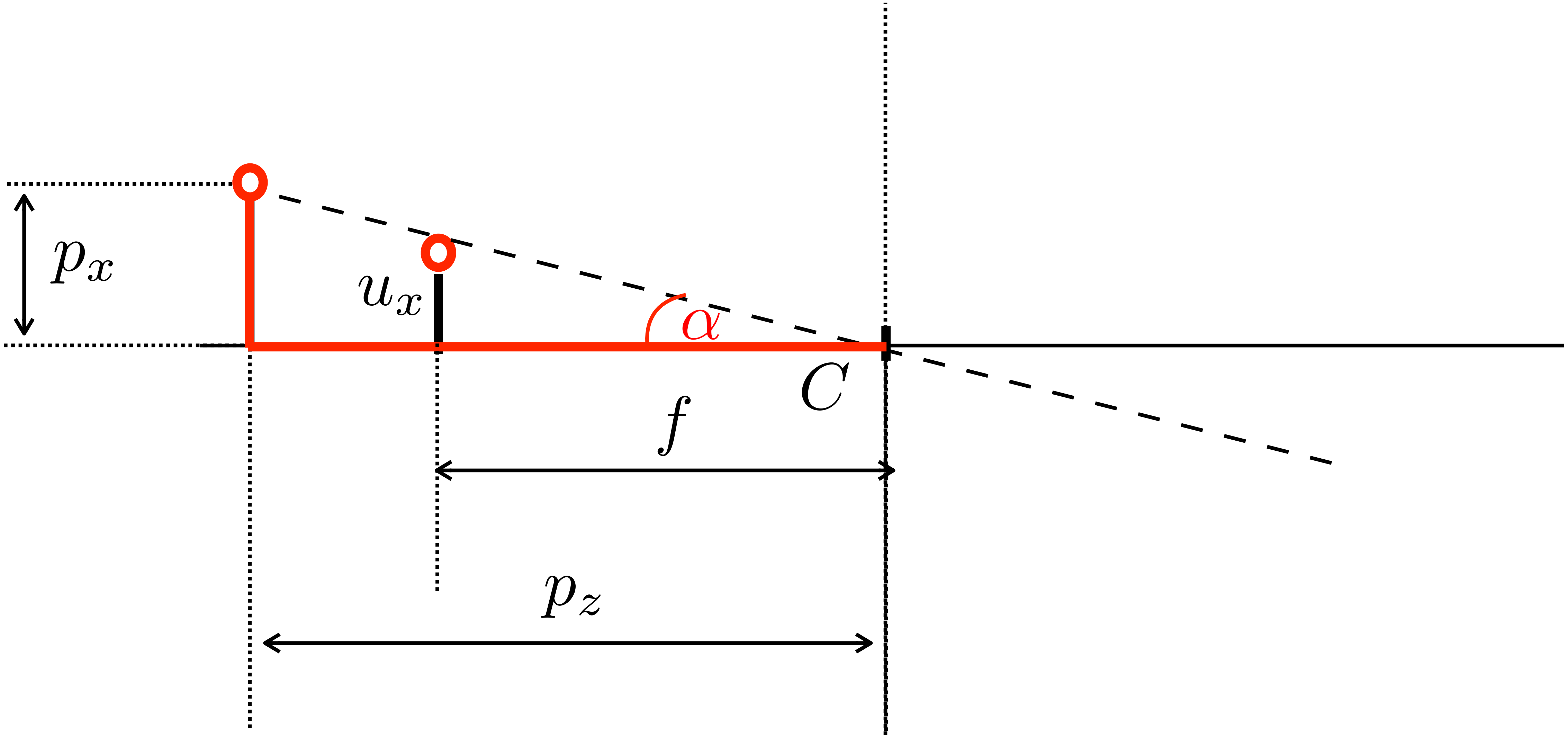
Projection of 3D point in \camera on image plane



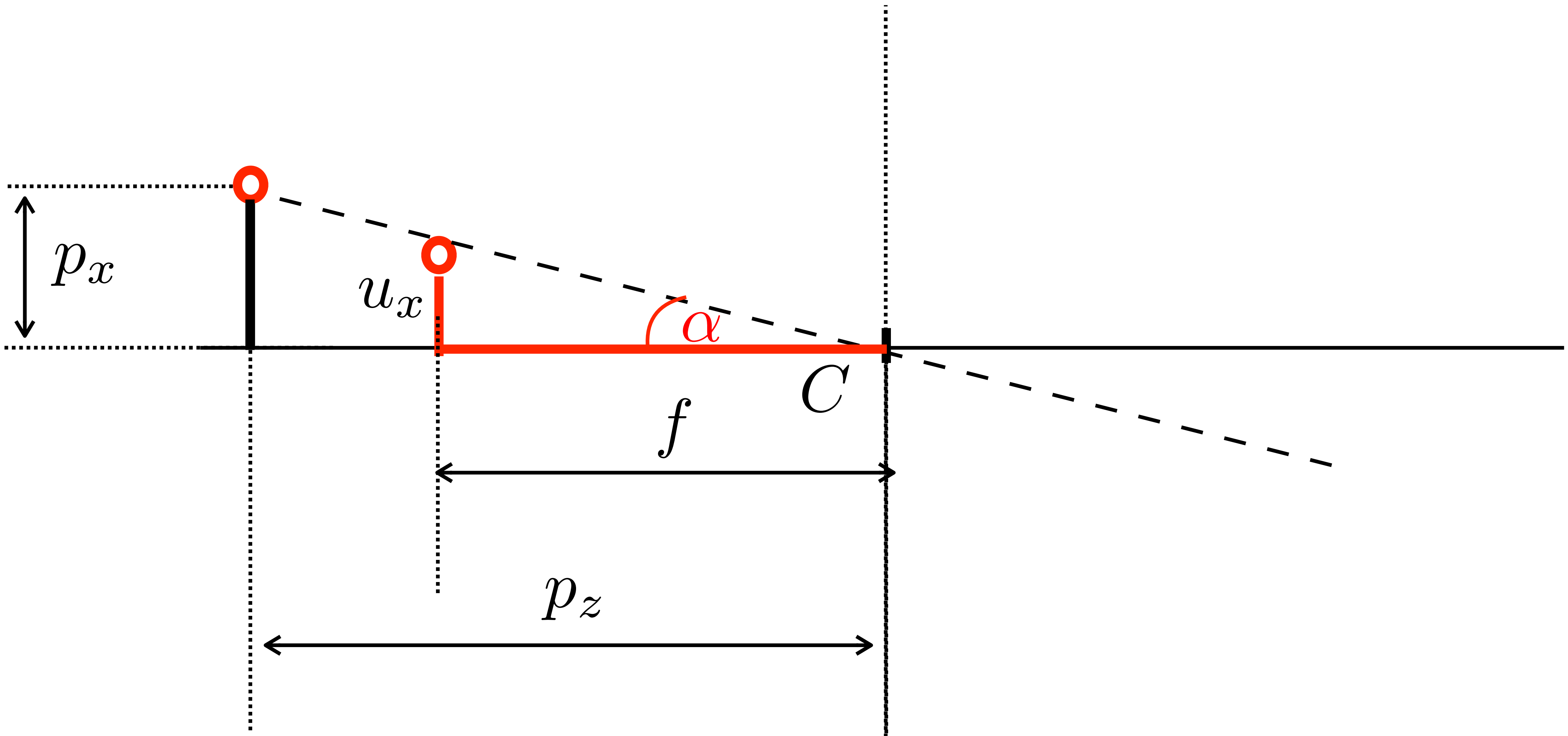

p_x
u_x
u_x
α
C
f
p_z



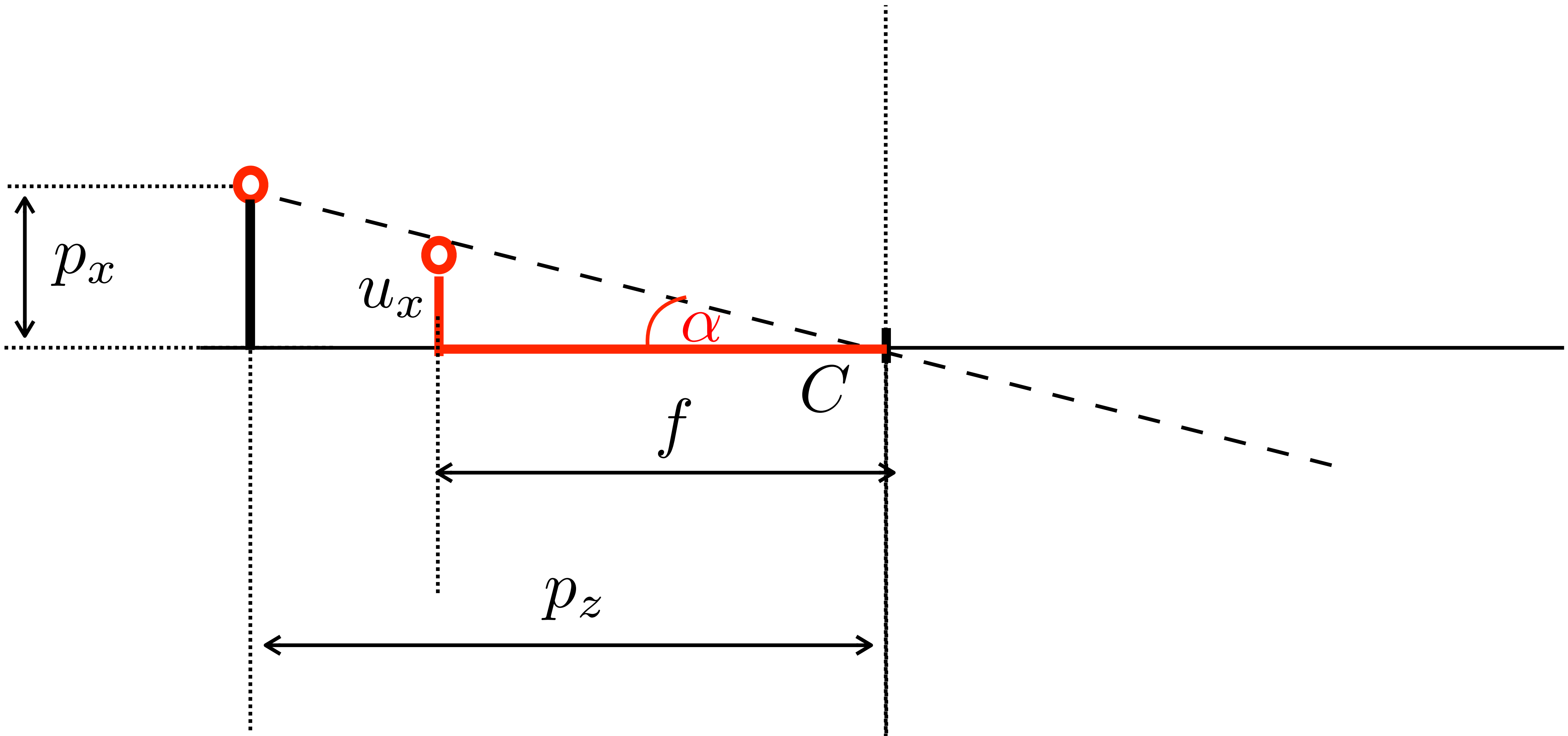
p_x
u_x
α
C
f
p_z



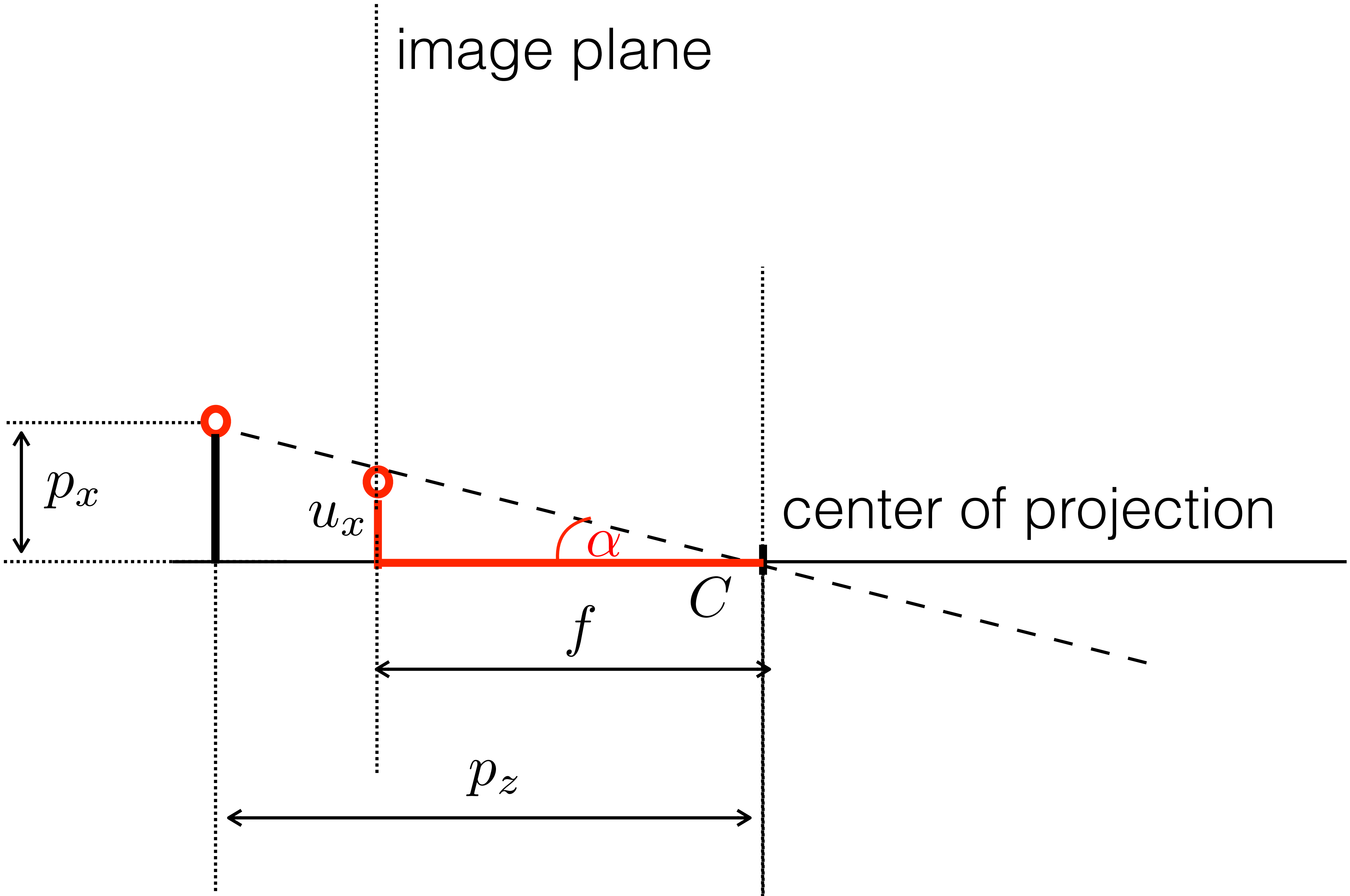
$$\mathrm{tg}(\alpha) = \frac{p_x}{p_z}$$



$$\mathrm{tg}(\alpha) = \frac{p_x}{p_z} = \boxed{\frac{u_x}{f}}$$



$$\mathrm{tg}(\alpha) = \frac{p_x}{p_z} = \frac{u_x}{f} \qquad \Rightarrow \qquad u_x = f\frac{p_x}{p_z}$$

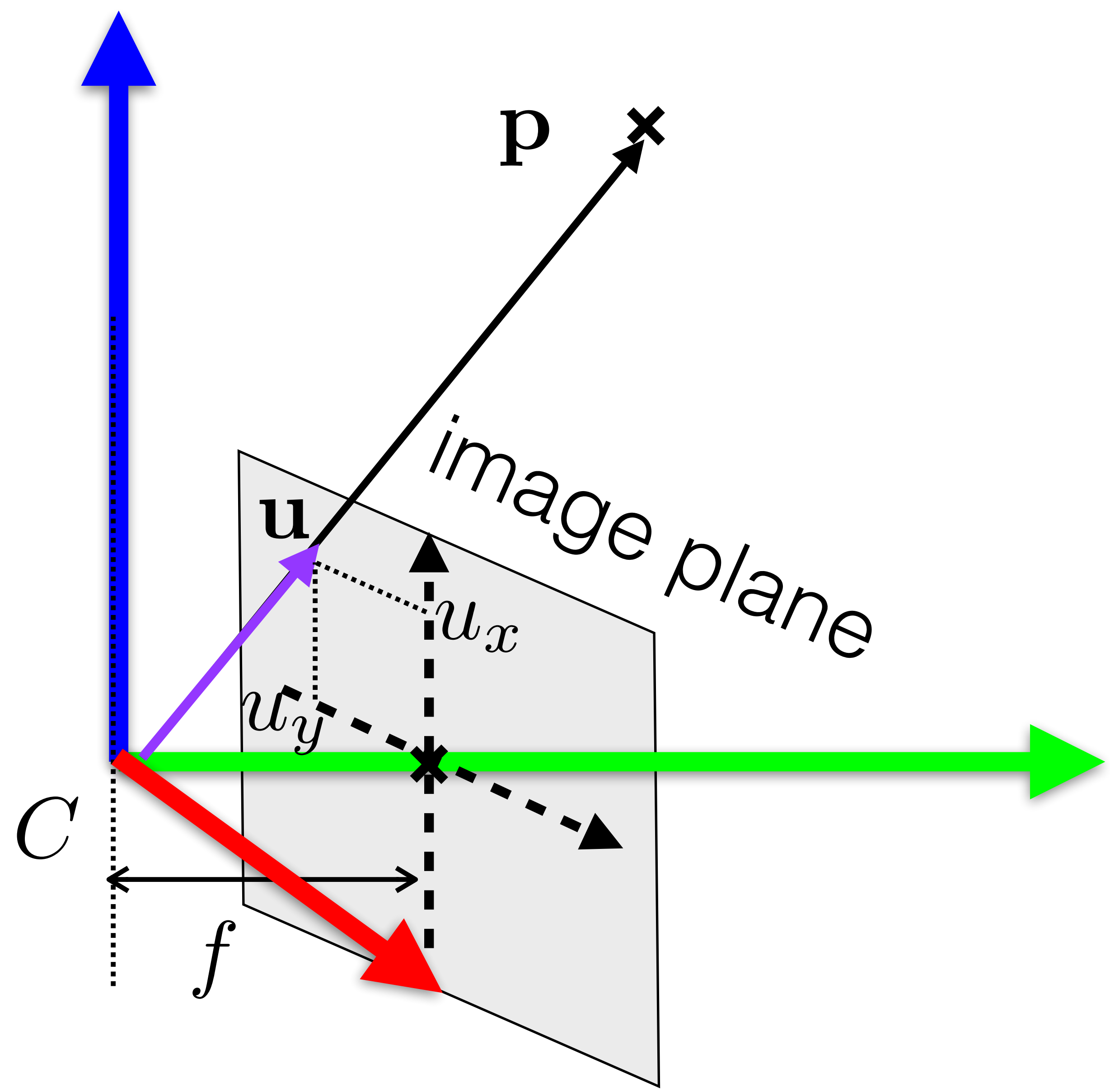


$$\mathrm{tg}(\alpha) = \frac{p_x}{p_z} = \frac{u_x}{f} \quad \Rightarrow \quad u_x = f\frac{p_x}{p_z}$$

Projection 3D points on the image plane

$$u_x = f\frac{p_x}{p_z}$$

$$u_y = f\frac{p_y}{p_z}$$



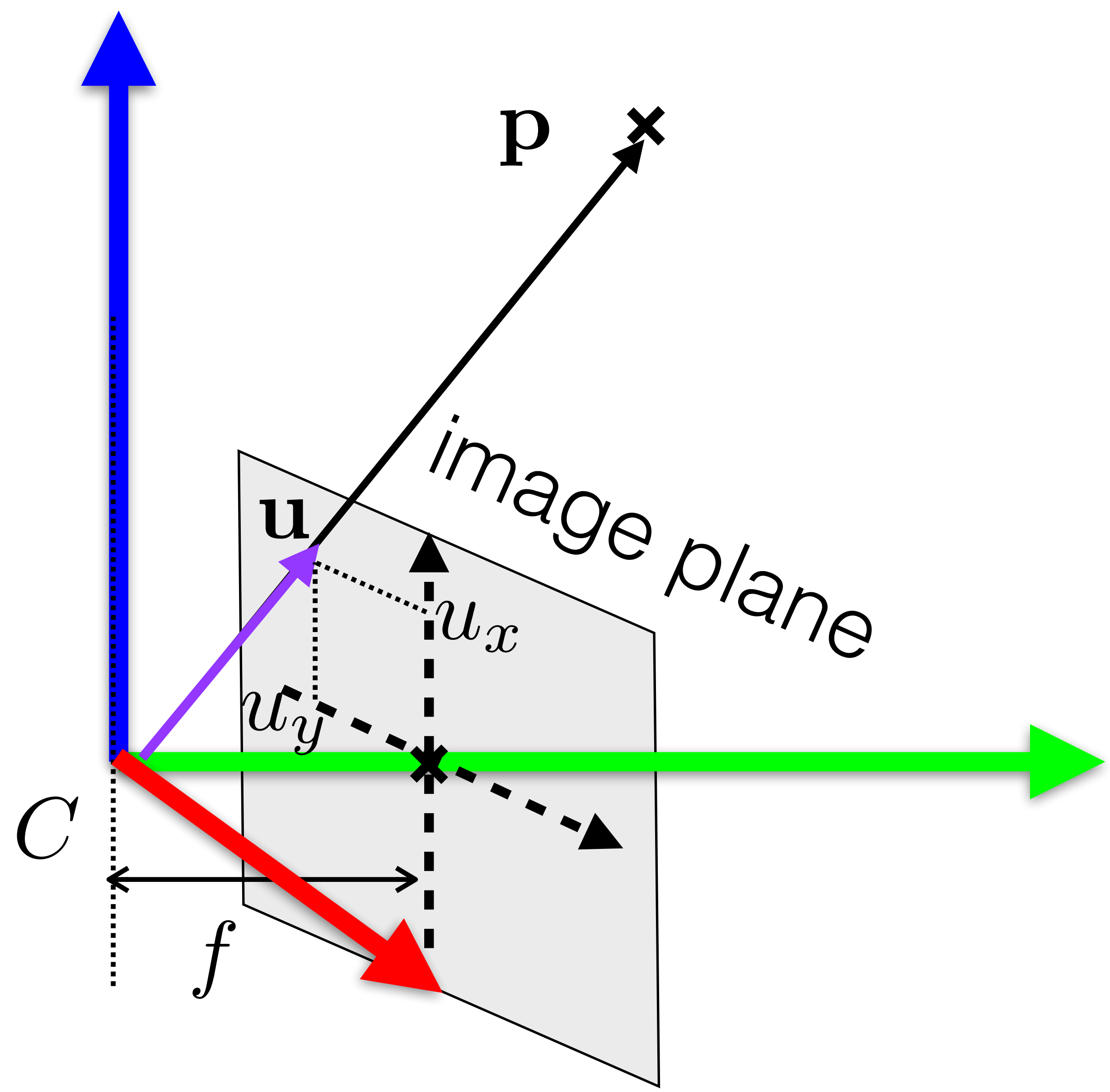
Projection 3D points on the image plane

$$u_x = f\frac{p_x}{p_z}$$

$$u_y = f\frac{p_y}{p_z}$$

$$\Rightarrow$$

$$\lambda u_x = f p_x$$
$$\lambda u_y = f p_y$$
$$\lambda = p_z$$



Projection 3D points on the image plane

$$u_x = f\frac{p_x}{p_z}$$

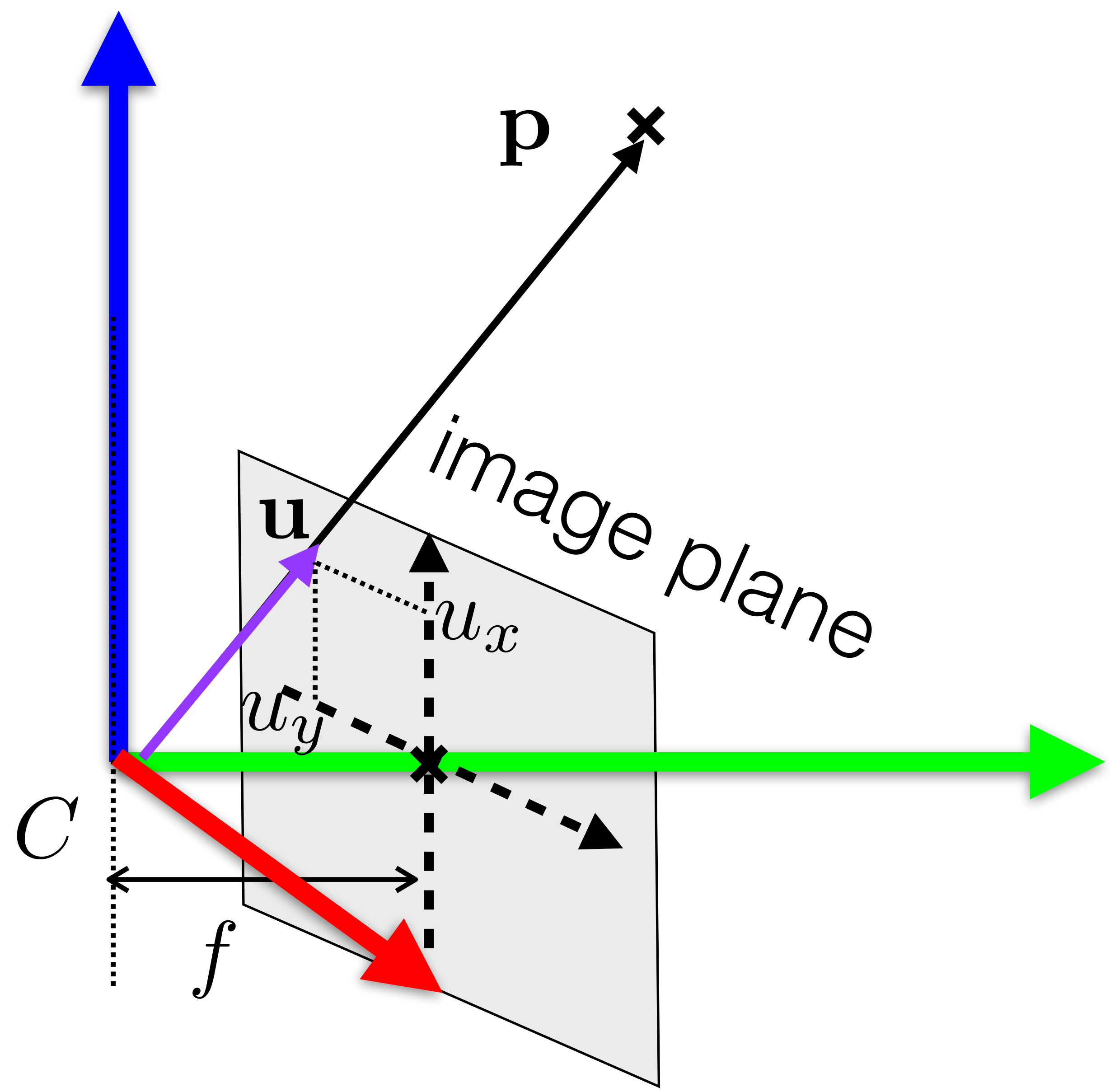
$$u_y = f\frac{p_y}{p_z}$$

$$\Rightarrow$$

$$\lambda u_x = f p_x$$
$$\lambda u_y = f p_y$$
$$\lambda = p_z$$

$$\Rightarrow$$

$$\lambda \begin{bmatrix} u_x \\ u_y \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}$$

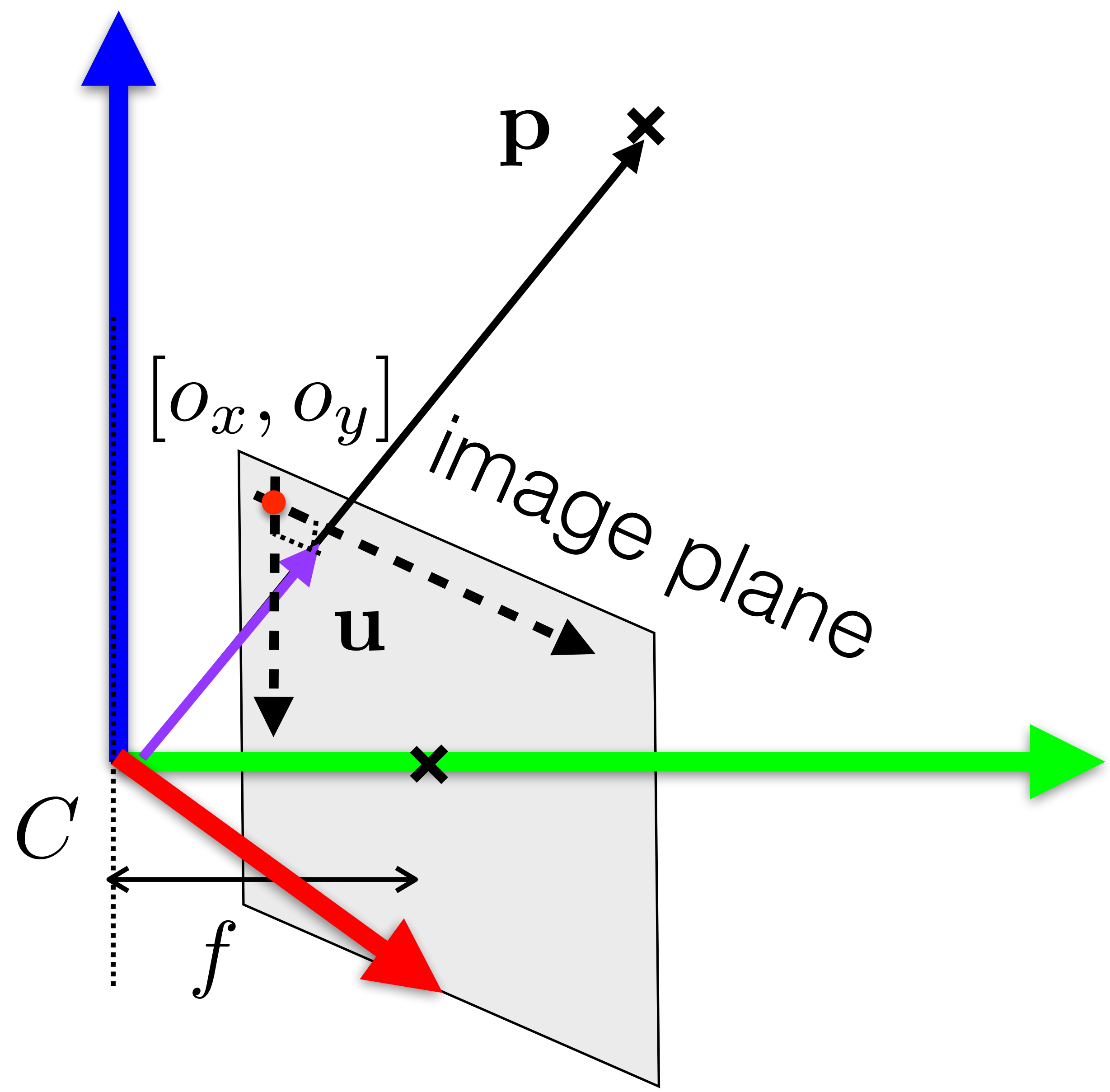


Projection 3D points on the image plane

$$u_x = o_x + s_x f \frac{p_x}{p_z}$$

$$u_y = o_y + s_y f \frac{p_y}{p_z}$$

$$\Rightarrow \quad \lambda \begin{bmatrix} u_x \\ u_y \\ 1 \end{bmatrix} = \begin{bmatrix} s_x f & 0 & o_x \\ 0 & s_y f & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}$$



Projection 3D points on the image plane

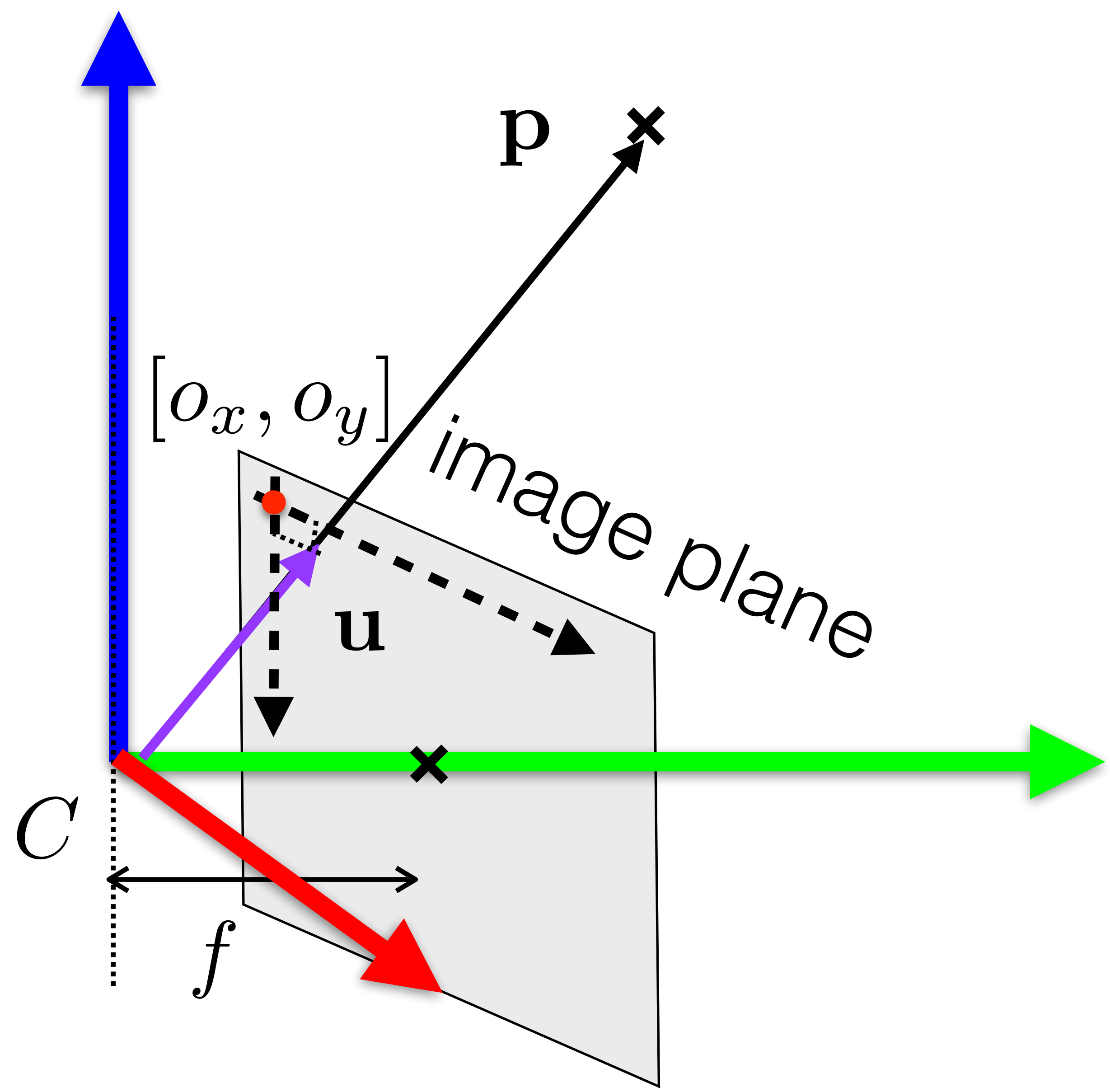
$$u_x = o_x + s_x f \frac{p_x}{p_z}$$

$$u_y = o_y + s_y f \frac{p_y}{p_z}$$

$$\Rightarrow \quad \lambda \begin{bmatrix} u_x \\ u_y \\ 1 \end{bmatrix} = \begin{bmatrix} s_x f & 0 & o_x \\ 0 & s_y f & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}$$

$\mathbf{K} \in \mathcal{R}^{3\times 3}$

upper-triangular, regular matrix with intrinsic parameters of the camera

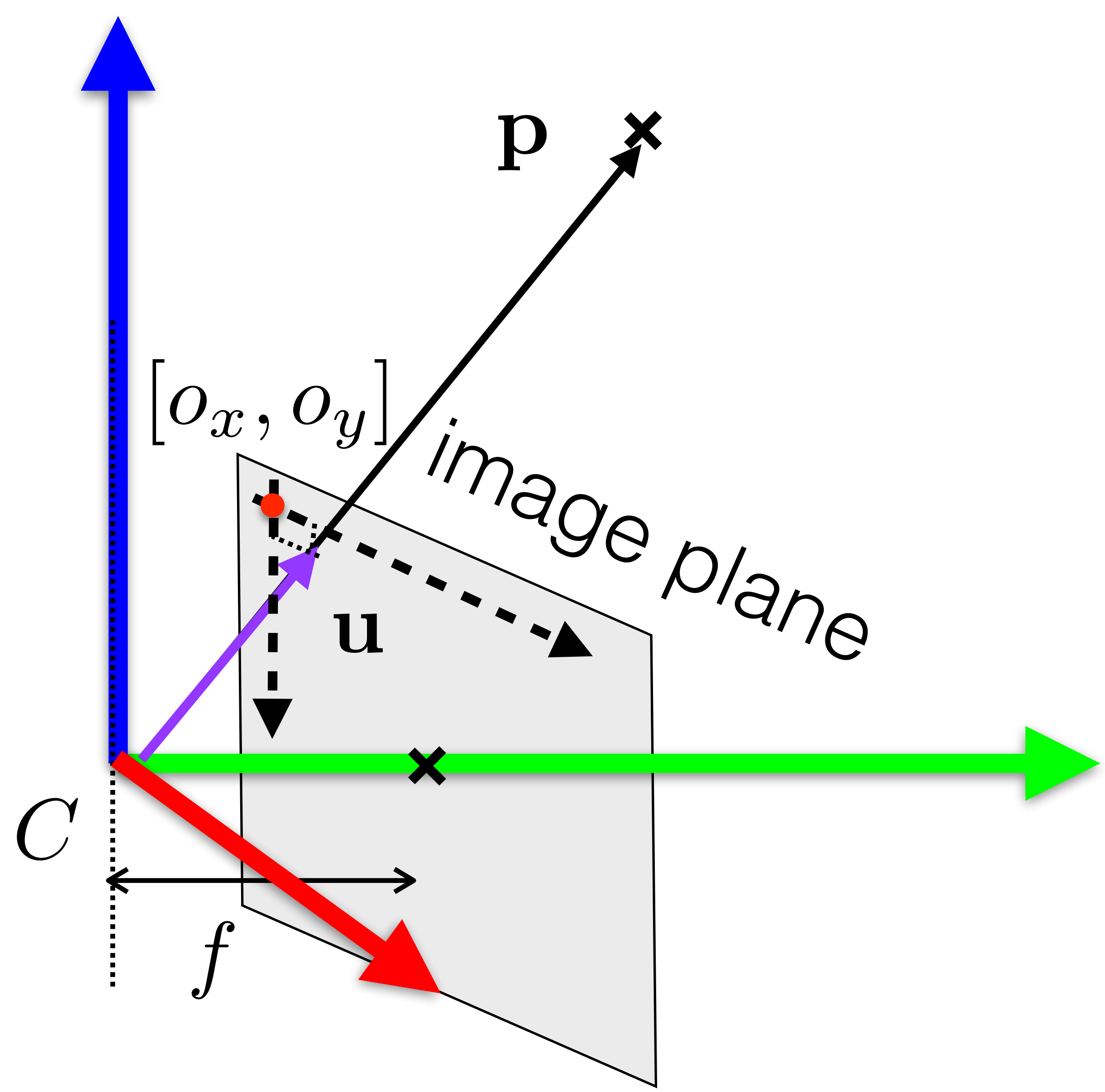


Projection 3D points on the image plane

$$u_x = o_x + s_x f \frac{p_x}{p_z}$$

$$u_y = o_y + s_y f \frac{p_y}{p_z}$$

$$\Rightarrow \quad \lambda \begin{bmatrix} u_x \\ u_y \\ 1 \end{bmatrix} = \begin{bmatrix} s_x f & 0 & o_x \\ 0 & s_y f & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}$$



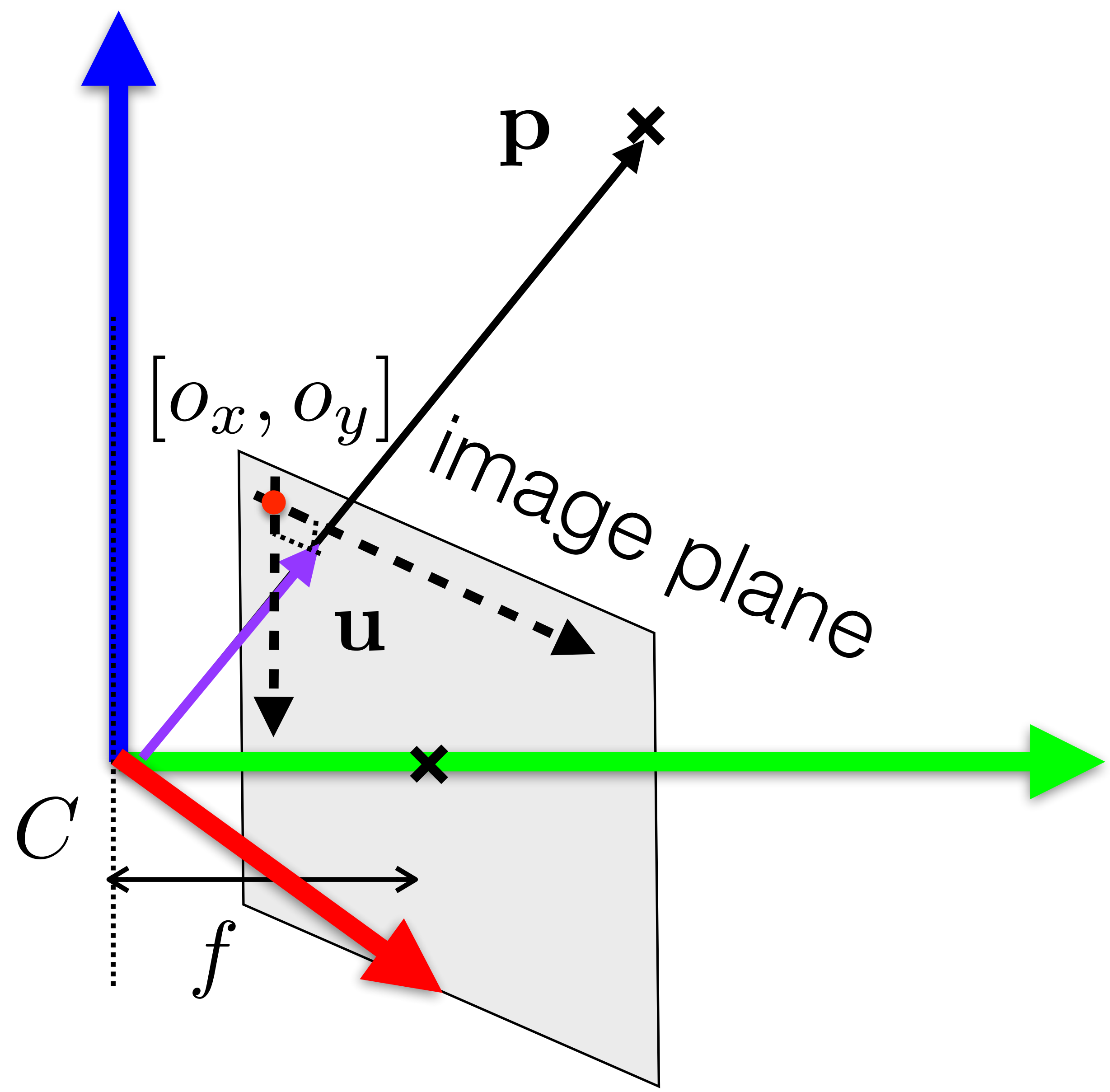
$\mathbf{K} \in \mathcal{R}^{3\times 3}$
upper-triangular, regular matrix with intrinsic parameters of the camera

$$\lambda \bar{\mathbf{u}} = \mathbf{K}\mathbf{p}$$

Projection 3D points on the image plane

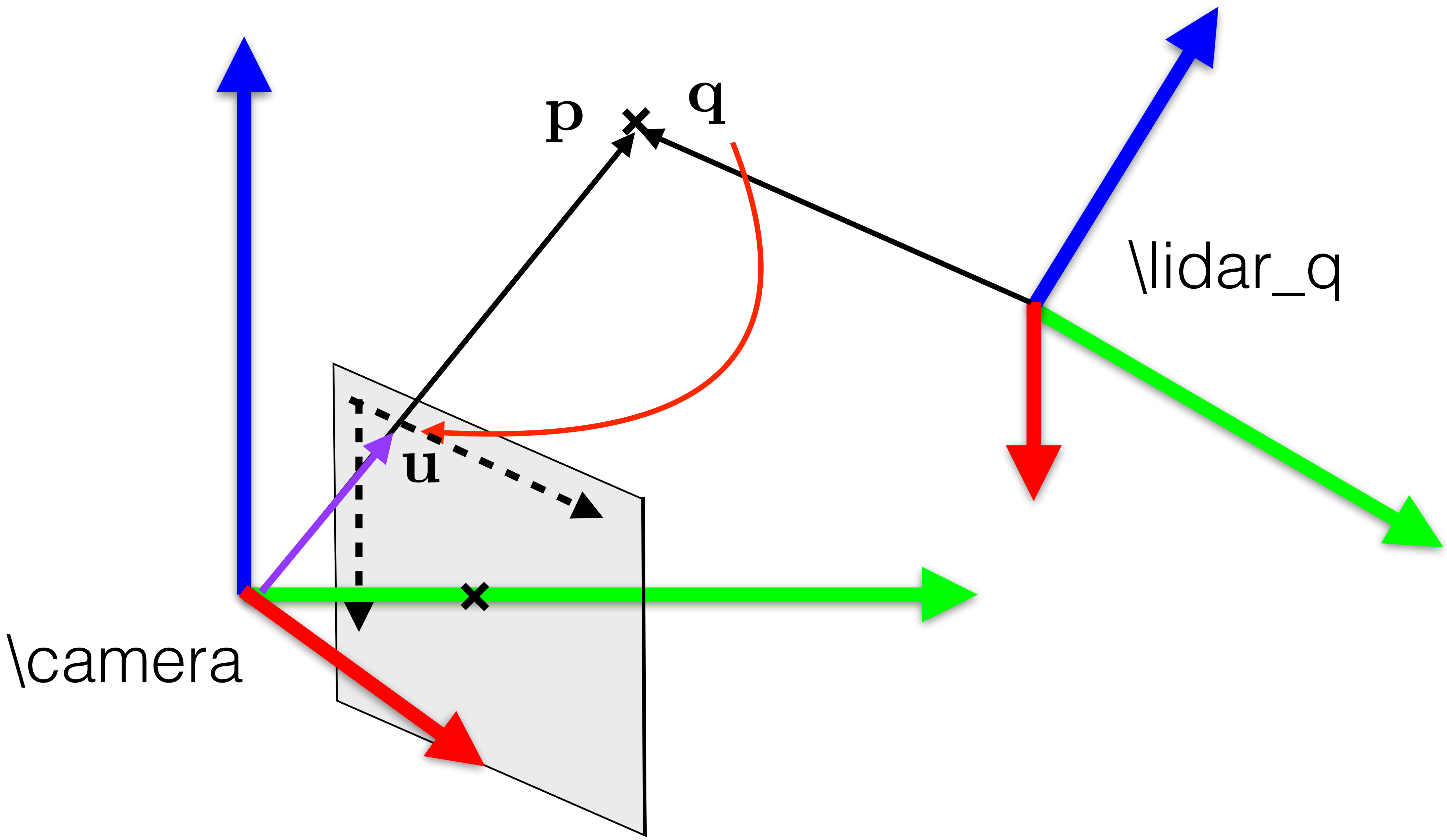
Applications:

- 3D->2D projecting 3D PCL on image plane (colorizing)
- 2D->3D raycasting (projecting detections into 3D map)
- RGBD->3D PCL
- field-of-view, focal length and spatial resolution



$$\lambda \bar{\mathbf{u}} = \mathbf{K}\mathbf{p}$$

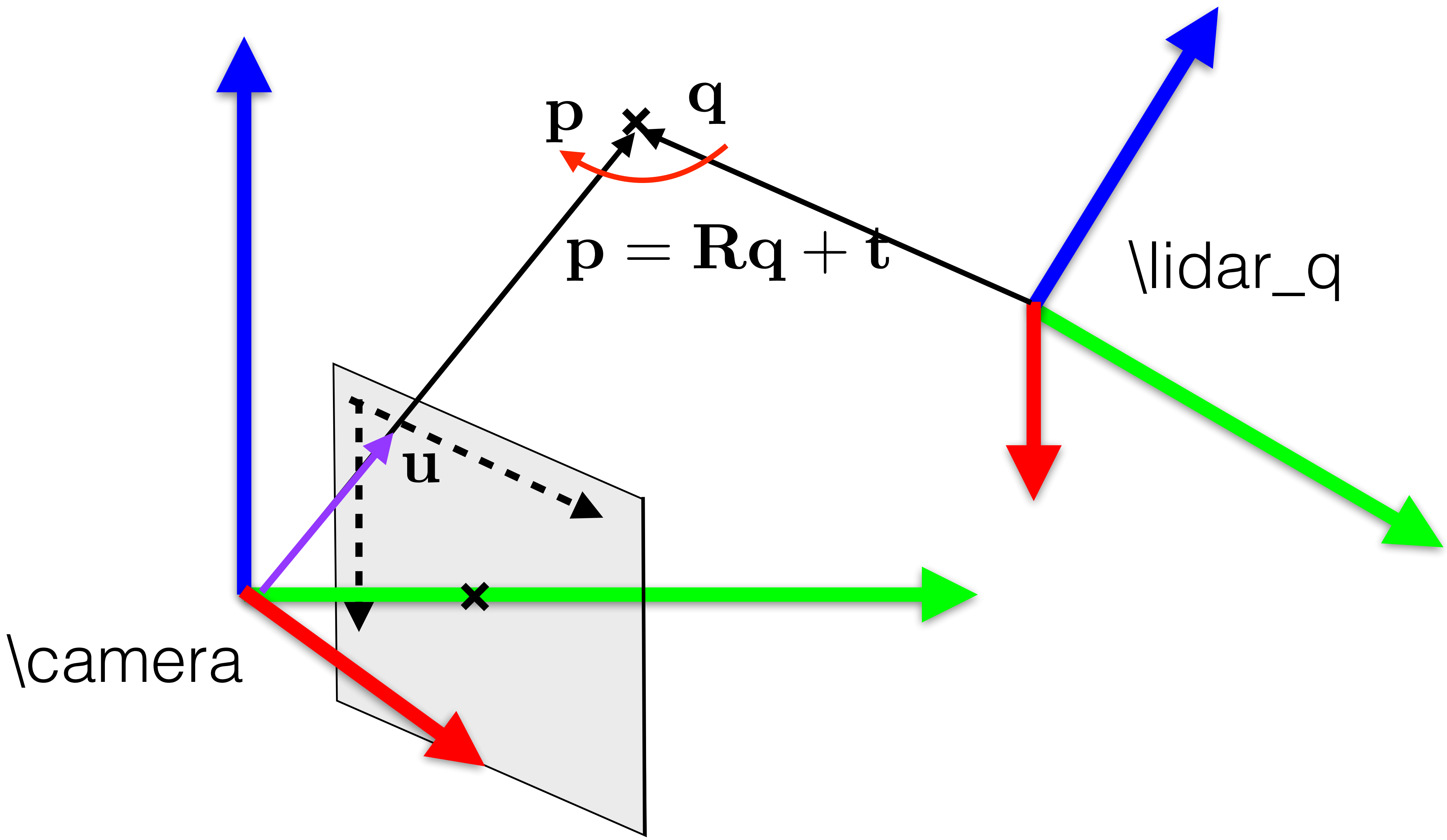
Let us have one lidar and one camera



Camera

Projection from lidar to image plane consists of two steps:

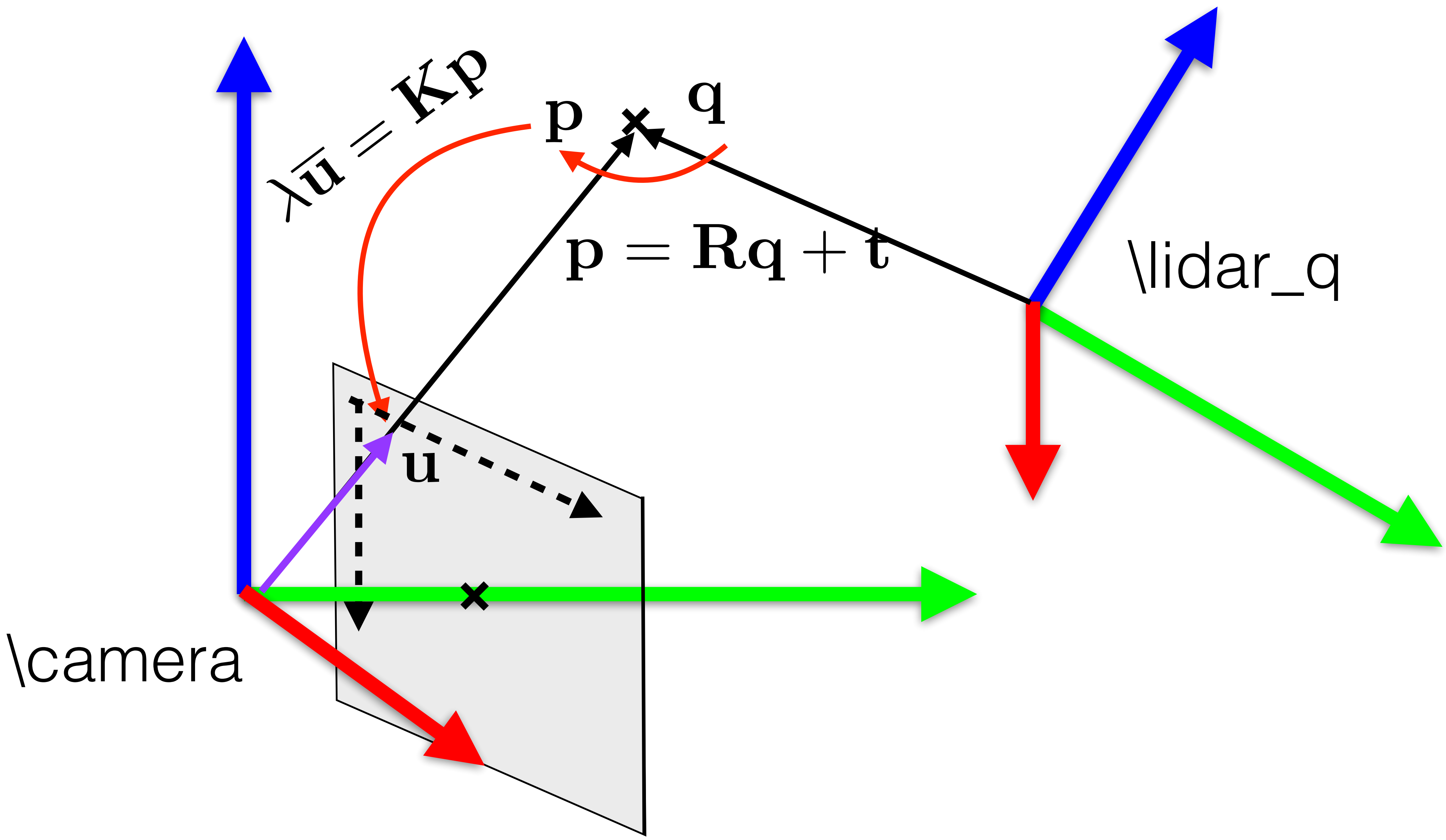
- transform of 3D point in \lidar_q $\mathbf{q} \in \mathcal{R}^3$ to \camera $\mathbf{p} \in \mathcal{R}^3$



Camera

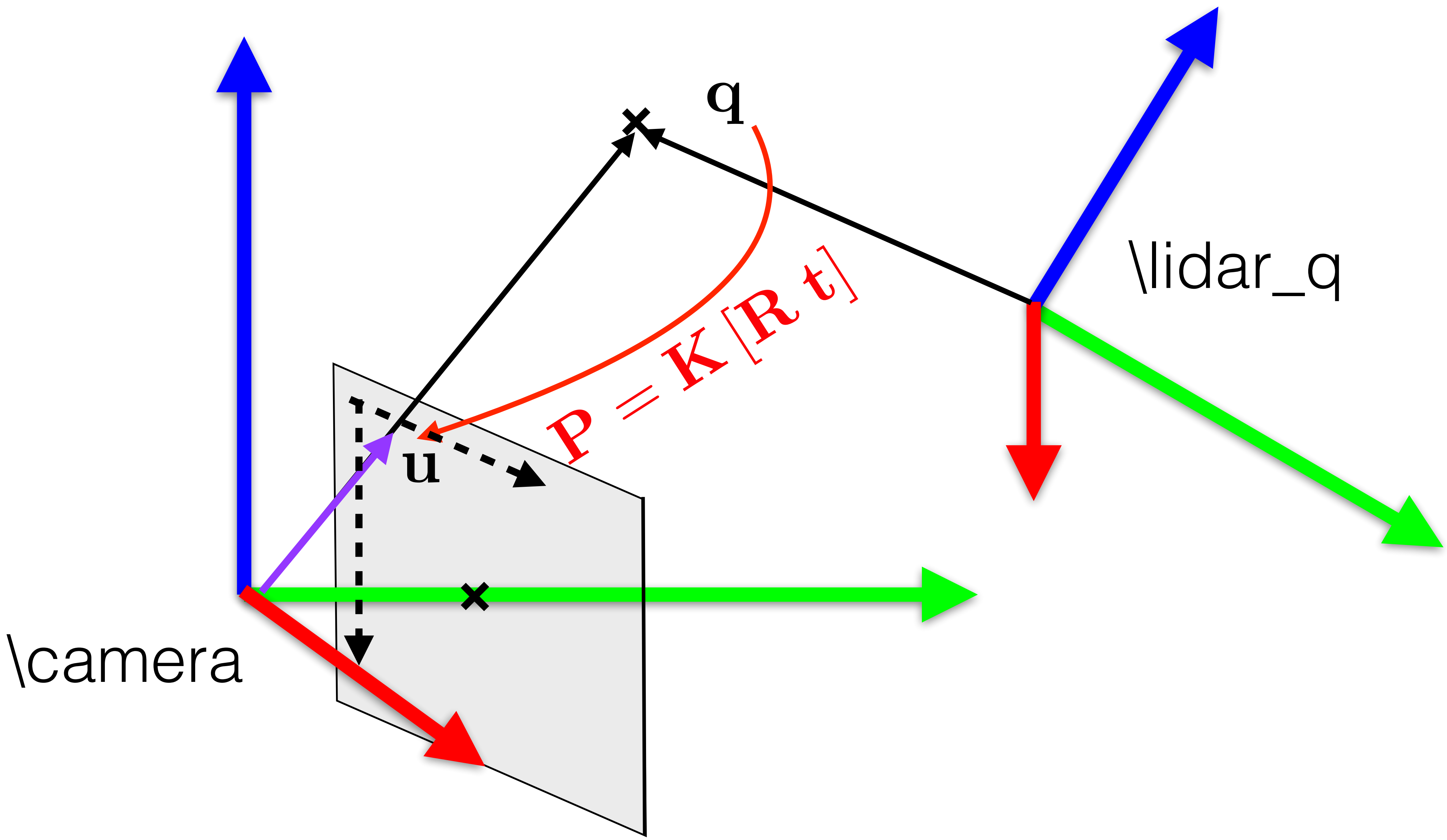
Projection from lidar to image plane consists of two steps:

- transform of 3D point in \lidar_q $\mathbf{q} \in \mathcal{R}^3$ to \camera $\mathbf{p} \in \mathcal{R}^3$
- projection of 3D point in \camera on image plane $\mathbf{u} \in \mathcal{R}^2$



\camera to \lidar_q calibration

$$\lambda \overline{\mathbf{u}} = \underbrace{\mathbf{K} \begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix}}_{\mathbf{P}} \overline{\mathbf{q}}$$



Projection 3D points on the image plane

$$\lambda \overline{\mathbf{u}} = \begin{bmatrix} s_x & s_o & o_x \\ 0 & s_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix} \overline{\mathbf{q}}$$

$\mathbf{K} \in \mathcal{R}^{3\times3}$

$\mathbf{P} \in \mathcal{R}^{3\times4}$

$\mathbf{K} \in \mathcal{R}^{3\times3}$ .... intrinsic parameters

$\mathbf{R} \in \mathcal{SO}(3), \mathbf{t} \in \mathcal{R}^3$...... extrinsic parameters

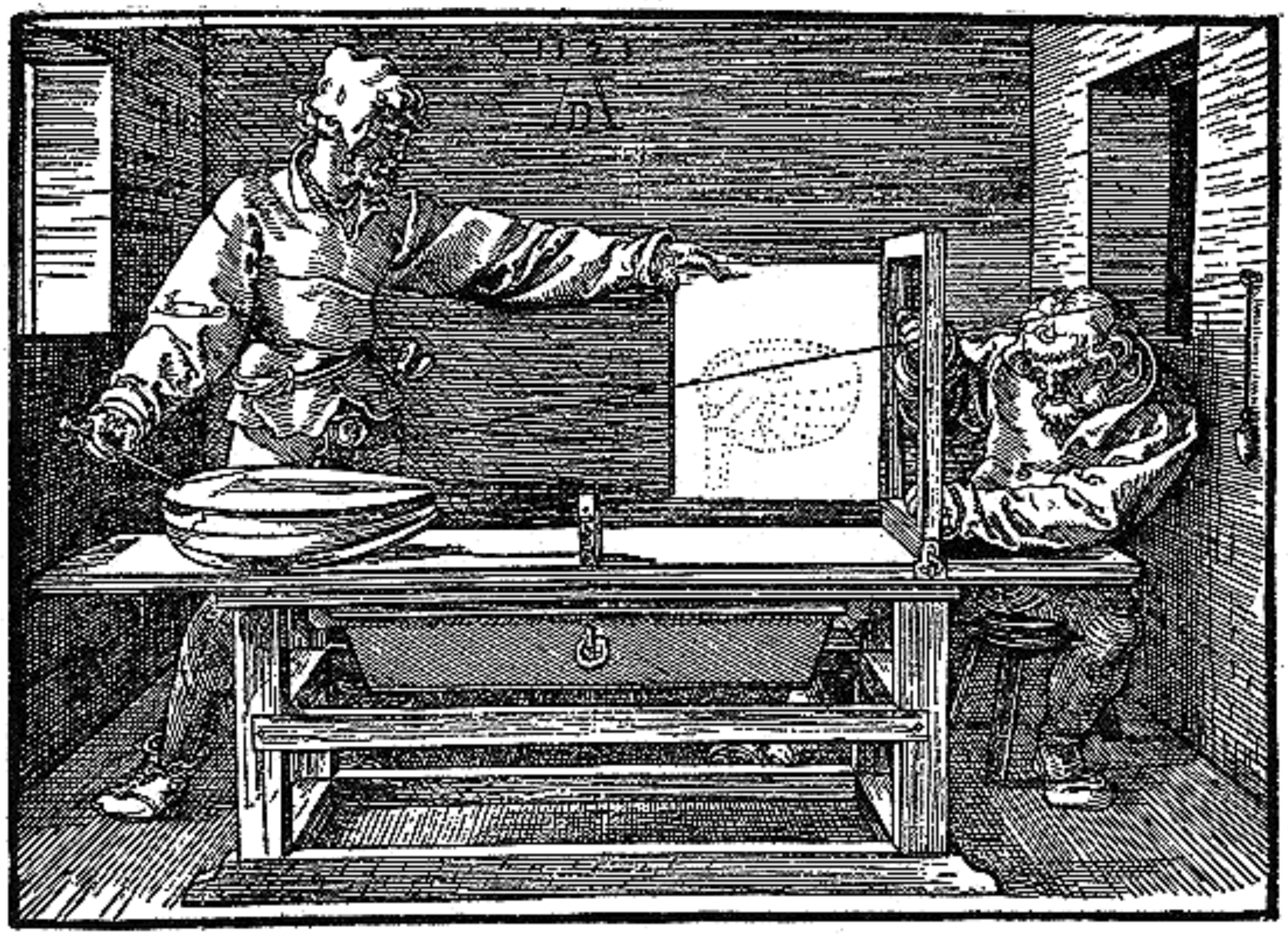
$\mathbf{P} \in \mathcal{R}^{3\times4}$ .... camera projection matrix

Example 1: Project point to a given camera.

Example 2: What is a ray of a pixel?

Example 3: Depth to 3D point-cloud?

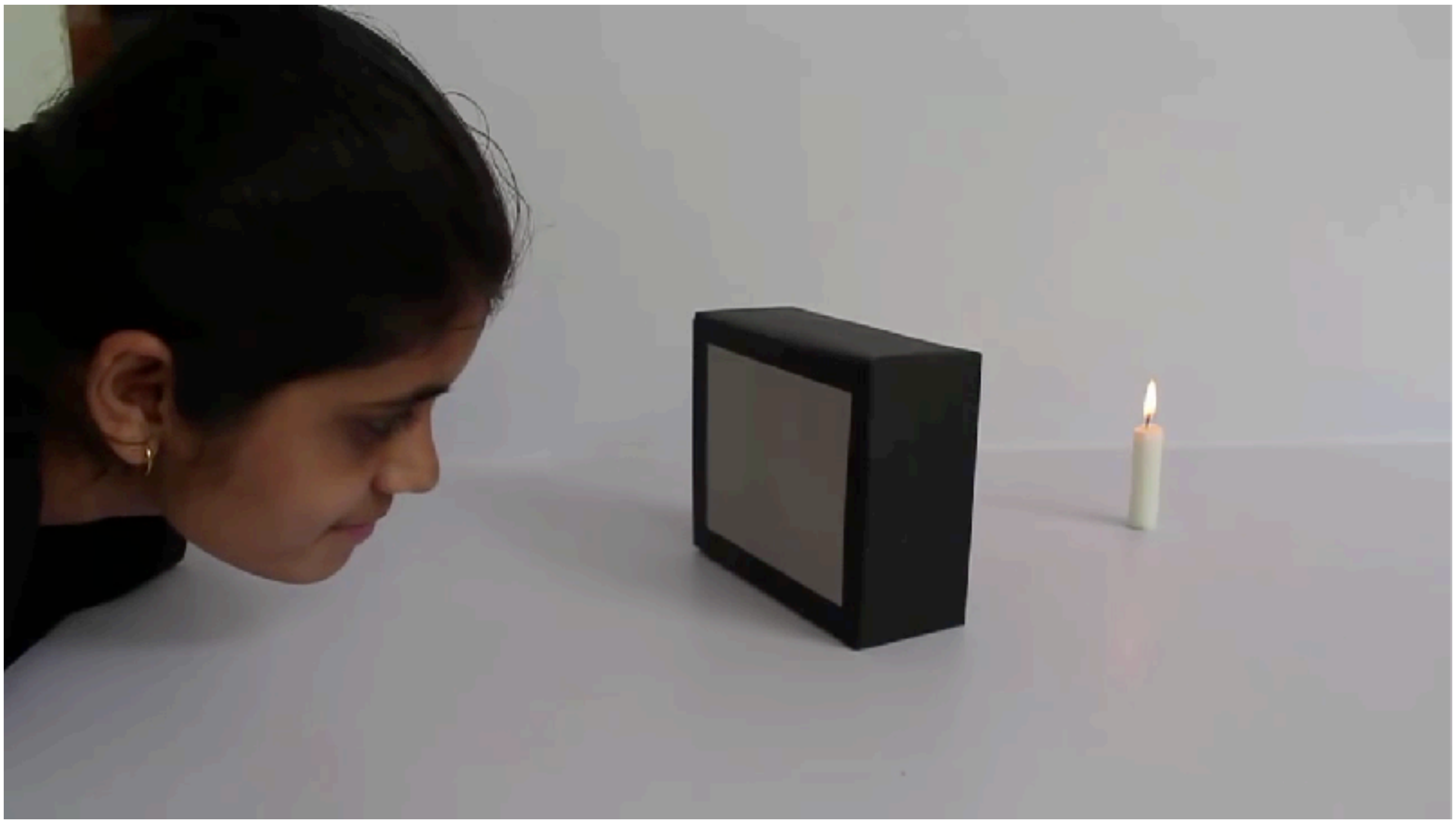
Projection of 3D point in \camera on image plane



Albrecht Durer (1545), Hitachi Viewmuseum

Projection of 3D point in \camera on image plane

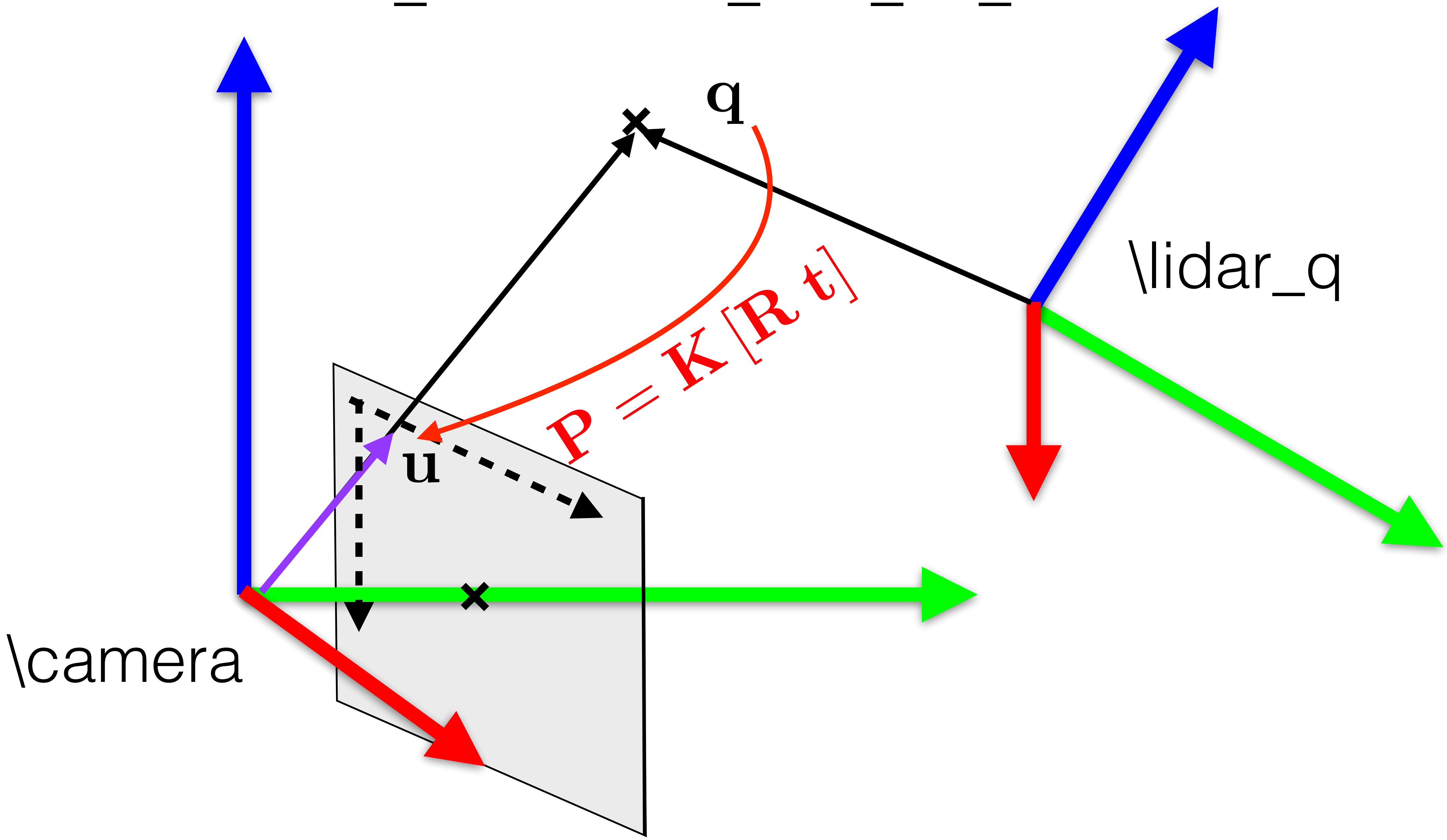
Pinhole camera model



\camera to \lidar_q calibration

$$\lambda \overline{\mathbf{u}} = \underbrace{\mathbf{K}\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}}_{\mathbf{P}} \overline{\mathbf{q}}$$

openCV: https://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html

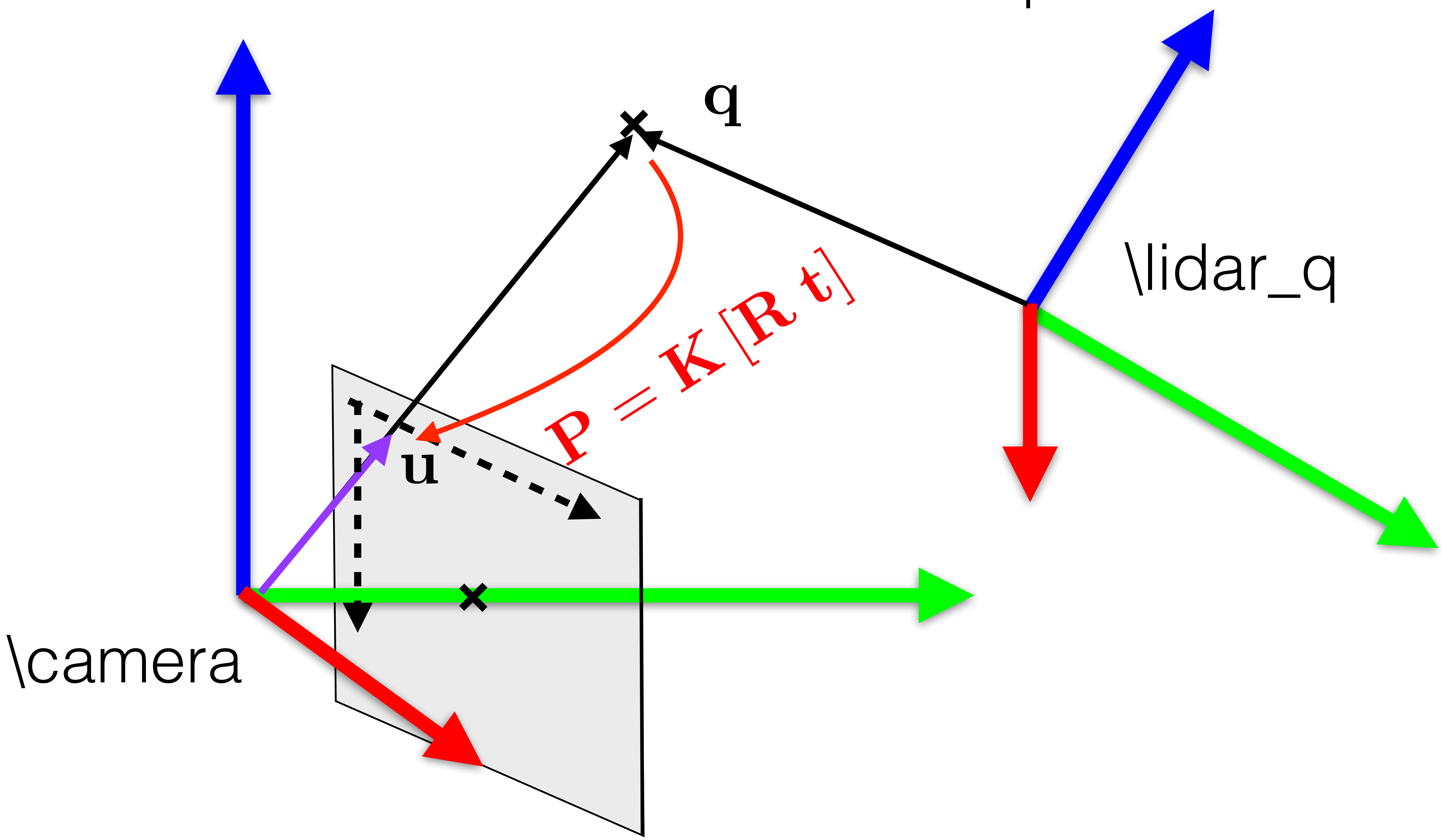


\camera to \lidar_q calibration

$$\lambda \overline{\mathbf{u}} = \mathbf{P} \overline{\mathbf{q}}$$

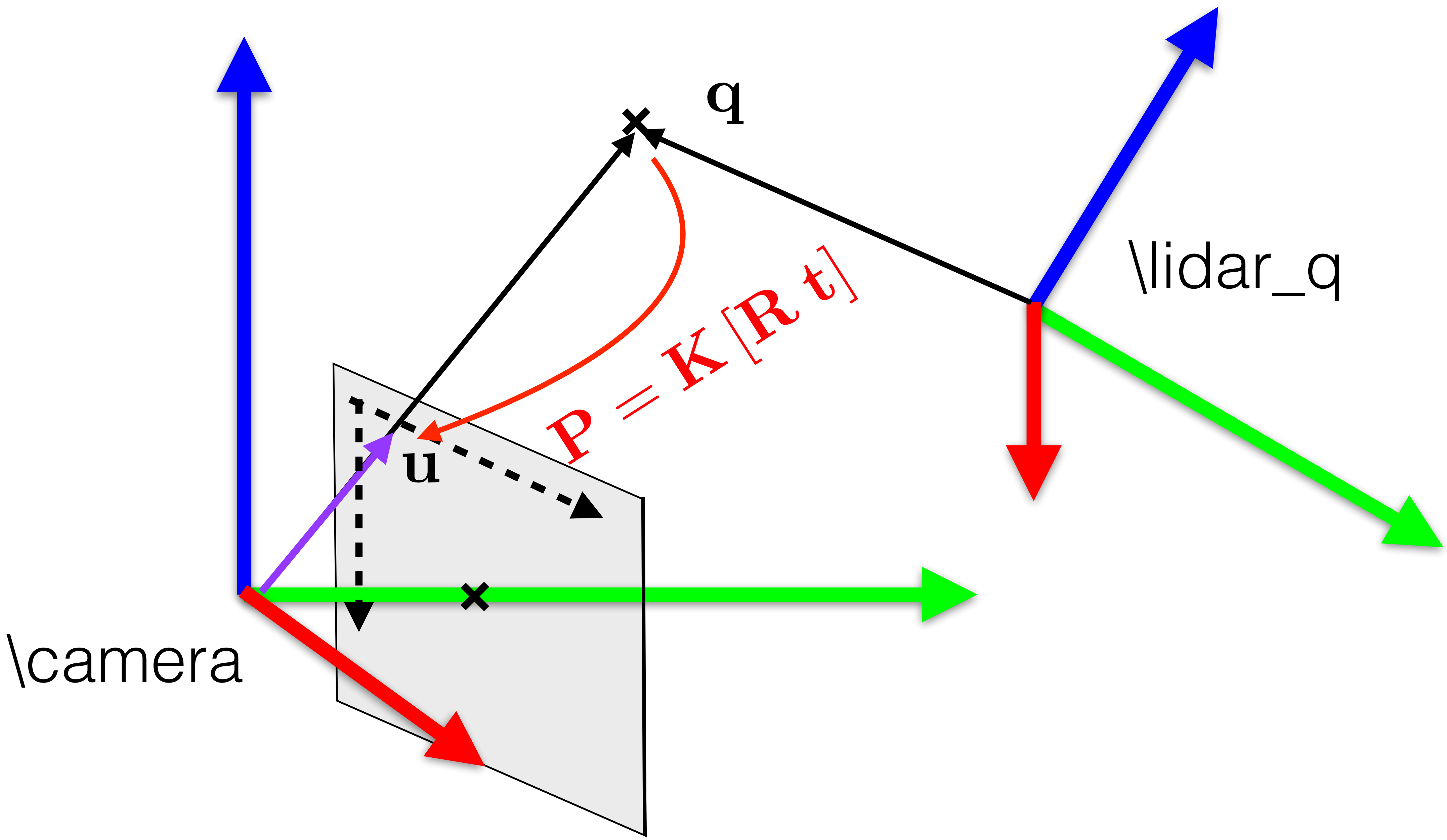
unknown

calibration from 2D-3D correspondences



\camera to \lidar_q calibration

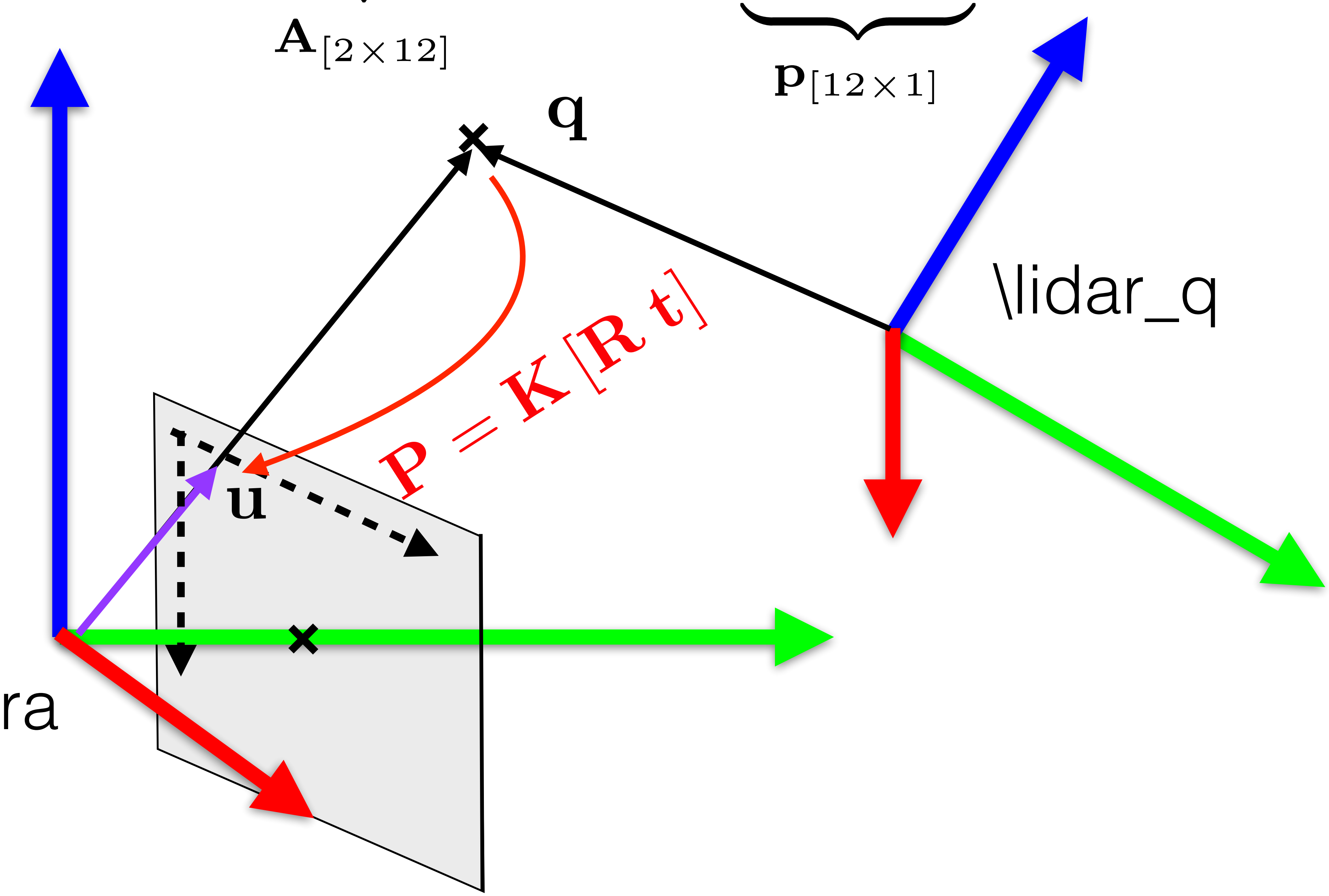
$$\lambda \overline{\mathbf{u}} = \mathbf{P}\overline{\mathbf{q}} \quad \Rightarrow \quad \underbrace{\begin{bmatrix} -\overline{\mathbf{q}}^\top & \mathbf{0}^\top & u_x\overline{\mathbf{q}}^\top \\ \mathbf{0}^\top & -\overline{\mathbf{q}}^\top & u_y\overline{\mathbf{q}}^\top \end{bmatrix}}_{\mathbf{A}_{[2\times 12]}} \underbrace{\begin{bmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{bmatrix}}_{\mathbf{p}_{[12\times 1]}} = \mathbf{0}_{[2\times 1]}$$



\camera to \lidar_q calibration

Each 2D-3D correspondence yields two equations:

$$\underbrace{\begin{bmatrix} -\bar{\mathbf{q}}_i^\top & \mathbf{0}^\top & u_{xi}\bar{\mathbf{q}}_i^\top \\ \mathbf{0}^\top & -\bar{\mathbf{q}}_i^\top & u_{yi}\bar{\mathbf{q}}_i^\top \end{bmatrix}}_{\mathbf{A}_{[2\times 12]}} \underbrace{\begin{bmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{bmatrix}}_{\mathbf{p}_{[12\times 1]}} = \mathbf{0}_{[2\times 1]}$$



\camera to \lidar_q calibration

Each 2D-3D correspondence yields two equations:

$$\underbrace{\begin{bmatrix} -\overline{\mathbf{q}}_i^\top & \mathbf{0}^\top & u_{xi}\overline{\mathbf{q}}_i^\top \\ \mathbf{0}^\top & -\overline{\mathbf{q}}_i^\top & u_{yi}\overline{\mathbf{q}}_i^\top \end{bmatrix}}_{\mathbf{A}_{[2\times 12]}} \underbrace{\begin{bmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{bmatrix}}_{\mathbf{p}_{[12\times 1]}} = \mathbf{0}_{[2\times 1]}$$

For N 2D-3D correspondences, we obtain (2Nx12) homogeneous linear system $\mathbf{A}\mathbf{p} = \mathbf{0}$

Assuming

- i.i.d. measurements and
- gaussian noise between left-hand-side and right-hand-side

$$\mathbf{p}^* = \operatorname{argmin} \|\mathtt{A}\mathbf{p}\|$$

\camera to \lidar_q calibration

Each 2D-3D correspondence yields two equations:

$$\underbrace{\begin{bmatrix} -\overline{\mathbf{q}}_i^\top & \mathbf{0}^\top & u_{xi}\overline{\mathbf{q}}_i^\top \\ \mathbf{0}^\top & -\overline{\mathbf{q}}_i^\top & u_{yi}\overline{\mathbf{q}}_i^\top \end{bmatrix}}_{\mathbf{A}_{[2\times 12]}} \underbrace{\begin{bmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{bmatrix}}_{\mathbf{p}_{[12\times 1]}} = \mathbf{0}_{[2\times 1]}$$

For N 2D-3D correspondences, we obtain (2Nx12) homogeneous linear system $\mathbf{Ap} = \mathbf{0}$

Assuming

- i.i.d. measurements and
- gaussian noise between left-hand-side and right-hand-side

$$\mathbf{p}^* = \text{argmin}\|\mathtt{A}\mathbf{p}\| \ \text{ subject to } \ \|\mathbf{p}\| = 1$$

$$\mathbf{p}^* = \text{argmin}\|\mathtt{A}\mathbf{p}\| \quad \text{subject to} \quad \|\mathbf{p}\| = 1$$

Lagrange function:

$$L(\mathbf{p}, \lambda) = \|\mathtt{A}\mathbf{p}\| + \lambda(1 - \|\mathbf{p}\|) = \mathbf{p}^\top \mathtt{A}^\top \mathtt{A}\mathbf{p} + \lambda(1 - \mathbf{p}^\top \mathbf{p})$$

Critical points:

$$\frac{\partial L(\mathbf{p}, \lambda)}{\partial \mathbf{p}} = 2\mathtt{A}^\top \mathtt{A}\mathbf{p} - 2\lambda\mathbf{p} = \mathbf{0}$$

$$\frac{\partial L(\mathbf{p}, \lambda)}{\partial \lambda} = 1 - \mathbf{p}^\top \mathbf{p} = \mathbf{0}$$

First equation is characteristic equation $(\mathtt{A}^\top \mathtt{A} - \lambda \mathtt{I})\mathbf{p} = \mathbf{0}$

Every eigen-vector of $\mathtt{A}^\top \mathtt{A}$ is the critical point choose one

Cost function in these eigen vectors is equal to eigen-values

$$\|\mathtt{A}\mathbf{p}\| = \mathbf{p}^\top \mathtt{A}^\top \mathtt{A}\mathbf{p} = \mathbf{p}^\top \lambda \mathbf{p} = \lambda \mathbf{p}^\top \mathbf{p} = \lambda\|\mathbf{p}\| = \lambda$$

Solution is the eigen-vector of $\mathtt{A}^\top \mathtt{A}$ with the smallest eigen-value

Summary camera calibration

- Manually estimate 2D-3D correspondences
- Build matrix

$$\mathbf{A} = \begin{bmatrix} -\mathbf{q}^\top & \mathbf{0}^\top & u_x\mathbf{q}^\top \\ \mathbf{0}^\top & -\mathbf{q}^\top & u_y\mathbf{q}^\top \\ . & . & . \\ . & . & . \\ . & . & . \end{bmatrix}$$

- Find eigen-values and eigen-vectors of $\mathbf{A}^\top\mathbf{A}$ (python: `numpy.linalg.eig`)
- Reshape the eigen-vector $\mathbf{p} \in \mathcal{R}^{12\times 1}$ with the smallest eigen-value to camera matrix $\mathbf{P} \in \mathcal{R}^{3\times 4}$
- Scale does not matter: $\mathbf{P} = \mathbf{P}/\|[\mathbf{p}_{31}, \mathbf{p}_{32}, \mathbf{p}_{33}]^\top\|$
- Optionally decompose:

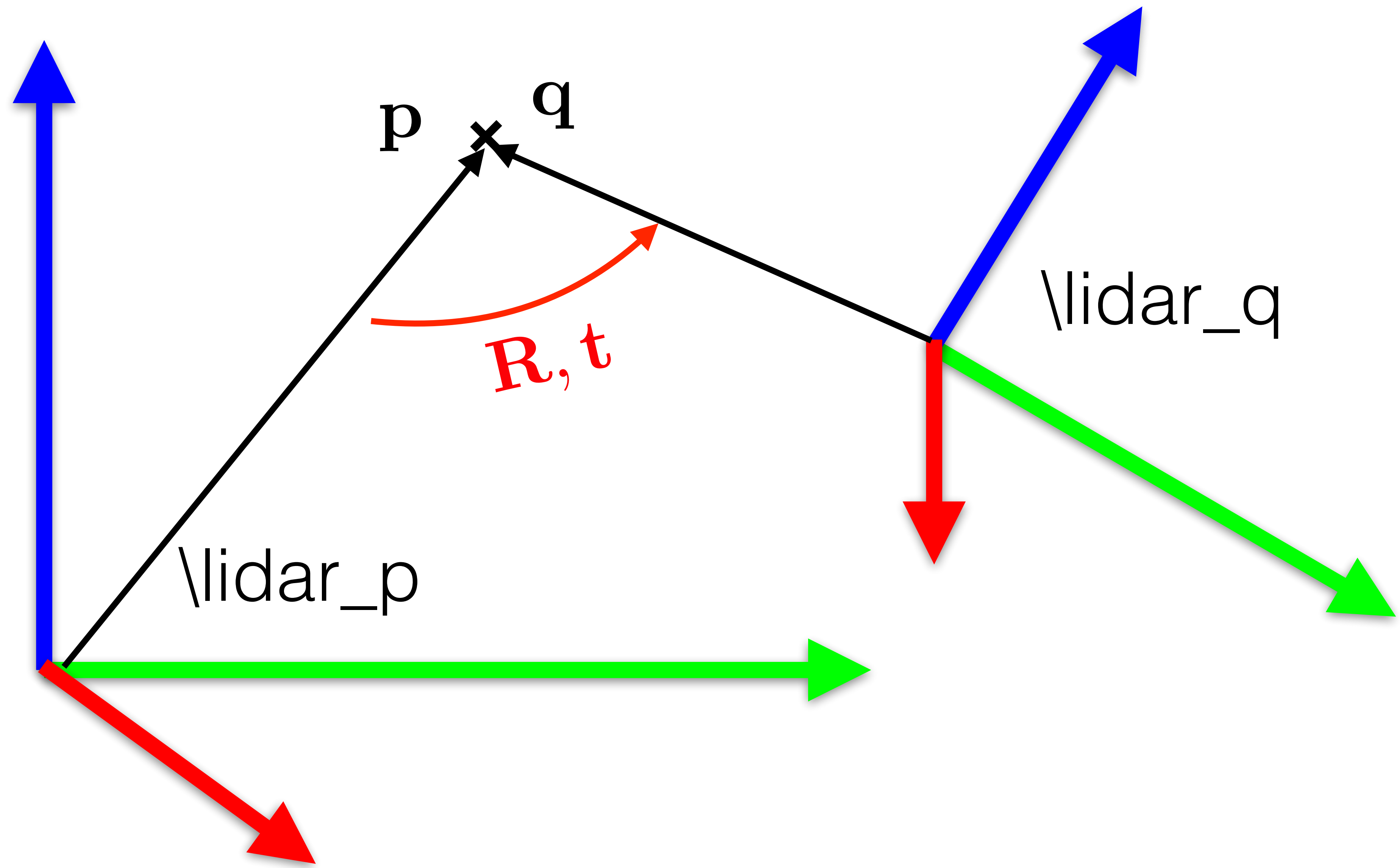
$$\mathbf{P} = [\underbrace{\mathbf{KR}}_{\mathbf{B}} \;\; \underbrace{\mathbf{Kt}}_{\mathbf{c}}] = [\mathbf{B} \;\; \mathbf{c}]$$

$$\mathbf{K}, \mathbf{R} = qr(\mathbf{B})$$
$$\mathbf{t} = \mathbf{K}^{-1}\mathbf{c}$$

(python: `numpy.linalg.qr`)

Summary

lidar-lidar calibration from 3D-3D correspondences

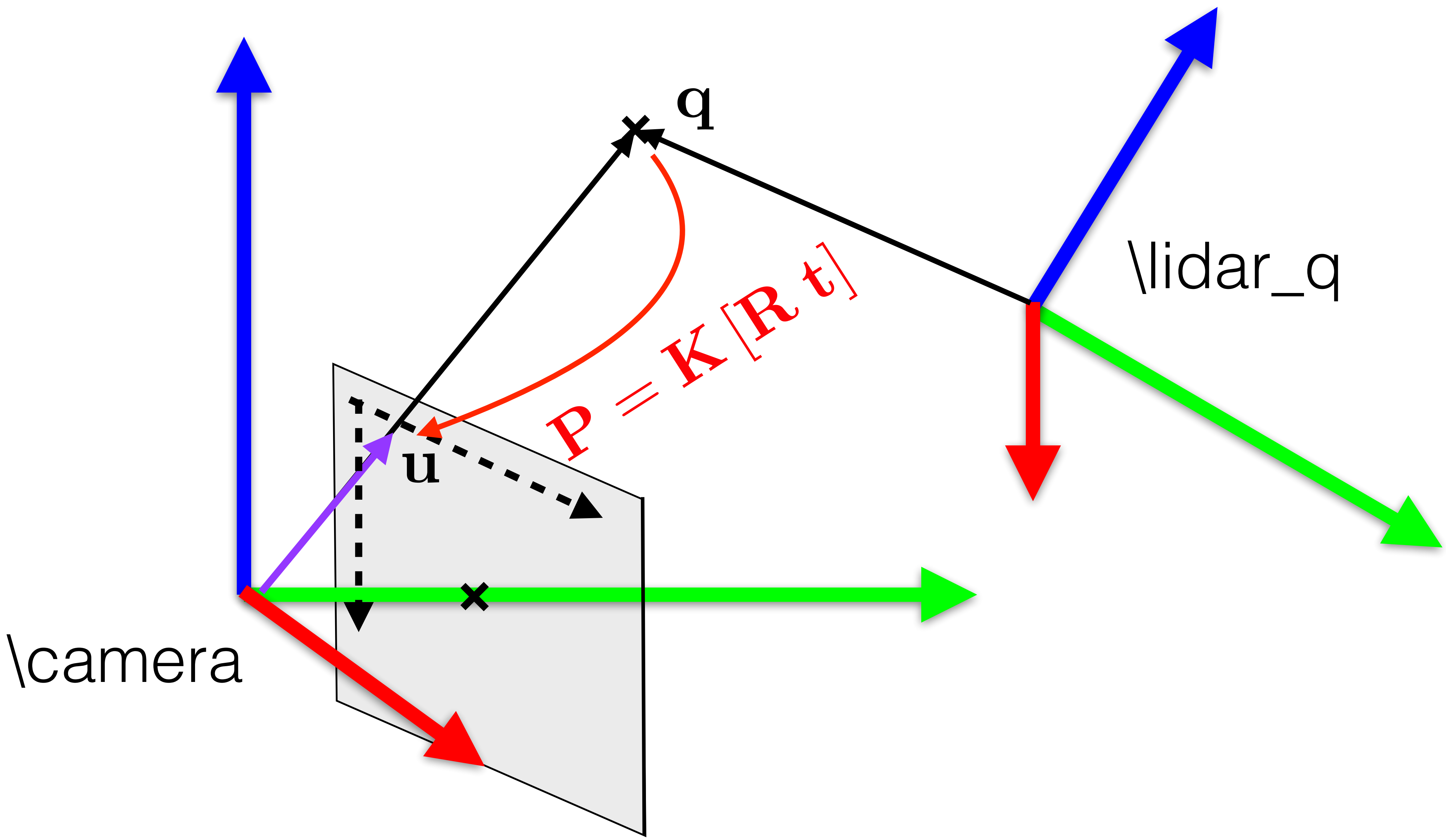


Solve: $$\mathbf{R}^*, \mathbf{t}^* = \arg\min_{\mathbf{R}\in SO(3), \mathbf{t}\in\mathcal{R}^3} \sum_i \|\mathbf{R}\mathbf{p}_i + \mathbf{t} - \mathbf{q}_i\|_2^2$$

Solution: $$\mathbf{R}^* = \mathbf{V}\mathbf{U}^\top$$
$$\mathbf{t}^* = \widetilde{\mathbf{q}} - \mathbf{R}^*\widetilde{\mathbf{p}}$$

Summary

camera-lidar calibration from 2D-3D correspondences



Solve: $\mathbf{p}^* = \text{argmin}||\mathbf{A}\mathbf{p}|| \;\; \text{subject to} \;\; ||\mathbf{p}|| = 1$

Solution: smallest eigen-vector of $\mathbf{A}^\top \mathbf{A}$

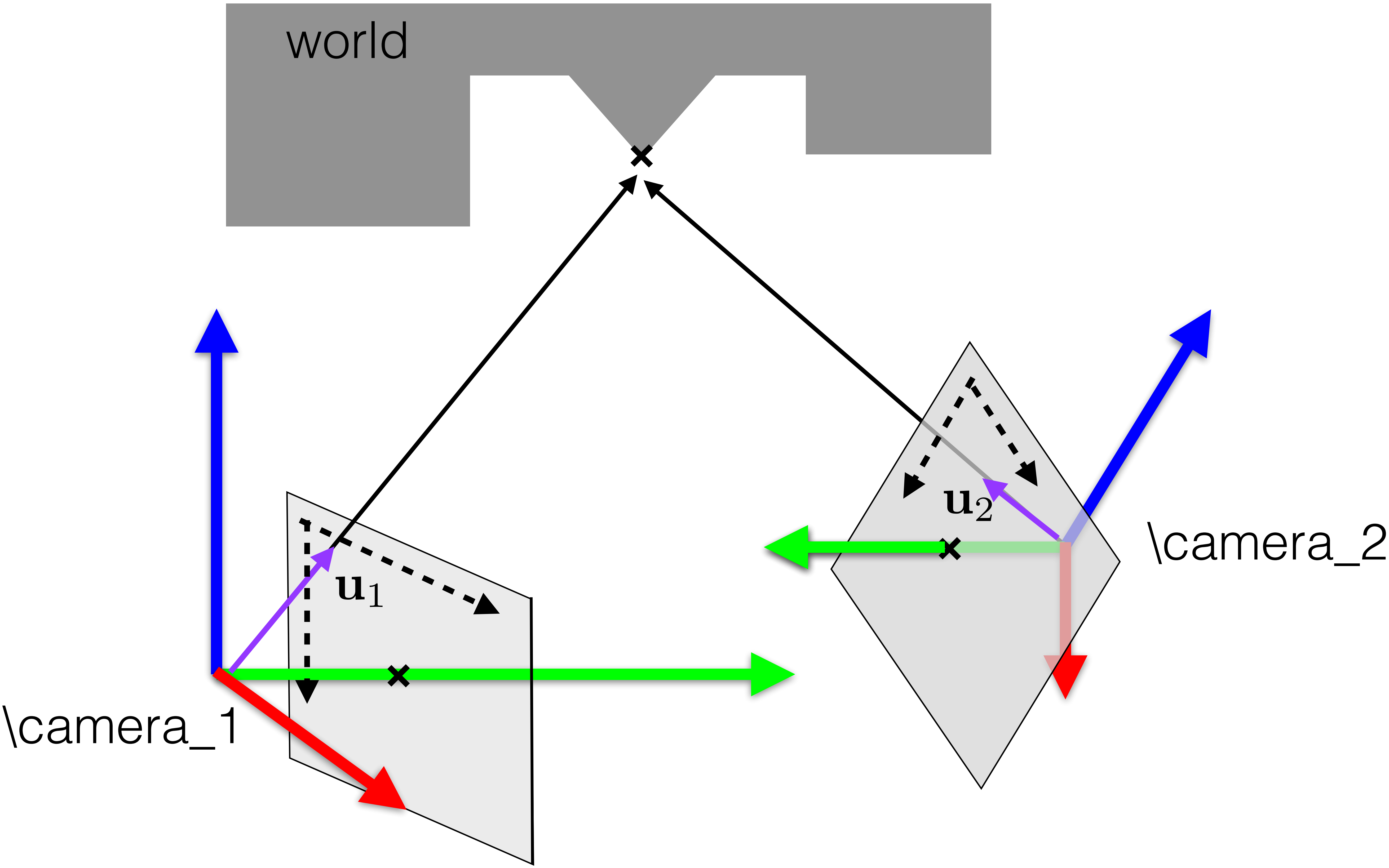
Summary
Broadcasting static transformation between two c.f. in ROS

Tutorial: http://wiki.ros.org/tf2/Tutorials/Writing%20a%20tf2%20static%20broadcaster%20%28Python%29

Stereo

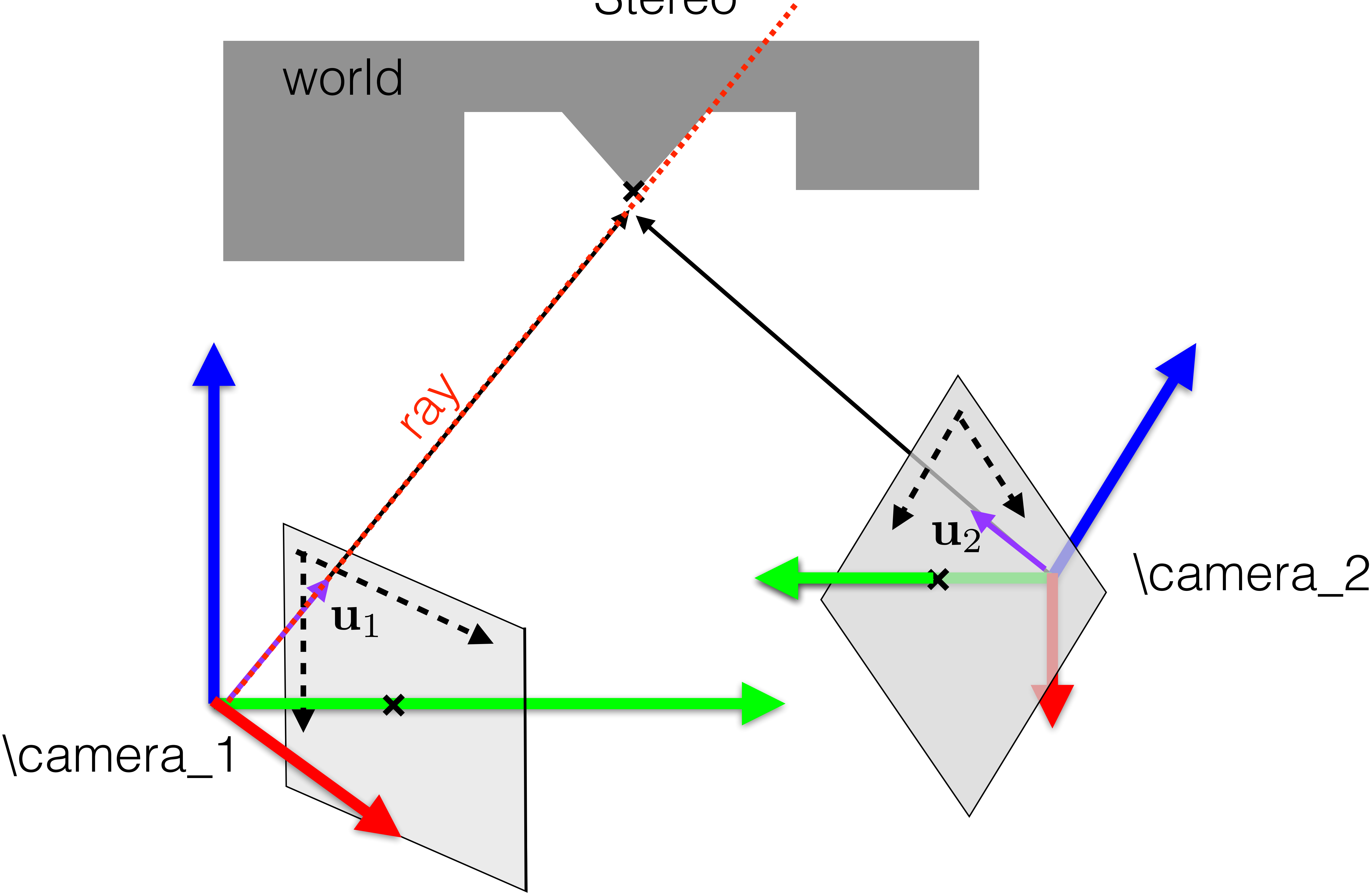
- Pair of cameras mounted on a common rigid body, which provide depth (or 3D point cloud).
- Simulate human binocular vision.
- In contrast to lidar, it is a passive sensor.

Stereo



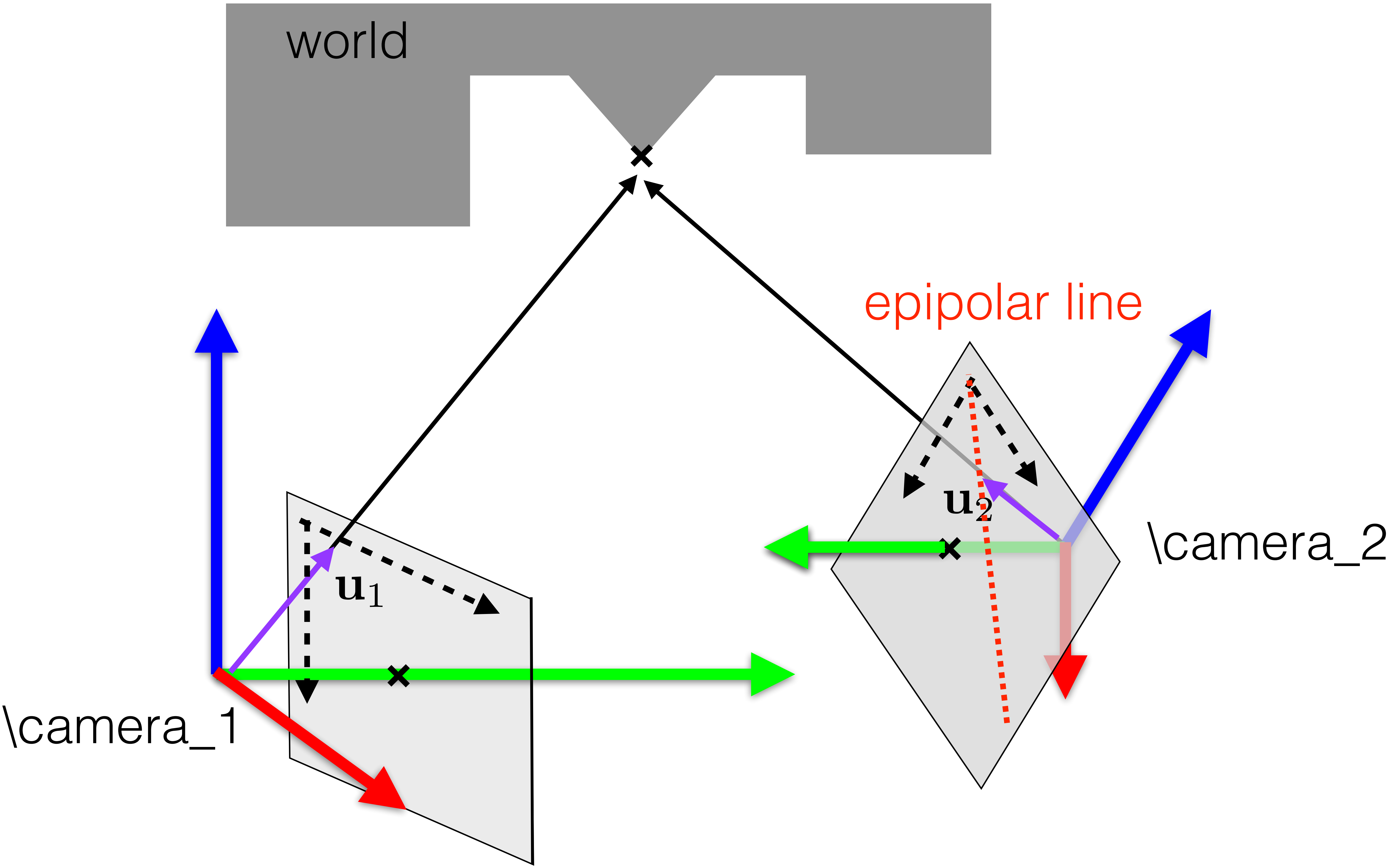
Given pixel $\mathbf{u}_1$ in \camera_1, where does the corresponding pixel $\mathbf{u}_2$ lie in \camera_2?

Stereo



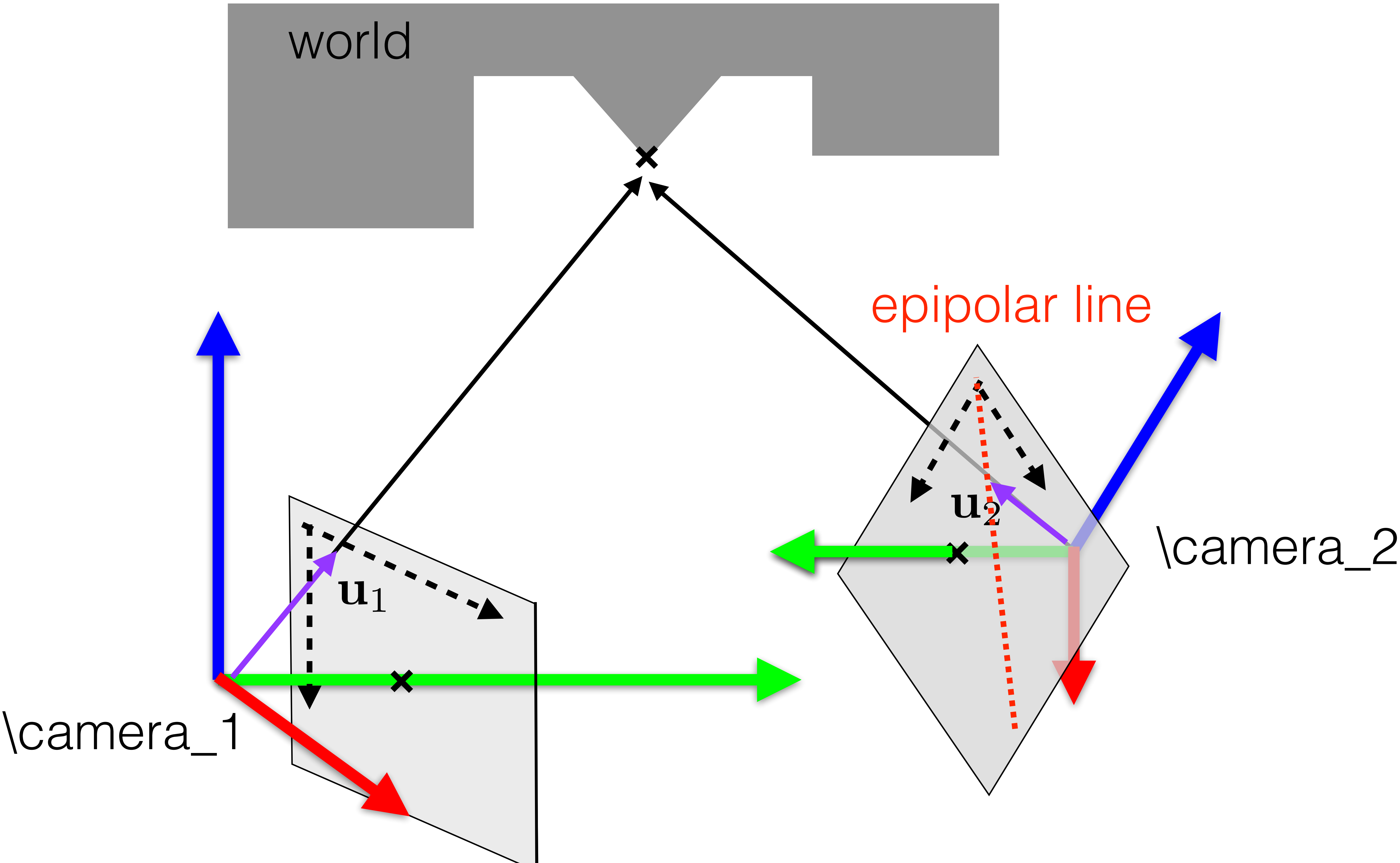
Given pixel $\mathbf{u}_1$ in \camera_1, where does the corresponding pixel $\mathbf{u}_2$ lie in \camera_2?

Stereo



Corresponding pixel lies on the epipolar line
(i.e. projection of the ray from camera_1 to \camera_2)

Stereo



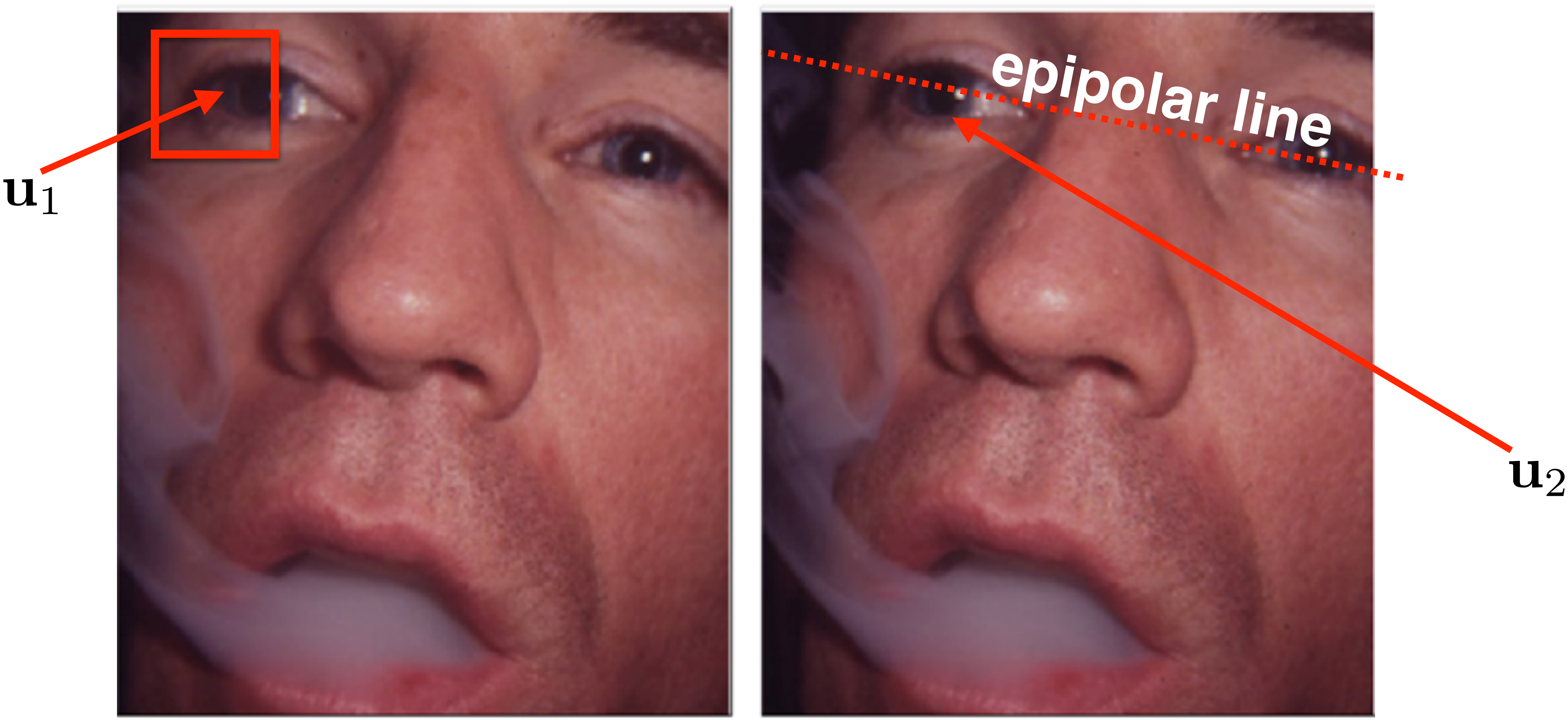
$$\mathcal{E} = \{\mathbf{u}_2 \in \mathcal{R}^2 \mid \; \overline{\mathbf{u}}_2^\top \underbrace{\mathbf{K}^{-1}(\mathbf{R} \times \mathbf{t})\mathbf{K}}_{\mathbf{F}} \overline{\mathbf{u}}_1 = 0\}$$

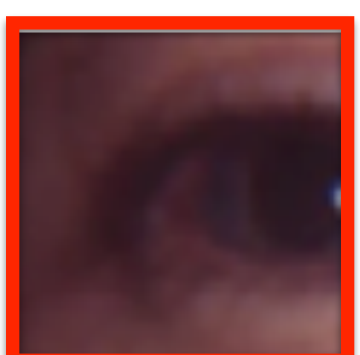
Stereo

Stereo

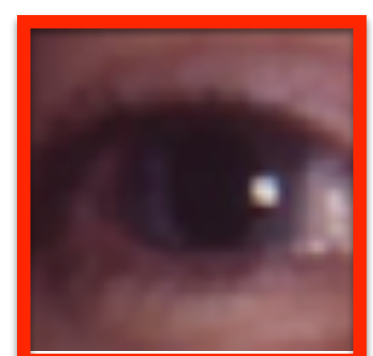
Stereo

Stereo



$I(\mathbf{x}) =$ 

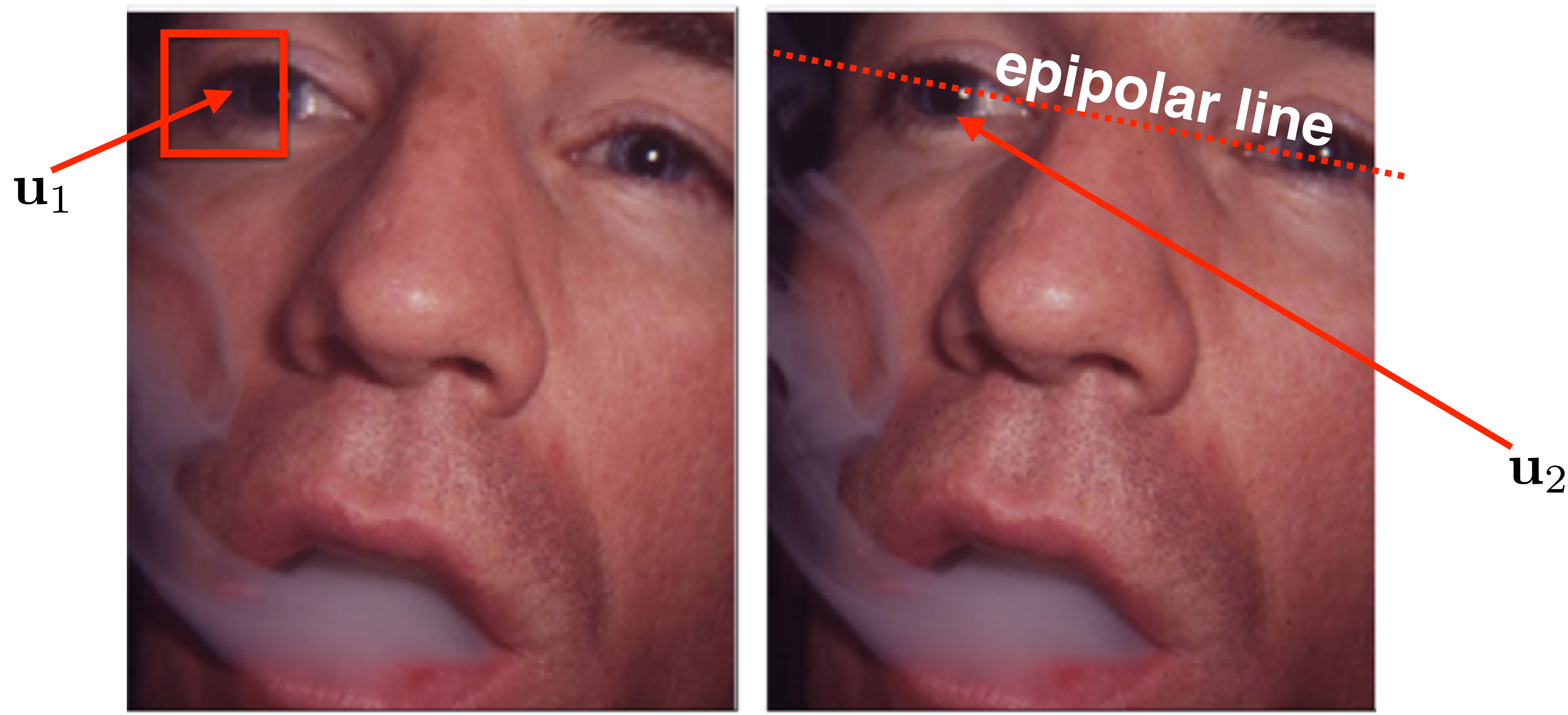
$\mathbf{x} \in \mathcal{W}$

$J(\mathbf{e} + \mathbf{x}) =$ 

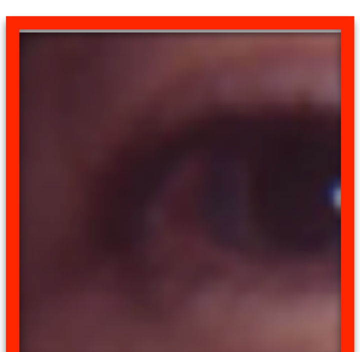
$\mathbf{x} \in \mathcal{W}$

$\mathbf{e} \in \mathcal{E}$

Stereo

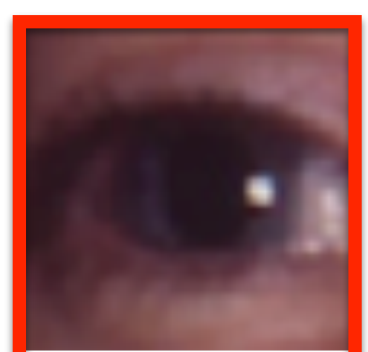


$I(\mathbf{x}) =$



$\mathbf{x} \in \mathcal{W}$

$J(\mathbf{e} + \mathbf{x}) =$



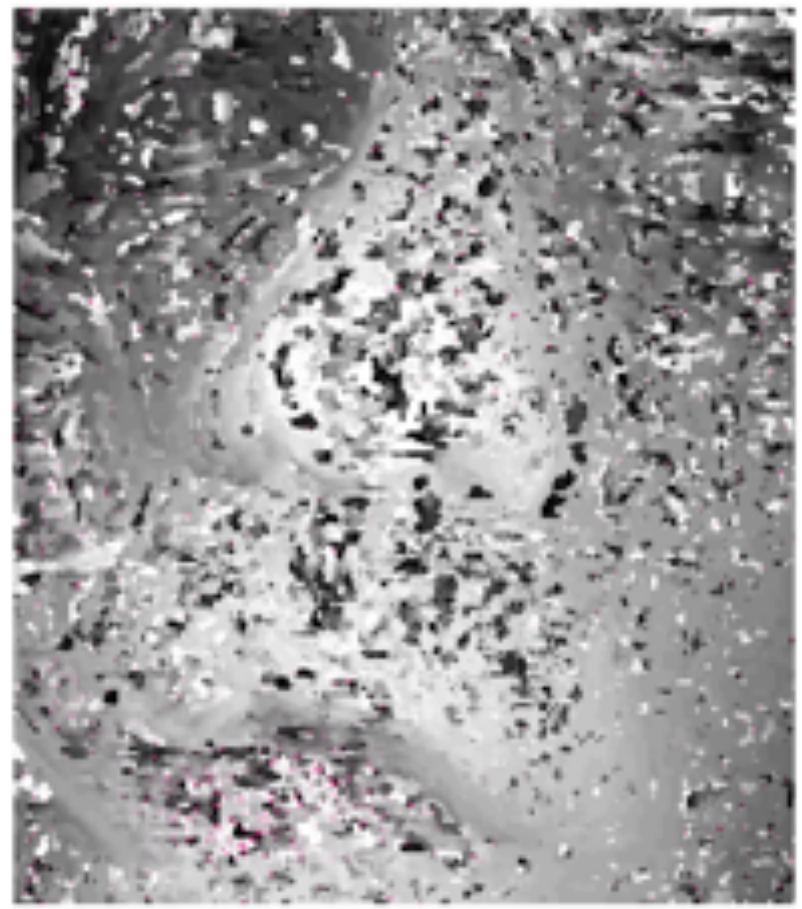
$\mathbf{x} \in \mathcal{W}$
$\mathbf{e} \in \mathcal{E}$

Similarity function:

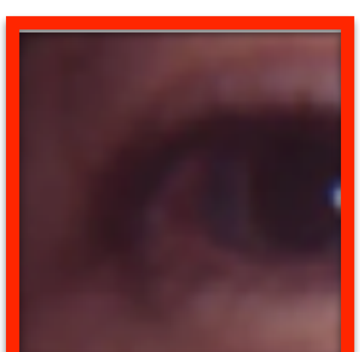
$$\sum_{\mathbf{x} \in \mathcal{W}} \Big(J(\mathbf{e} + \mathbf{x}) - I(\mathbf{x}) \Big)^2$$

Stereo

greedy solution

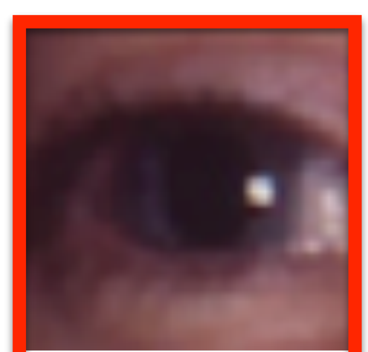


$$I(\mathbf{x}) =$$



$$\mathbf{x} \in \mathcal{W}$$

$$J(\mathbf{e}+\mathbf{x}) =$$



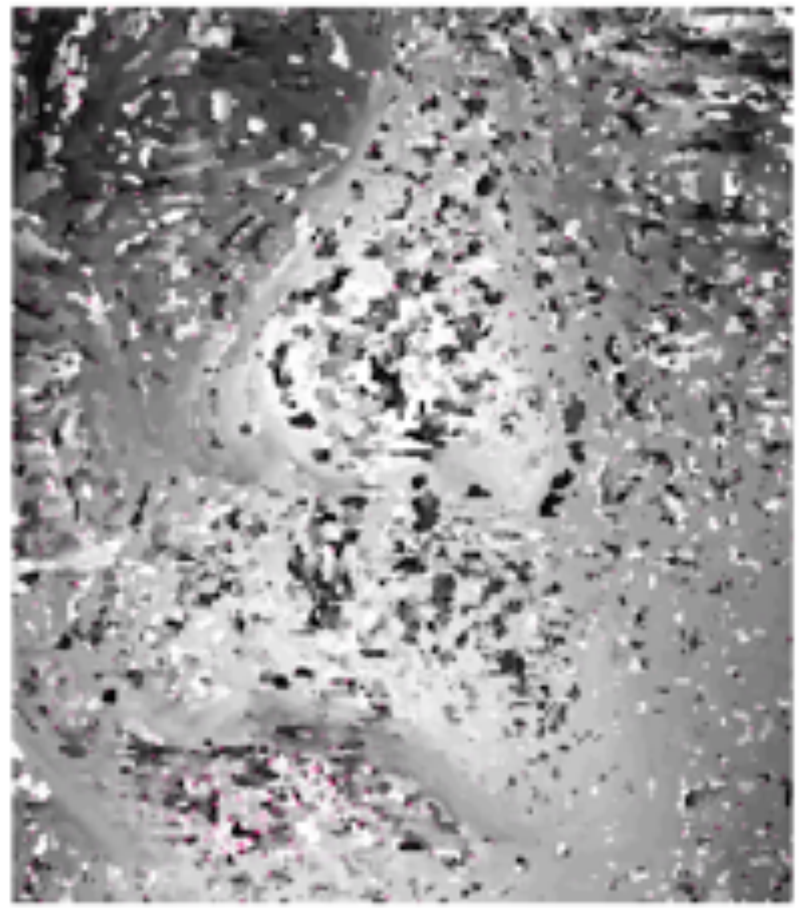
$$\mathbf{x} \in \mathcal{W}$$
$$\mathbf{e} \in \mathcal{E}$$

Similarity function:

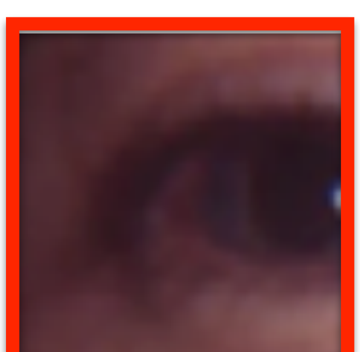
$$\sum_{\mathbf{x}\in\mathcal{W}} \Big(J(\mathbf{e}+\mathbf{x}) - I(\mathbf{x}) \Big)^2$$

Stereo

greedy solution | line smoothness

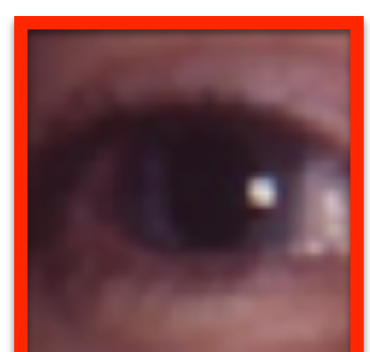


$I(\mathbf{x}) =$



$\mathbf{x} \in \mathcal{W}$

$J(\mathbf{e}+\mathbf{x}) =$



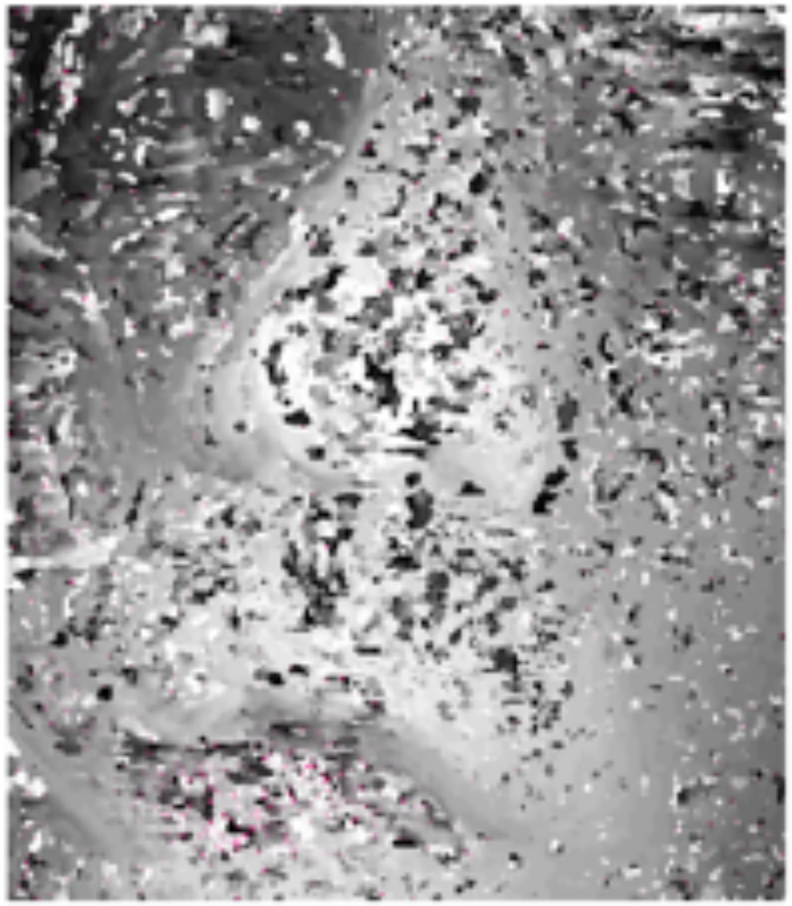
$\mathbf{x} \in \mathcal{W}$
$\mathbf{e} \in \mathcal{E}$

Similarity function:

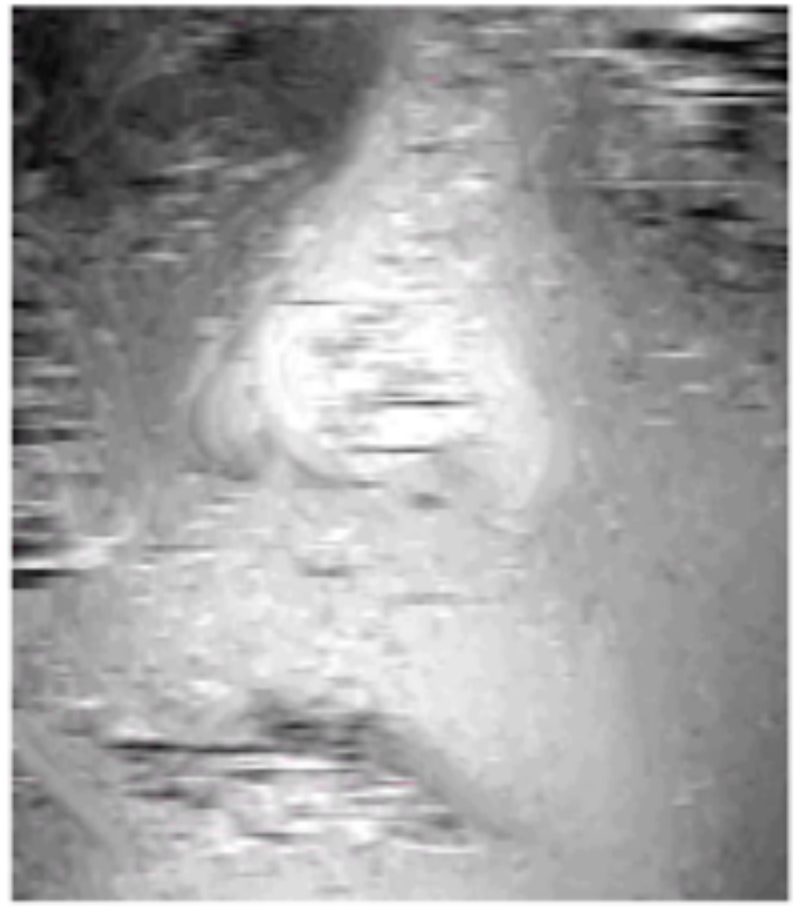
$$\sum_{\mathbf{x}\in\mathcal{W}} \Big(J(\mathbf{e}+\mathbf{x}) - I(\mathbf{x}) \Big)^2$$

Stereo

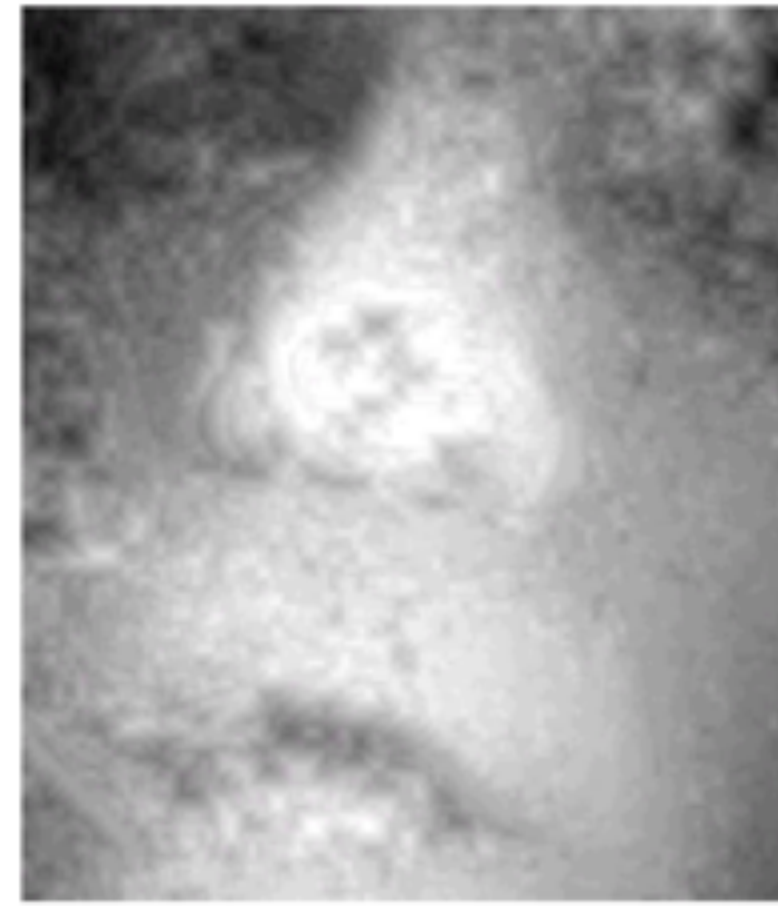
greedy solution

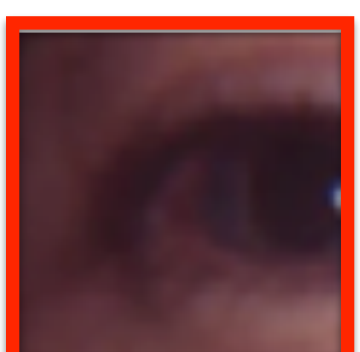


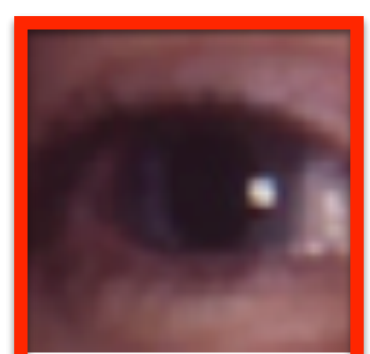
line smoothness



neighbourhood smoothness



$I(\mathbf{x}) =$  $\mathbf{x} \in \mathcal{W}$

$J(\mathbf{e} + \mathbf{x}) =$  $\mathbf{x} \in \mathcal{W}$, $\mathbf{e} \in \mathcal{E}$

Similarity function:

$$\sum_{\mathbf{x} \in \mathcal{W}} \Big(J(\mathbf{e} + \mathbf{x}) - I(\mathbf{x}) \Big)^2$$

Stereo: summary

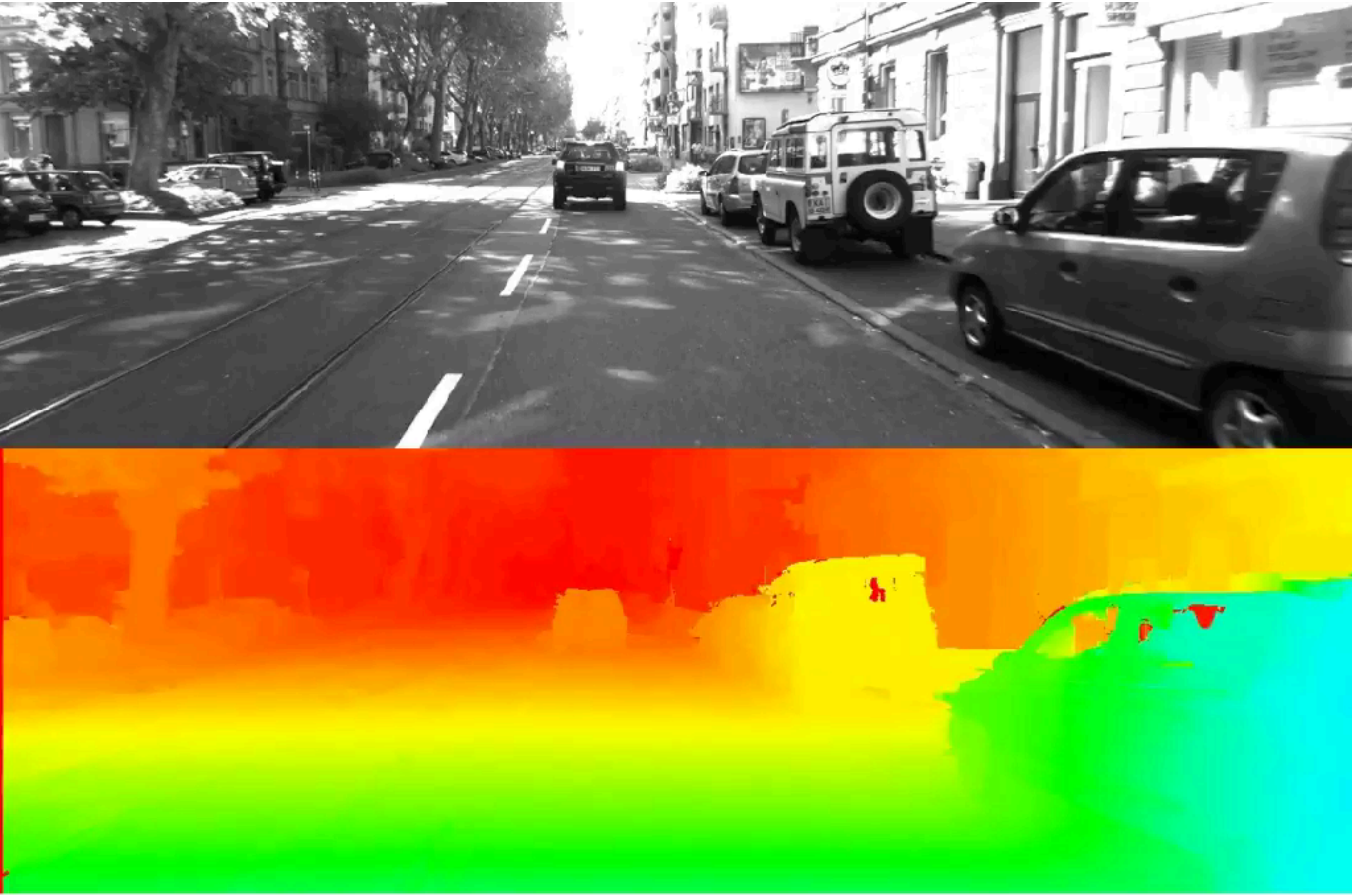
- Passive depth sensor created from pair of cameras.
- Inaccurate on long distance (sub-pixel disparity).
- Works well on textured, not reflective, smooth surfaces.
- Computationally demanding optimisation.
- Some OpenCV implementation:

```
stereo = cv2.createStereoBM(numDisparities=16,
blockSize=15)
depth = stereo.compute(imgL,imgR)
```

https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_calib3d/py_depthmap/py_depthmap.html

https://docs.opencv.org/3.1.0/d3/d14/tutorial_ximgproc_disparity_filtering.html#gsc.tab=0

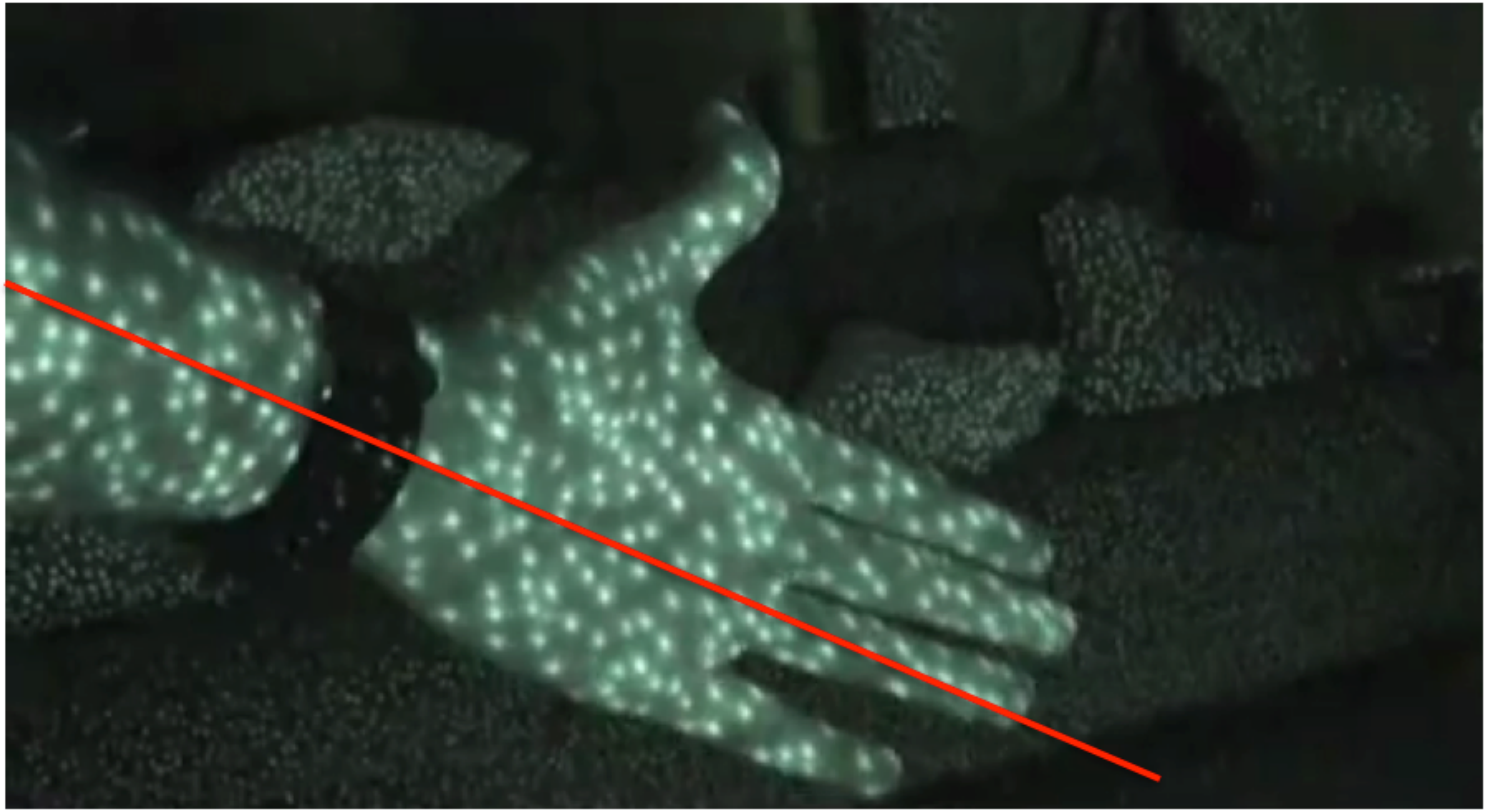
Stereo: summary



Kinect

- **Stereo** looks at the same object two-times and estimate its depth from two RGB images.
- **Kinect** avoid ambiguity by actively projecting a unique IR pattern on the object and search for its known appearance in the IR camera.

Kinect

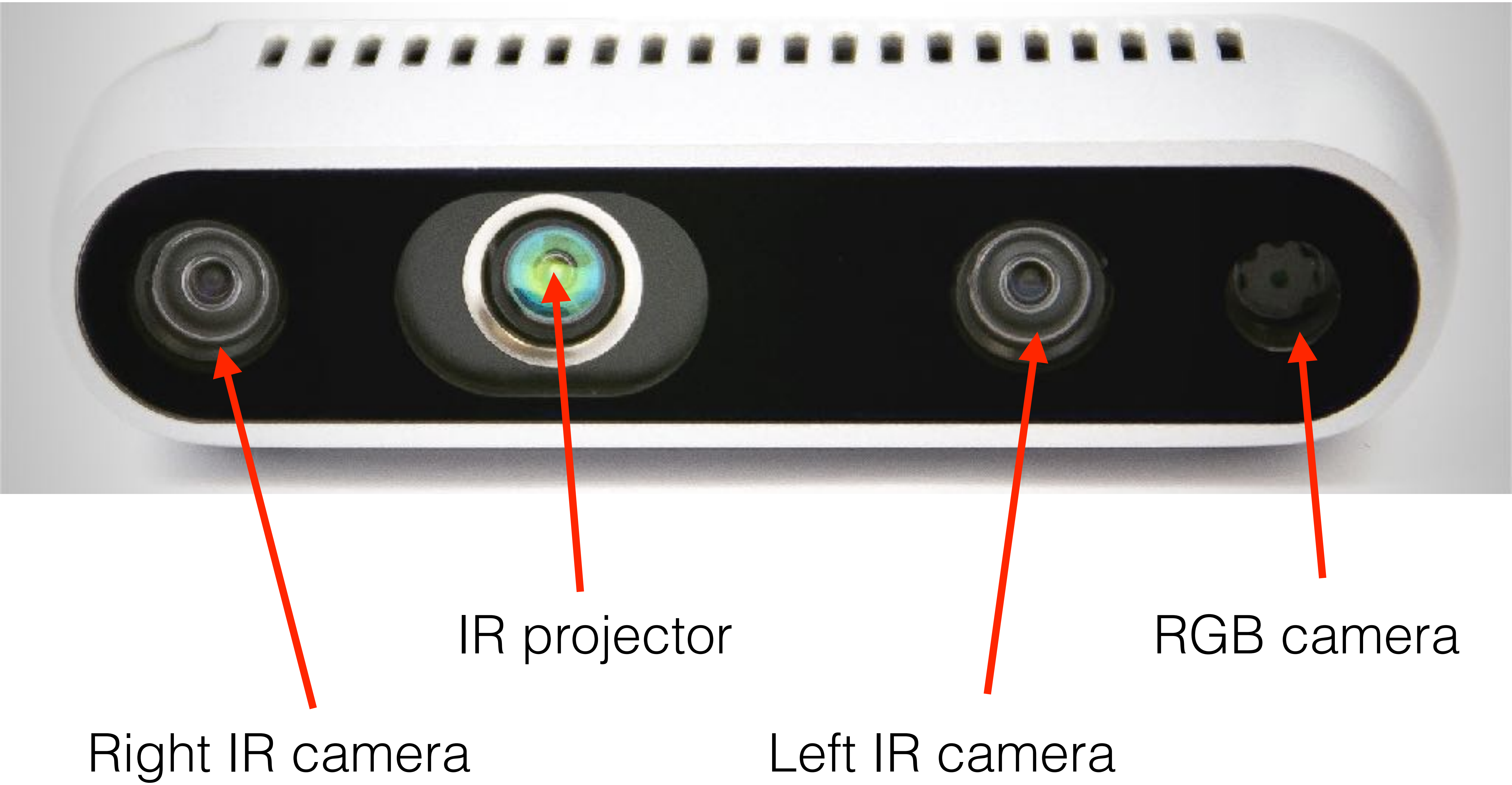


- Fixed camera-projector relative position.
- Correspondence between projected patch and observed patch lies on the epipolar line.

Summary: Kinect

- Active depth sensor consisting of IR camera and projector.
- Does not work outdoor due to strong illumination.
- Inaccurate on long distances.
- It does not require well textured surface.
- Cheap and fast solution for indoor robotics.

RealSense

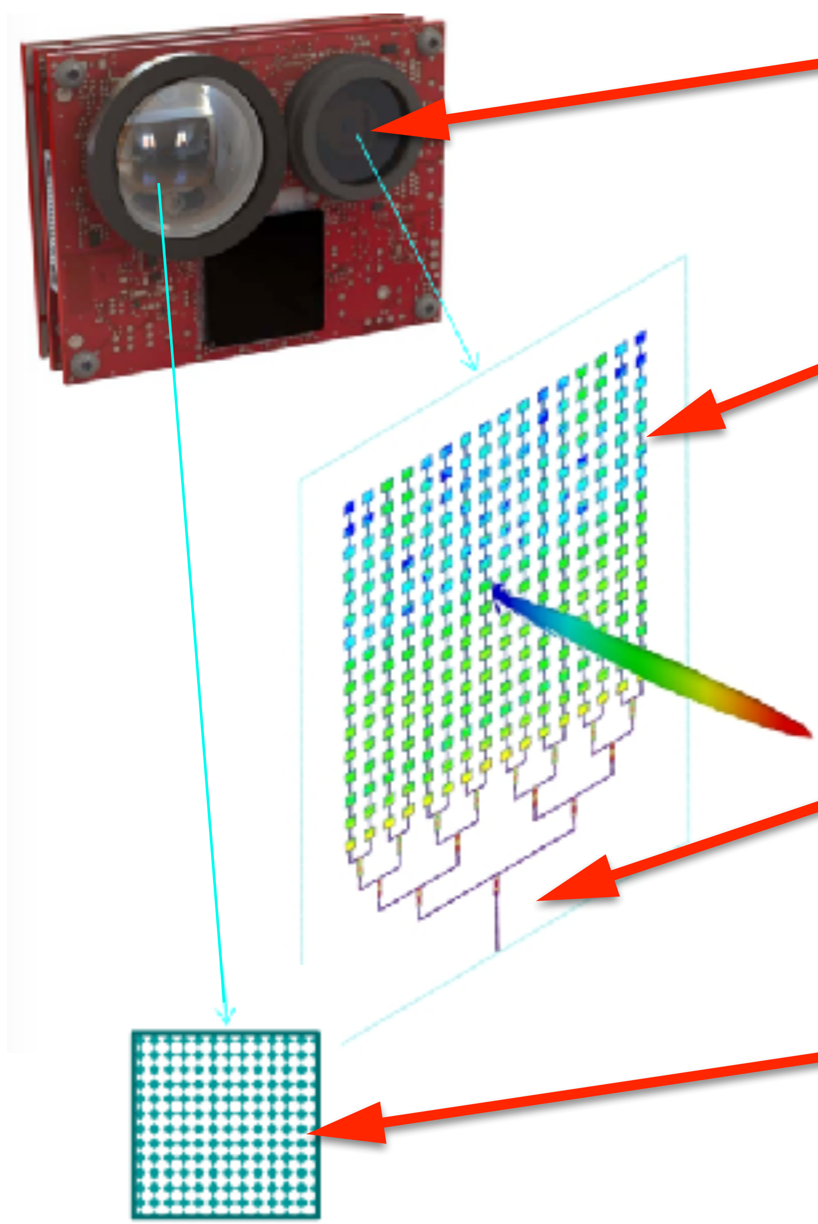


- **Indoor:** IR projector avoid ambiguities by projecting unique IR pattern
- **Outdoor:** It work like stereo in IR spectrum.

Solid-state lidar

Lidar with independent steering of depth-measuring rays

S3 principle



Emitted laser beams

Transmitted through Optical Phased Array

Controlling optical properties of OPA elements, allows to steer laser beams in desired directions

Reflected laser beams are captured by SPAD array

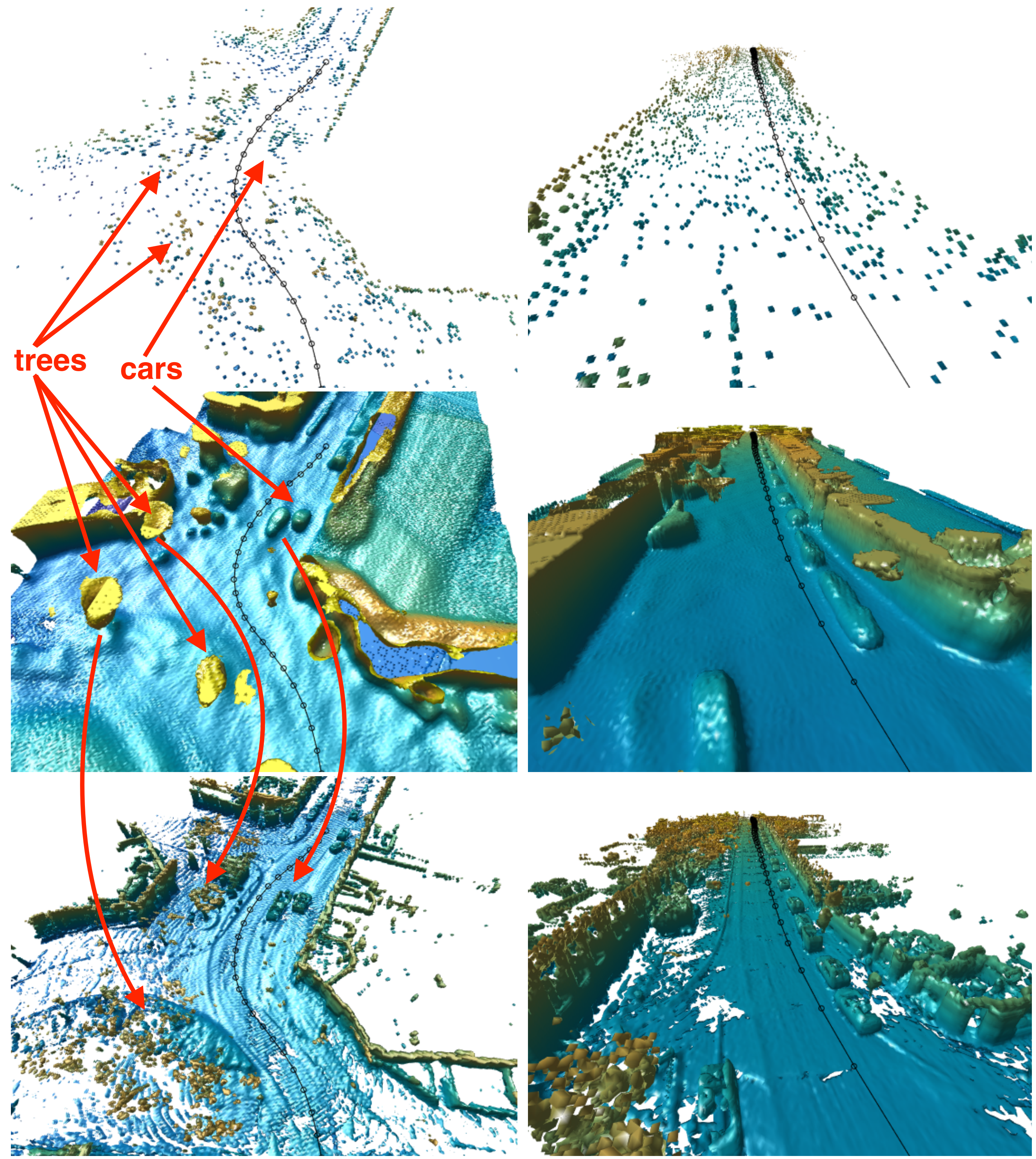
Images of S3 Lidar redistributed with permission of Quanergy Systems (http://quanergy.com)

Experiment: Qualitative evaluation

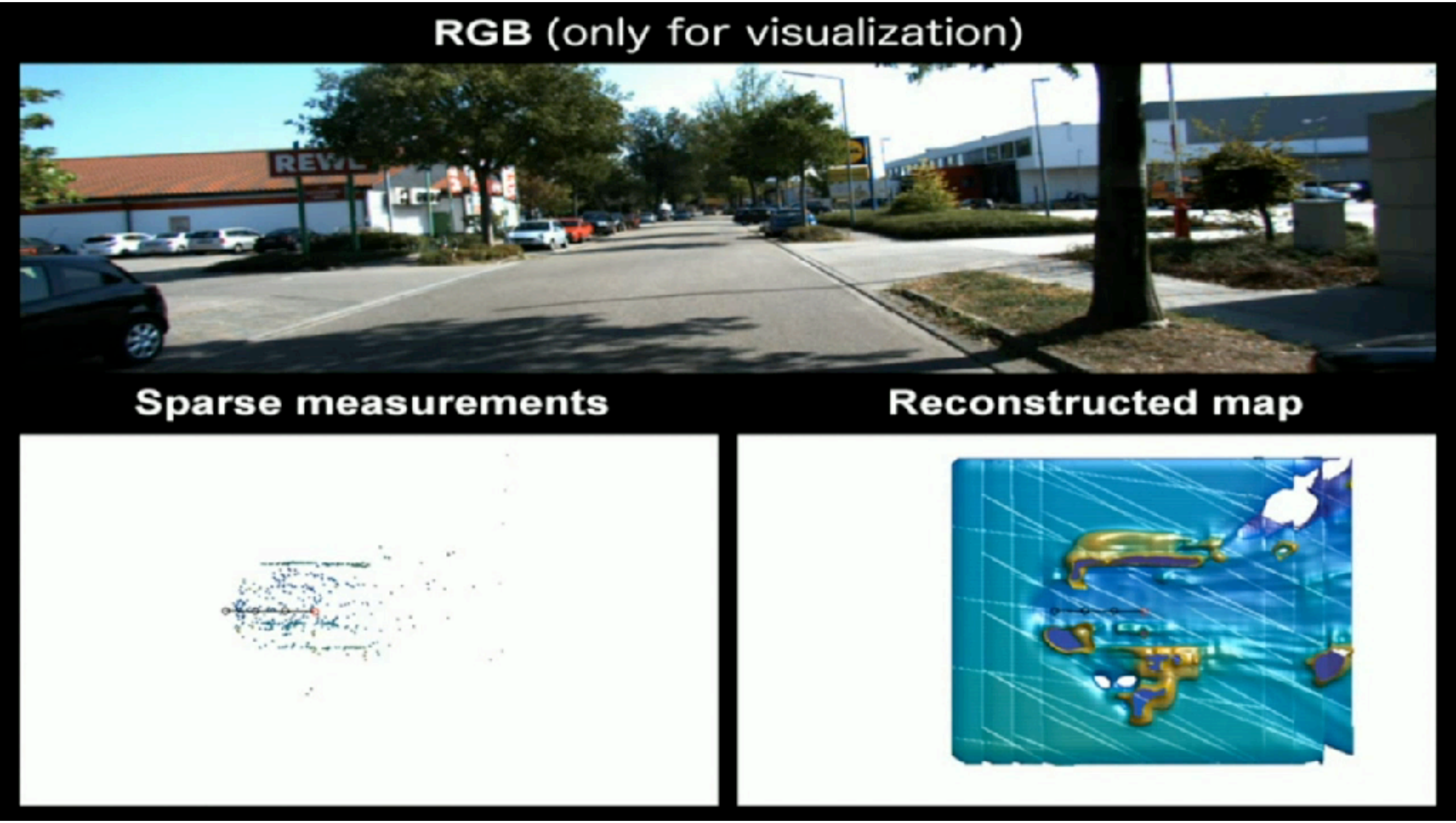
Sparse measurements

Reconstructed map

Ground truth



Active mapping [Zimmermann, Petricek et al. ICCV 2017]



[1] Zimmermann, Petricek, Salansky, Svoboda, Learning for Active 3D Mapping, **ICCV oral**, 2017 https://arxiv.org/abs/1708.02074