Computer Vision: Cameras

Raquel Urtasun

TTI Chicago

Jan 29, 2013

Readings

- Chapter 2.1, 3.6, 4.3 and 6.1 of Szeliski's book
- Chapter 1 of Forsyth & Ponce

What did we see in class last week?

Image Alignment Algorithm

Given images A and B

1. Compute image features for A and B
2. Match features between A and B
3. Compute homography between A and B using least squares on set of matches

Is there a problem with this?

[Source: N. Snavely]

Image Alignment Algorithm

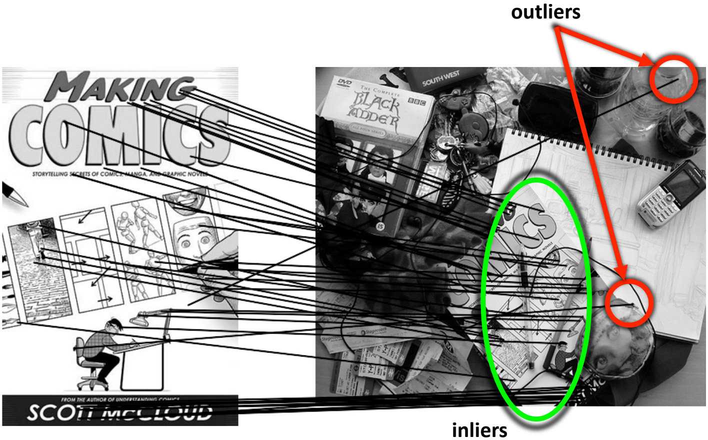
Given images A and B

1. Compute image features for A and B
2. Match features between A and B
3. Compute homography between A and B using least squares on set of matches

Is there a problem with this?

[Source: N. Snavely]

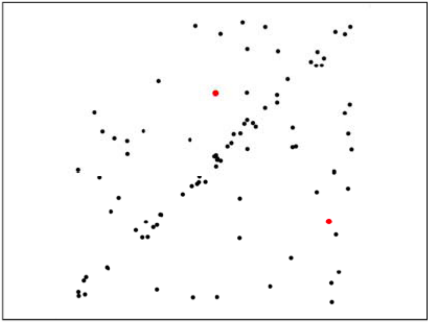
Robustness



[Source: N. Snavely]

RANSAC for line fitting example

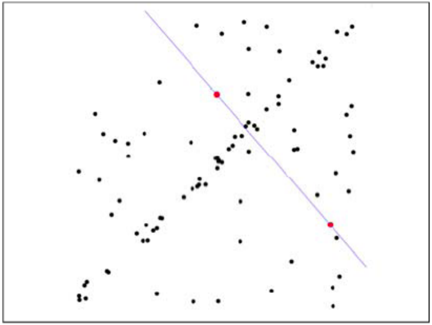
1. Randomly select minimal subset of points
2. Hypothesize a model



[Source: R. Raguram]

RANSAC for line fitting example

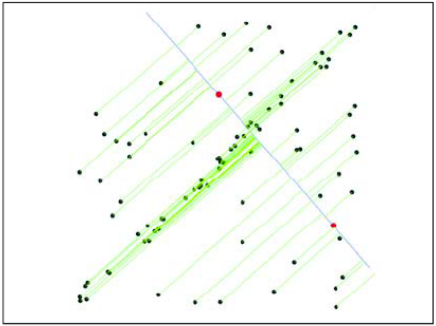
1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function



[Source: R. Raguram]

RANSAC for line fitting example

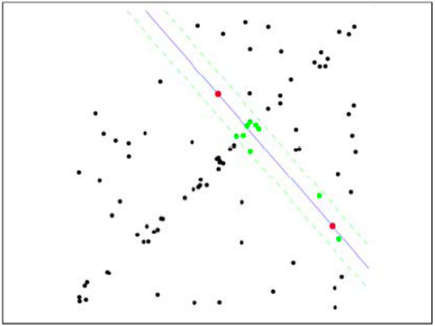
1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model



[Source: R. Raguram]

RANSAC for line fitting example

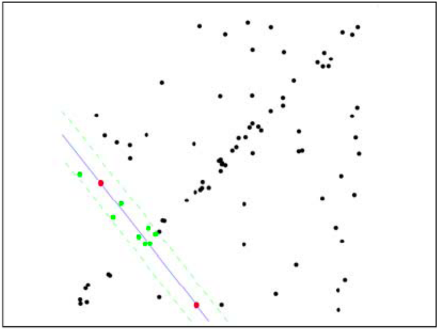
1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat hypothesize and verify loop



[Source: R. Raguram]

RANSAC for line fitting example

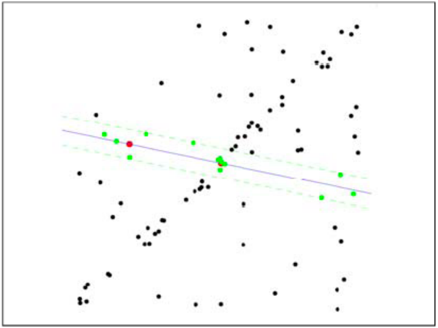
1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat hypothesize and verify loop



[Source: R. Raguram]

RANSAC for line fitting example

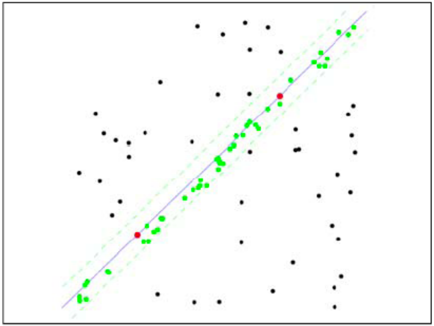
1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat hypothesize and verify loop
6. Choose model with largest set of inliers



[Source: R. Raguram]

RANSAC for line fitting example

1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat hypothesize and verify loop
6. Choose model with largest set of inliers



[Source: R. Raguram]

RANSAC

- **Inlier threshold** related to the amount of noise we expect in inliers
- Often model noise as Gaussian with some standard deviation (e.g., 3 pixels)

RANSAC

- **Inlier threshold** related to the amount of noise we expect in inliers
- Often model noise as Gaussian with some standard deviation (e.g., 3 pixels)
- **Number of rounds** related to the percentage of outliers we expect, and the probability of success we'd like to guarantee

RANSAC

- **Inlier threshold** related to the amount of noise we expect in inliers
- Often model noise as Gaussian with some standard deviation (e.g., 3 pixels)
- **Number of rounds** related to the percentage of outliers we expect, and the probability of success we'd like to guarantee
- Suppose there are 20% outliers, and we want to find the correct answer with 99% probability

RANSAC

- **Inlier threshold** related to the amount of noise we expect in inliers
- Often model noise as Gaussian with some standard deviation (e.g., 3 pixels)
- **Number of rounds** related to the percentage of outliers we expect, and the probability of success we'd like to guarantee
- Suppose there are 20% outliers, and we want to find the correct answer with 99% probability
- How many rounds do we need?

RANSAC

- **Inlier threshold** related to the amount of noise we expect in inliers
- Often model noise as Gaussian with some standard deviation (e.g., 3 pixels)
- **Number of rounds** related to the percentage of outliers we expect, and the probability of success we'd like to guarantee
- Suppose there are 20% outliers, and we want to find the correct answer with 99% probability
- How many rounds do we need?

How many rounds?

- Sufficient number of trials S must be tried.
- Let p be the probability that any given correspondence is valid and P be the total probability of success after S trials.

How many rounds?

- Sufficient number of trials S must be tried.
- Let p be the probability that any given correspondence is valid and P be the total probability of success after S trials.
- The likelihood in one trial that all k random samples are inliers is p^k

How many rounds?

- Sufficient number of trials S must be tried.
- Let p be the probability that any given correspondence is valid and P be the total probability of success after S trials.
- The likelihood in one trial that all k random samples are inliers is p^k
- The likelihood that S such trials will all fail is

$$1 - P = (1 - p^k)^S$$

How many rounds?

- Sufficient number of trials S must be tried.
- Let p be the probability that any given correspondence is valid and P be the total probability of success after S trials.
- The likelihood in one trial that all k random samples are inliers is p^k
- The likelihood that S such trials will all fail is

$$1 - P = (1 - p^k)^S$$

- The required minimum number of trials is

$$S = \frac{\log(1 - P)}{\log(1 - p^k)}$$

How many rounds?

- Sufficient number of trials S must be tried.
- Let p be the probability that any given correspondence is valid and P be the total probability of success after S trials.
- The likelihood in one trial that all k random samples are inliers is p^k
- The likelihood that S such trials will all fail is

$$1 - P = (1 - p^k)^S$$

- The required minimum number of trials is

$$S = \frac{\log(1 - P)}{\log(1 - p^k)}$$

- The number of trials grows quickly with the number of sample points used.

How many rounds?

- Sufficient number of trials S must be tried.
- Let p be the probability that any given correspondence is valid and P be the total probability of success after S trials.
- The likelihood in one trial that all k random samples are inliers is p^k
- The likelihood that S such trials will all fail is

$$1 - P = (1 - p^k)^S$$

- The required minimum number of trials is

$$S = \frac{\log(1 - P)}{\log(1 - p^k)}$$

- The number of trials grows quickly with the number of sample points used.
- Use the minimum number of sample points k possible for any given trial

How many rounds?

- Sufficient number of trials S must be tried.
- Let p be the probability that any given correspondence is valid and P be the total probability of success after S trials.
- The likelihood in one trial that all k random samples are inliers is p^k
- The likelihood that S such trials will all fail is

$$1 - P = (1 - p^k)^S$$

- The required minimum number of trials is

$$S = \frac{\log(1 - P)}{\log(1 - p^k)}$$

- The number of trials grows quickly with the number of sample points used.
- Use the minimum number of sample points k possible for any given trial

RANSAC pros and cons

Pros

- Simple and general
- Applicable to many different problems

RANSAC pros and cons

Pros

- Simple and general
- Applicable to many different problems
- Often works well in practice

RANSAC pros and cons

Pros

- Simple and general
- Applicable to many different problems
- Often works well in practice

Cons

- Parameters to tune

RANSAC pros and cons

Pros

- Simple and general
- Applicable to many different problems
- Often works well in practice

Cons

- Parameters to tune
- Sometimes too many iterations are required

RANSAC pros and cons

Pros

- Simple and general
- Applicable to many different problems
- Often works well in practice

Cons

- Parameters to tune
- Sometimes too many iterations are required
- Can fail for extremely low inlier ratios

RANSAC pros and cons

Pros

- Simple and general
- Applicable to many different problems
- Often works well in practice

Cons

- Parameters to tune
- Sometimes too many iterations are required
- Can fail for extremely low inlier ratios
- We can often do better than brute-force sampling

[Source: N. Snavely]

RANSAC pros and cons

Pros

- Simple and general
- Applicable to many different problems
- Often works well in practice

Cons

- Parameters to tune
- Sometimes too many iterations are required
- Can fail for extremely low inlier ratios
- We can often do better than brute-force sampling

[Source: N. Snavely]

RANSAC as Voting

- An example of a "voting"-based fitting scheme
- Each hypothesis gets voted on by each data point, best hypothesis wins

RANSAC as Voting

- An example of a "voting"-based fitting scheme
- Each hypothesis gets voted on by each data point, best hypothesis wins
- There are many other types of voting schemes, e.g., Hough transforms

[Source: N. Snavely]

RANSAC as Voting

- An example of a "voting"-based fitting scheme
- Each hypothesis gets voted on by each data point, best hypothesis wins
- There are many other types of voting schemes, e.g., Hough transforms

[Source: N. Snavely]

Hough Transform

- Alternatively one can have the points vote for the parameters.
- Let's consider a simple example, we want to obtain lines in images

Hough Transform

- Alternatively one can have the points vote for the parameters.
- Let's consider a simple example, we want to obtain lines in images
- Let's make each point (e.g., edge) cast a vote for the possible lines that go through

Hough Transform

- Alternatively one can have the points vote for the parameters.
- Let's consider a simple example, we want to obtain lines in images
- Let's make each point (e.g., edge) cast a vote for the possible lines that go through
- Parameterization is very important, in this case $x \cos\theta + y \sin\theta = r$

Hough Transform

- Alternatively one can have the points vote for the parameters.
- Let's consider a simple example, we want to obtain lines in images
- Let's make each point (e.g., edge) cast a vote for the possible lines that go through
- Parameterization is very important, in this case $x \cos\theta + y \sin\theta = r$
- How many degrees of freedom?

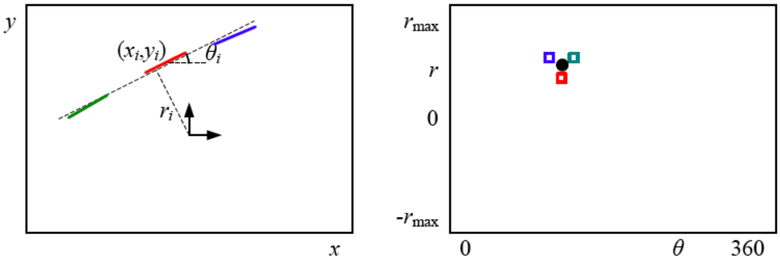


Figure: Images from Szeliski's book

Hough Transform

- Alternatively one can have the points vote for the parameters.
- Let's consider a simple example, we want to obtain lines in images
- Let's make each point (e.g., edge) cast a vote for the possible lines that go through
- Parameterization is very important, in this case $x \cos\theta + y \sin\theta = r$
- How many degrees of freedom?

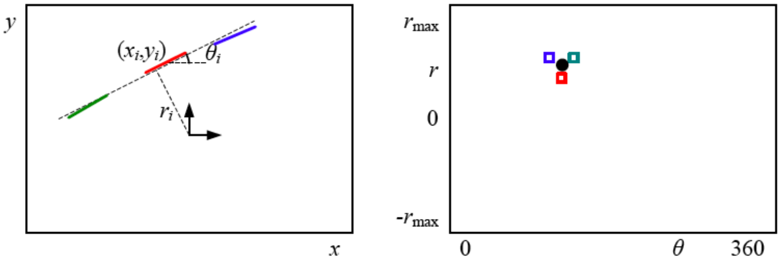
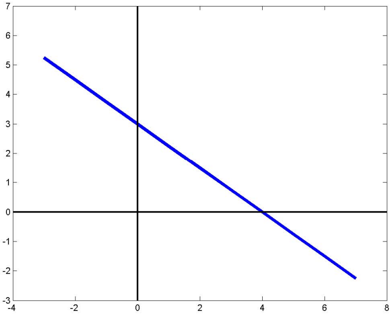


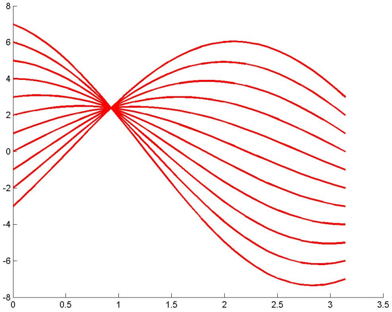
Figure: Images from Szeliski's book

Example Hough Transform

With the parameterization $x \cos\theta + y \sin\theta = r$

- Points in picture represent sinusoids in parameter space
- Points in parameter space represent lines in picture
- Example $0.6x + 0.4y = 2.4$, Sinusoids intersect at $r = 2.4$, $\theta = 0.9273$



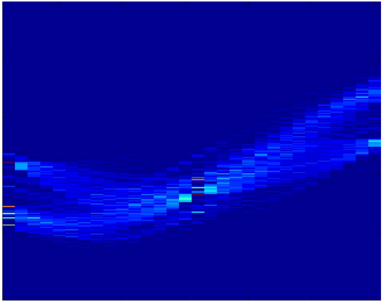


[Source: M. Kazhdan]

Hough Transform Algorithm

With the parameterization $x \cos\theta + y \sin\theta = r$

- Let $r \in [-R, R]$ and $\theta \in [0, \pi)$
- For each edge point (x_i, y_i), calculate: $\hat{r} = x_i \cos\hat{\theta} + y_i \sin\hat{\theta} \quad \forall\hat{\theta} \in [0, \pi)$
- Increase accumulator $A(\hat{r}, \hat{\theta}) = A(\hat{r}, \hat{\theta}) + 1$



- Threshold the accumulator values to get parameters for detected lines

[Source: M. Kazhdan]

Compensating for errors

- Errors can cause $A(\mathbf{p}')$ to be incremented, where p' is close to the actual parameter p
- Compensate for the uncertainty of measurement in parameter space

Compensating for errors

- Errors can cause $A(\mathbf{p}')$ to be incremented, where p' is close to the actual parameter p
- Compensate for the uncertainty of measurement in parameter space
- Smooth the accumulator by incrementing counts of nearby cells according to some point-spread function h

Compensating for errors

- Errors can cause $A(\mathbf{p}')$ to be incremented, where p' is close to the actual parameter p
- Compensate for the uncertainty of measurement in parameter space
- Smooth the accumulator by incrementing counts of nearby cells according to some point-spread function h
- Equivalent to convolving $A * h$

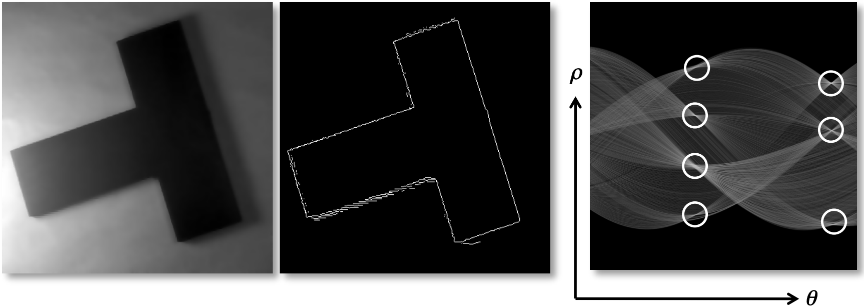
[Source: M. Kazhdan]

Compensating for errors

- Errors can cause $A(\mathbf{p}')$ to be incremented, where p' is close to the actual parameter p
- Compensate for the uncertainty of measurement in parameter space
- Smooth the accumulator by incrementing counts of nearby cells according to some point-spread function h
- Equivalent to convolving $A * h$

[Source: M. Kazhdan]

Hough Example



[Source: N. Snavely]

Coming back to our problem ...

Generalized Hough Transform vs Ransac

- It's not feasible to check all combinations of features by fitting a model to each possible subset.
- First, cycle through features, cast votes for model parameters: location, scale, orientation of the model object.

Coming back to our problem ...

Generalized Hough Transform vs Ransac

- It's not feasible to check all combinations of features by fitting a model to each possible subset.
- First, cycle through features, cast votes for model parameters: location, scale, orientation of the model object.
- Look for model parameters that receive a lot of votes, and verify them.

Coming back to our problem ...

Generalized Hough Transform vs Ransac

- It's not feasible to check all combinations of features by fitting a model to each possible subset.
- First, cycle through features, cast votes for model parameters: location, scale, orientation of the model object.
- Look for model parameters that receive a lot of votes, and verify them.
- Noise & clutter features will cast votes too, but their votes should be inconsistent with the majority of good features.

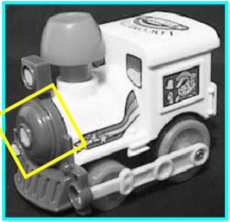
Coming back to our problem ...

Generalized Hough Transform vs Ransac

- It's not feasible to check all combinations of features by fitting a model to each possible subset.
- First, cycle through features, cast votes for model parameters: location, scale, orientation of the model object.
- Look for model parameters that receive a lot of votes, and verify them.
- Noise & clutter features will cast votes too, but their votes should be inconsistent with the majority of good features.

Generalized Hough Transform

- If we use scale, rotation, and translation invariant local features, then each feature match gives an alignment hypothesis (for scale, translation, and orientation of model in image).



[Source: S. Lazebnik]

Generalized Hough Transform

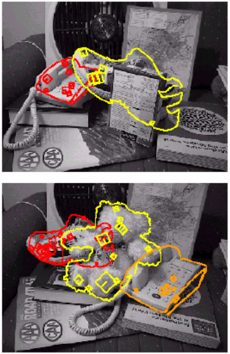
- A hypothesis generated by a single match is in general unreliable,
- Let each match vote for a hypothesis in Hough space.

[Source: K. Grauman]

Recognition Example

Background subtract for model boundaries

Objects recognized,



Recognition in spite of occlusion

Problems of Voting

- Noise/clutter can lead to as many votes as true target
- Bin size for the accumulator array must be chosen carefully

Problems of Voting

- Noise/clutter can lead to as many votes as true target
- Bin size for the accumulator array must be chosen carefully
- In practice, good idea to make broad bins and spread votes to nearby bins, since verification stage can prune bad vote peaks

Problems of Voting

- Noise/clutter can lead to as many votes as true target
- Bin size for the accumulator array must be chosen carefully
- In practice, good idea to make broad bins and spread votes to nearby bins, since verification stage can prune bad vote peaks

Comparison Verification

Generalized Hough Transform

- Each single correspondence votes for all consistent parameters
- Represents uncertainty on the parameter space

Comparison Verification

Generalized Hough Transform

- Each single correspondence votes for all consistent parameters
- Represents uncertainty on the parameter space
- Complexity: Beyond 4D space is impractical

Comparison Verification

Generalized Hough Transform

- Each single correspondence votes for all consistent parameters
- Represents uncertainty on the parameter space
- Complexity: Beyond 4D space is impractical
- Can handle high outlier/inlier ratio

Comparison Verification

Generalized Hough Transform

- Each single correspondence votes for all consistent parameters
- Represents uncertainty on the parameter space
- Complexity: Beyond 4D space is impractical
- Can handle high outlier/inlier ratio

Ransac

- Minimal subset of correspondences to estimate the model, then count inliers

Comparison Verification

Generalized Hough Transform

- Each single correspondence votes for all consistent parameters
- Represents uncertainty on the parameter space
- Complexity: Beyond 4D space is impractical
- Can handle high outlier/inlier ratio

Ransac

- Minimal subset of correspondences to estimate the model, then count inliers
- Represent uncertainty in image space

Comparison Verification

Generalized Hough Transform

- Each single correspondence votes for all consistent parameters
- Represents uncertainty on the parameter space
- Complexity: Beyond 4D space is impractical
- Can handle high outlier/inlier ratio

Ransac

- Minimal subset of correspondences to estimate the model, then count inliers
- Represent uncertainty in image space
- Must look at all points to check for inliers at each iteration

Comparison Verification

Generalized Hough Transform

- Each single correspondence votes for all consistent parameters
- Represents uncertainty on the parameter space
- Complexity: Beyond 4D space is impractical
- Can handle high outlier/inlier ratio

Ransac

- Minimal subset of correspondences to estimate the model, then count inliers
- Represent uncertainty in image space
- Must look at all points to check for inliers at each iteration
- Scales better with high dimensionality of parameter space.

Comparison Verification

Generalized Hough Transform

- Each single correspondence votes for all consistent parameters
- Represents uncertainty on the parameter space
- Complexity: Beyond 4D space is impractical
- Can handle high outlier/inlier ratio

Ransac

- Minimal subset of correspondences to estimate the model, then count inliers
- Represent uncertainty in image space
- Must look at all points to check for inliers at each iteration
- Scales better with high dimensionality of parameter space.

Panoramas

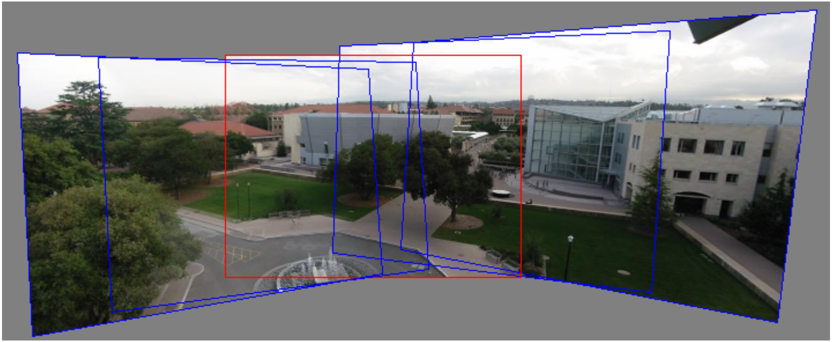
Given two images:

1. Detect features
2. Match features
3. Compute a homography using RANSAC
4. Combine the images together (somehow)

What if we have more than two images?

Creating Panoramas

- Can we use homographies to create a 360 panorama?



- In order to figure this out, we need to learn what a camera is

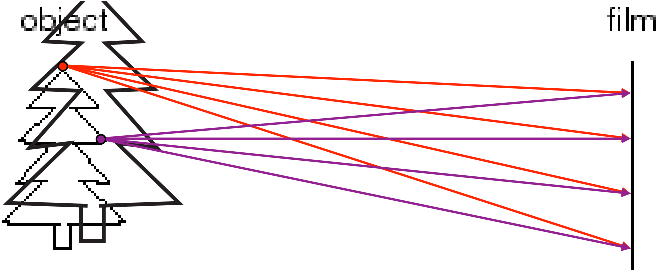
[Source: N. Snavely]

360 Panorama

[Source: N. Snavely]

Let's look at cameras

Image Formation

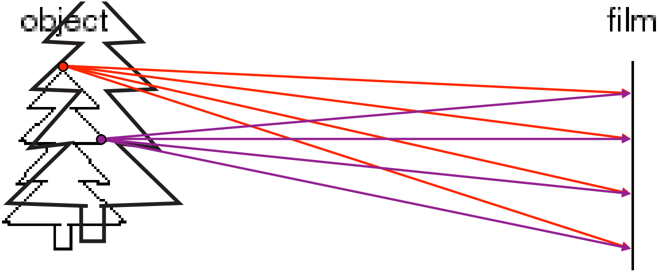


Lets design a camera

- Idea 1: put a piece of film in front of an object
- Do we get a reasonable image?

[Source: N. Snavely]

Image Formation

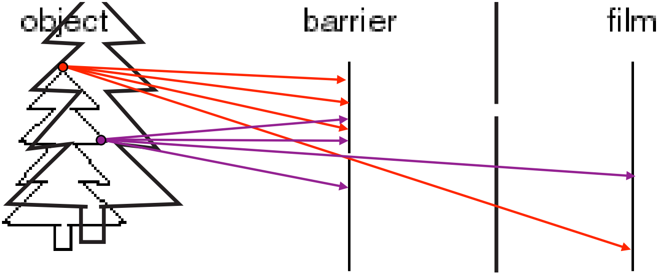


Lets design a camera

- Idea 1: put a piece of film in front of an object
- Do we get a reasonable image?

[Source: N. Snavely]

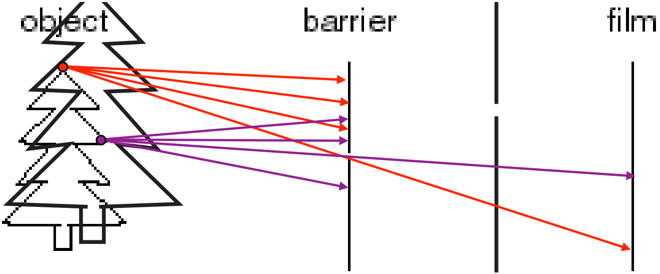
Pinhole Camera



Add a barrier to block off most of the rays

- This reduces blurring
- The opening known as the aperture

Pinhole Camera

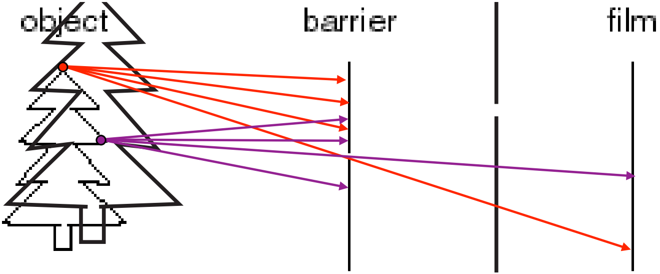


Add a barrier to block off most of the rays

- This reduces blurring
- The opening known as the aperture
- How does this transform the image?

[Source: N. Snavely]

Pinhole Camera

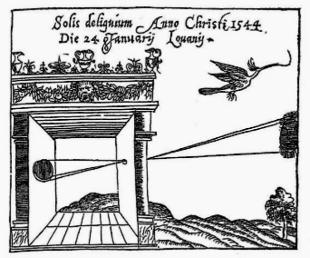


Add a barrier to block off most of the rays

- This reduces blurring
- The opening known as the aperture
- How does this transform the image?

[Source: N. Snavely]

Camera Obscura

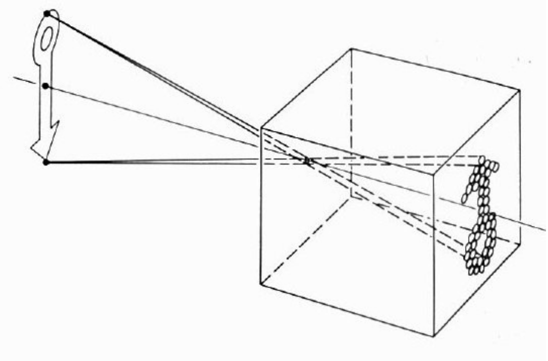


Gemma Frisius, 1558

- Basic principle known to Mozi (470-390 BC), Aristotle (384-322 BC)
- Drawing aid for artists: described by Leonardo da Vinci (1452-1519)

[Source: A. Efros]

Camera Obscura



[Source: N. Snavely]

Home made pinhole camera

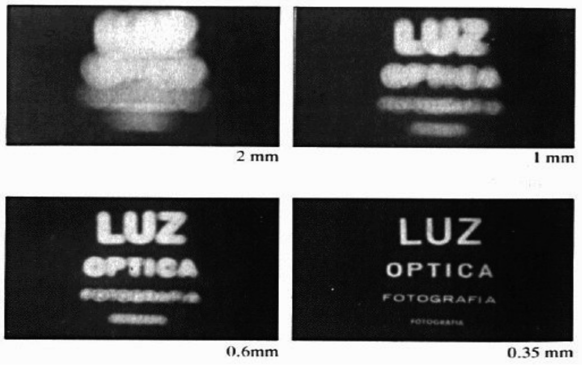
Why so blurry?

Slide by A. Efros

http://www.debevec.org/Pinhole/

[Source: N. Snavely]

Shrinking the aperture

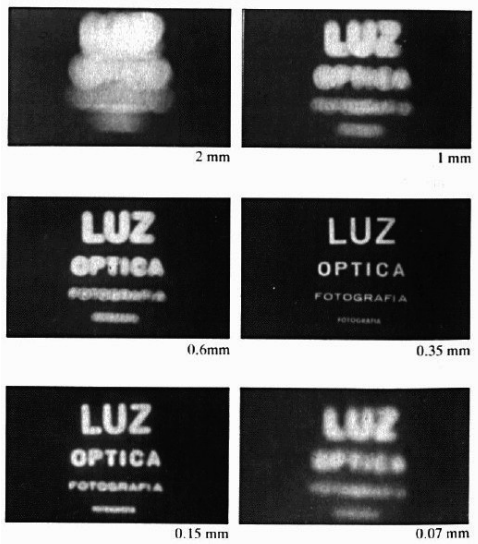


Why not make the aperture as small as possible?

- Less light gets through
- Diffraction effects...

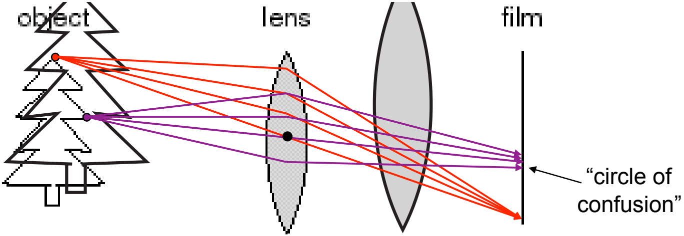
[Source: N. Snavely]

Shrinking the aperture



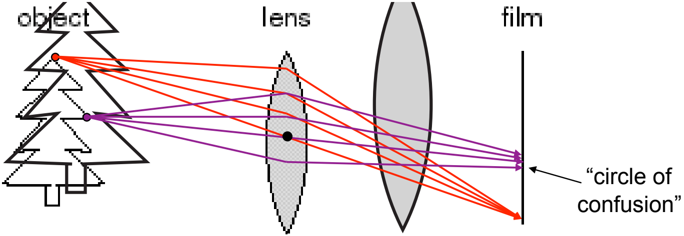
[Source: N. Snavely]

Adding a lens



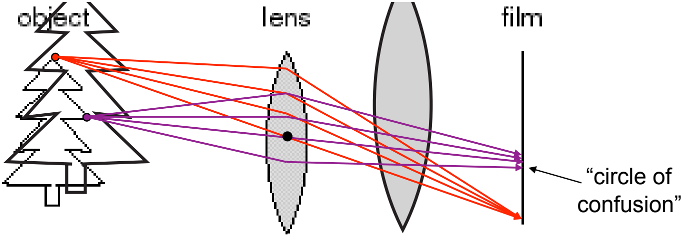
- A lens focuses light onto the film
- There is a specific distance at which objects are **in focus**

Adding a lens



- A lens focuses light onto the film
- There is a specific distance at which objects are **in focus**
- Other points project to a **circle of confusion** in the image

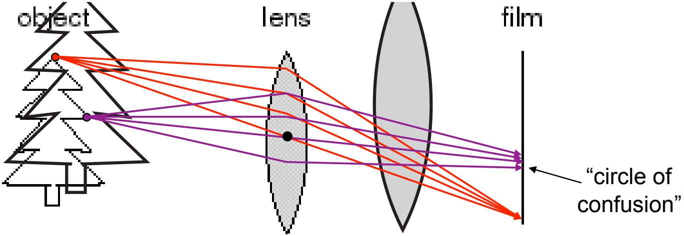
Adding a lens



- A lens focuses light onto the film
- There is a specific distance at which objects are **in focus**
- Other points project to a **circle of confusion** in the image
- Changing the shape of the lens changes this distance

[Source: N. Snavely]

Adding a lens



- A lens focuses light onto the film
- There is a specific distance at which objects are **in focus**
- Other points project to a **circle of confusion** in the image
- Changing the shape of the lens changes this distance

[Source: N. Snavely]

Projection

[Source: N. Snavely]

Projection

[Source: N. Snavely]

3D to 2D projections

- How are 3D primitives projected onto the image plane?
- We can do this using a linear 3D to 2D projection matrix

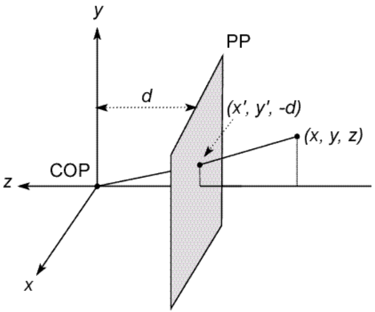
3D to 2D projections

- How are 3D primitives projected onto the image plane?
- We can do this using a linear 3D to 2D projection matrix
- Different types, the most commonly used:
    - Perspective
    - Orthography

3D to 2D projections

- How are 3D primitives projected onto the image plane?
- We can do this using a linear 3D to 2D projection matrix
- Different types, the most commonly used:
  - Perspective
  - Orthography

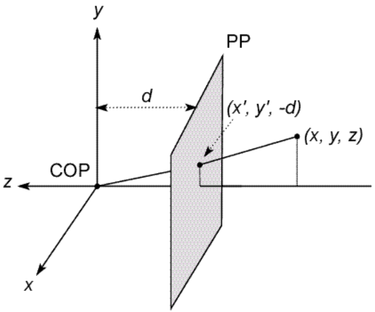
Modeling projection



The **coordinate system**

- We will use the pinhole model as an approximation
- Put the **optical center** (Center Of Projection) at the origin

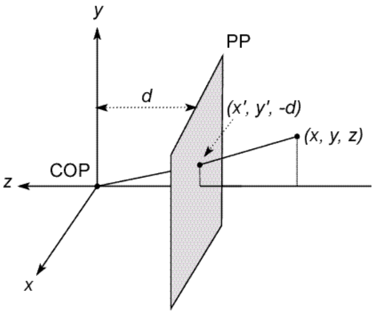
Modeling projection



The **coordinate system**

- We will use the pinhole model as an approximation
- Put the **optical center** (Center Of Projection) at the origin
- Put the **image plane** (Projection Plane) in front of the COP. Why?

Modeling projection

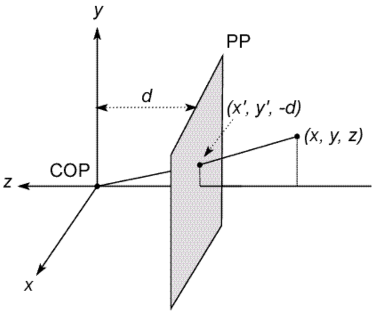


The **coordinate system**

- We will use the pinhole model as an approximation
- Put the **optical center** (Center Of Projection) at the origin
- Put the **image plane** (Projection Plane) in front of the COP. Why?
- The camera looks down the negative z axis, for right-handed-coordinates

[Source: N. Snavely]

Modeling projection

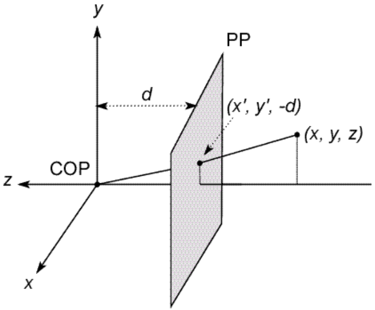


The **coordinate system**

- We will use the pinhole model as an approximation
- Put the **optical center** (Center Of Projection) at the origin
- Put the **image plane** (Projection Plane) in front of the COP. Why?
- The camera looks down the negative z axis, for right-handed-coordinates

[Source: N. Snavely]

Modeling projection

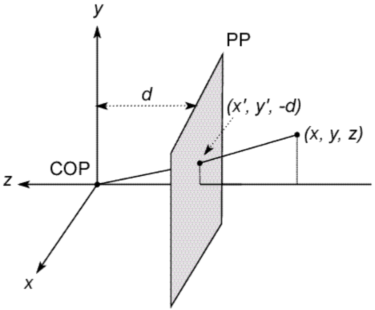


Projection Equations

- Compute intersection with PP of ray from (x,y,z) to COP. How?
- Derived using similar triangles

$$(x, y, z) \to (-d\frac{x}{z}, -d\frac{y}{z}, -d)$$

Modeling projection



Projection Equations

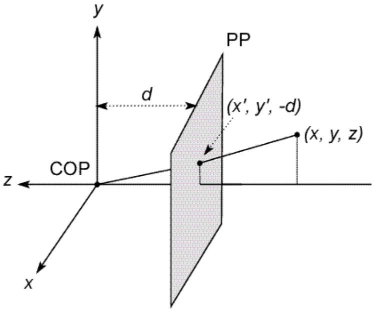
- Compute intersection with PP of ray from (x,y,z) to COP. How?
- Derived using similar triangles

$$(x, y, z) \rightarrow (-d\frac{x}{z}, -d\frac{y}{z}, -d)$$

- Get the projection by throwing the last coordinate

[Source: N. Snavely]

Modeling projection



Projection Equations

- Compute intersection with PP of ray from (x,y,z) to COP. How?
- Derived using similar triangles

$$(x, y, z) \rightarrow (-d\frac{x}{z}, -d\frac{y}{z}, -d)$$

- Get the projection by throwing the last coordinate

[Source: N. Snavely]

Modeling Projection

- This is NOT a linear transformation as a division by z is non-linear

Homogeneous coordinates to the rescue!

$$(x, y) \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \qquad (x, y, z) \Rightarrow \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

homogeneous image coordinates

homogeneous scene coordinates

Converting *from* homogeneous coordinates

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} \Rightarrow (x/w, y/w) \qquad \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \Rightarrow (x/w, y/w, z/w)$$

[Source: N. Snavely]

Perspective Projection

- Projection is a matrix multiply using homogeneous coordinates

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1/d & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ -z/d \end{bmatrix} \Rightarrow (-d\frac{x}{z}, \quad -d\frac{y}{z})$$

divide by third coordinate

- This is known as **perspective projection**

Perspective Projection

- Projection is a matrix multiply using homogeneous coordinates

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1/d & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ -z/d \end{bmatrix} \Rightarrow (-d\frac{x}{z}, \quad -d\frac{y}{z})$$

divide by third coordinate

- This is known as **perspective projection**
- The matrix is called the **projection matrix**

Perspective Projection

- Projection is a matrix multiply using homogeneous coordinates

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1/d & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ -z/d \end{bmatrix} \Rightarrow (-d\frac{x}{z}, \quad -d\frac{y}{z})$$

divide by third coordinate

- This is known as **perspective projection**
- The matrix is called the **projection matrix**
- Can also represent as a 4x4 matrix

[Source: N. Snavely]

Perspective Projection

- Projection is a matrix multiply using homogeneous coordinates

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1/d & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ -z/d \end{bmatrix} \Rightarrow (-d\frac{x}{z}, \quad -d\frac{y}{z})$$

divide by third coordinate

- This is known as **perspective projection**
- The matrix is called the **projection matrix**
- Can also represent as a 4x4 matrix

[Source: N. Snavely]

Perspective Projection

- How does scaling the projection matrix change the transformation?

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1/d & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ -z/d \end{bmatrix} \Rightarrow (-d\frac{x}{z}, \quad -d\frac{y}{z})$$

$$\begin{bmatrix} -d & 0 & 0 & 0 \\ 0 & -d & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} -dx \\ -dy \\ z \end{bmatrix} \Rightarrow (-d\frac{x}{z}, \quad -d\frac{y}{z})$$

- It is not possible to recover the distance of the 3D point from the image.

[Source: N. Snavely]

Perspective Projection

- How does scaling the projection matrix change the transformation?

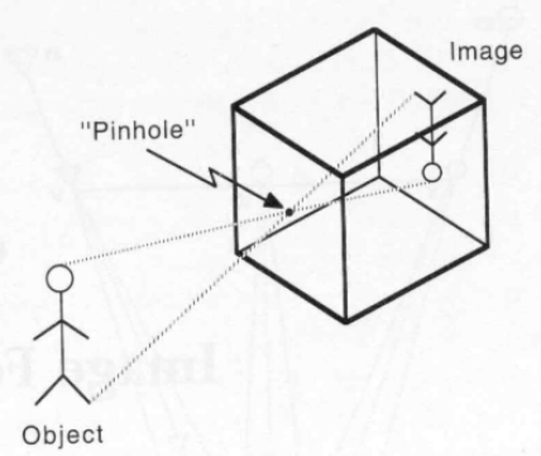
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1/d & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ -z/d \end{bmatrix} \Rightarrow (-d\frac{x}{z}, \quad -d\frac{y}{z})$$

$$\begin{bmatrix} -d & 0 & 0 & 0 \\ 0 & -d & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} -dx \\ -dy \\ z \end{bmatrix} \Rightarrow (-d\frac{x}{z}, \quad -d\frac{y}{z})$$

- It is not possible to recover the distance of the 3D point from the image.

[Source: N. Snavely]

Perspective Projection

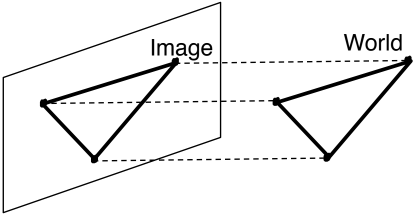


Perspective Projection

[Source: S. Seitz]

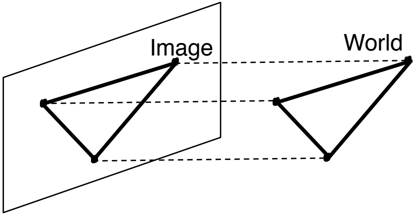
Orthographic Projection

- Requires no division and simply drops the z coordinate.
- Special case of perspective projection where the distance from the COP to the PP is infinity



Orthographic Projection

- Requires no division and simply drops the *z* coordinate.
- Special case of perspective projection where the distance from the COP to the PP is infinity

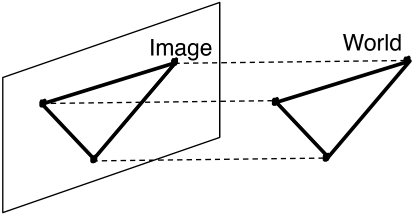


- Let **p** be a 3D point and **x** a 2D point, we can write

$$\mathbf{x} = \begin{bmatrix} \mathbf{I}_{2\times 2} & 0_{2\times 1} \end{bmatrix} \mathbf{p}$$

Orthographic Projection

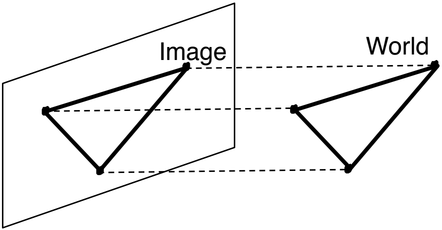
- Requires no division and simply drops the z coordinate.
- Special case of perspective projection where the distance from the COP to the PP is infinity



- Let $\mathbf{p}$ be a 3D point and $\mathbf{x}$ a 2D point, we can write

$$\mathbf{x} = \begin{bmatrix} \mathbf{I}_{2\times 2} & 0_{2\times 1} \end{bmatrix} \mathbf{p}$$

More on Orthographic Projection



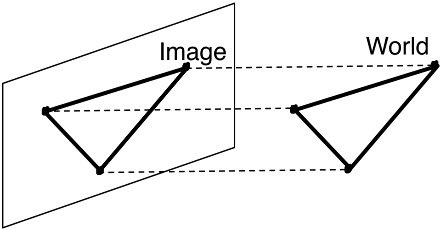
- Let **p** be a 3D point and **x** a 2D point, we can write

$$\mathbf{x} = \begin{bmatrix} \mathbf{I}_{2\times 2} & 0_{2\times 1} \end{bmatrix} \mathbf{p}$$

- It can also be written in homogeneous coordinates

$$\hat{\mathbf{x}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \hat{\mathbf{p}}$$

More on Orthographic Projection



- Let $\mathbf{p}$ be a 3D point and $\mathbf{x}$ a 2D point, we can write

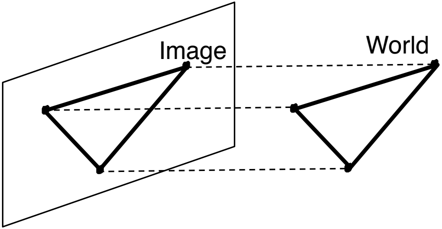
$$\mathbf{x} = \begin{bmatrix} \mathbf{I}_{2\times 2} & 0_{2\times 1} \end{bmatrix} \mathbf{p}$$

- It can also be written in homogeneous coordinates

$$\hat{\mathbf{x}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \hat{\mathbf{p}}$$

- Is an approximate model for long focal length lenses and objects whose depth is shallow relative to their distance to the camera.

More on Orthographic Projection



- Let $\mathbf{p}$ be a 3D point and $\mathbf{x}$ a 2D point, we can write

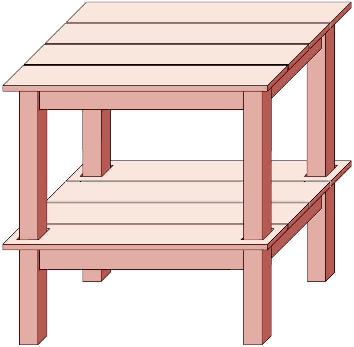
$$\mathbf{x} = \begin{bmatrix} \mathbf{I}_{2\times 2} & 0_{2\times 1} \end{bmatrix} \mathbf{p}$$

- It can also be written in homogeneous coordinates

$$\hat{\mathbf{x}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \hat{\mathbf{p}}$$

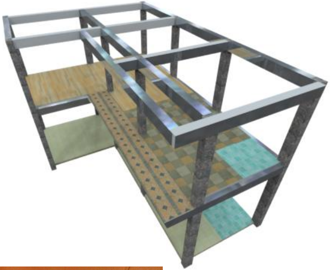
- Is an approximate model for long focal length lenses and objects whose depth is shallow relative to their distance to the camera.

Orthographic Projection



[Source: N. Snavely]

Perspective Projection



[Source: N. Snavely]

Variants of Orthographic

- In practice, world coordinates need to be scaled to fit onto an image sensor (e.g., transform to pixels)
- This is why **scaled orthographic**, also called **weak perspective** is more commonly used

$$\mathbf{x} = \begin{bmatrix} s\mathbf{I}_{2\times 2} & 0_{2\times 1} \end{bmatrix}$$

Variants of Orthographic

- In practice, world coordinates need to be scaled to fit onto an image sensor (e.g., transform to pixels)
- This is why **scaled orthographic**, also called **weak perspective** is more commonly used

$$\mathbf{x} = \begin{bmatrix} s\mathbf{I}_{2\times 2} & 0_{2\times 1} \end{bmatrix}$$

- And in homogeneous

$$\hat{\mathbf{x}} = \begin{bmatrix} s & 0 & 0 & 0 \\ 0 & s & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \hat{\mathbf{p}}$$

Variants of Orthographic

- In practice, world coordinates need to be scaled to fit onto an image sensor (e.g., transform to pixels)
- This is why **scaled orthographic**, also called **weak perspective** is more commonly used

$$\mathbf{x} = \begin{bmatrix} s\mathbf{I}_{2\times 2} & 0_{2\times 1} \end{bmatrix}$$

- And in homogeneous

$$\hat{\mathbf{x}} = \begin{bmatrix} s & 0 & 0 & 0 \\ 0 & s & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \hat{\mathbf{p}}$$

- This is equivalent to first projecting the world points onto a local fronto-parallel image plane and then scaling this image using regular perspective projection

Variants of Orthographic

- In practice, world coordinates need to be scaled to fit onto an image sensor (e.g., transform to pixels)
- This is why **scaled orthographic**, also called **weak perspective** is more commonly used

$$\mathbf{x} = \begin{bmatrix} s\mathbf{I}_{2\times 2} & 0_{2\times 1} \end{bmatrix}$$

- And in homogeneous

$$\hat{\mathbf{x}} = \begin{bmatrix} s & 0 & 0 & 0 \\ 0 & s & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \hat{\mathbf{p}}$$

- This is equivalent to first projecting the world points onto a local fronto-parallel image plane and then scaling this image using regular perspective projection
- Is a popular model for reconstructing the 3D shape of objects far away from the camera

Variants of Orthographic

- In practice, world coordinates need to be scaled to fit onto an image sensor (e.g., transform to pixels)
- This is why **scaled orthographic**, also called **weak perspective** is more commonly used

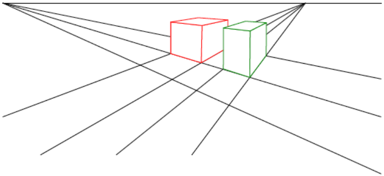
$$\mathbf{x} = \begin{bmatrix} s\mathbf{I}_{2\times 2} & 0_{2\times 1} \end{bmatrix}$$

- And in homogeneous

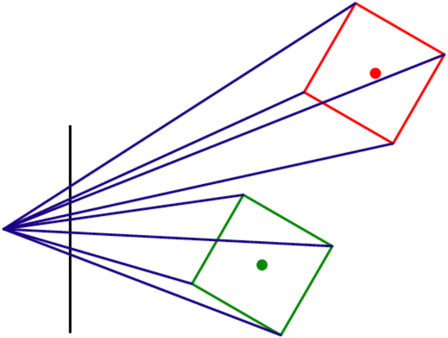
$$\hat{\mathbf{x}} = \begin{bmatrix} s & 0 & 0 & 0 \\ 0 & s & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \hat{\mathbf{p}}$$

- This is equivalent to first projecting the world points onto a local fronto-parallel image plane and then scaling this image using regular perspective projection
- Is a popular model for reconstructing the 3D shape of objects far away from the camera

Variants of Orthographic

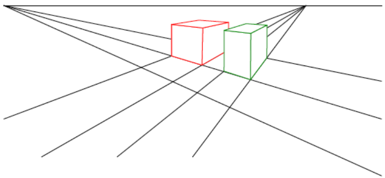


3D World

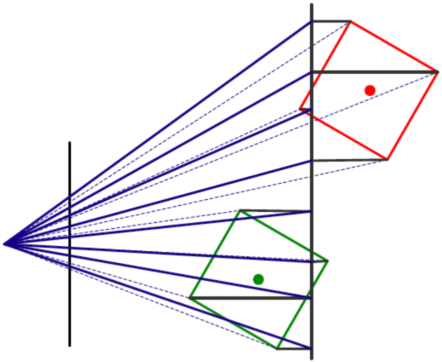


Perspective Projection

Variants of Orthographic



3D World



Orthographic Projection

Variants of Orthographic

- **Affine projection**, also called **paraperspective**
- Object points are again first projected onto a local reference parallel to the image plane.

Variants of Orthographic

- **Affine projection**, also called **paraperspective**
- Object points are again first projected onto a local reference parallel to the image plane.
- However, rather than being projected orthogonally to this plane, they are projected parallel to the line of sight to the object center

Variants of Orthographic

- **Affine projection**, also called **paraperspective**
- Object points are again first projected onto a local reference parallel to the image plane.
- However, rather than being projected orthogonally to this plane, they are projected parallel to the line of sight to the object center
- This is follow by projection onto the final image plane, which amounts to a scaling

Variants of Orthographic

- **Affine projection**, also called **paraperspective**
- Object points are again first projected onto a local reference parallel to the image plane.
- However, rather than being projected orthogonally to this plane, they are projected parallel to the line of sight to the object center
- This is follow by projection onto the final image plane, which amounts to a scaling
- Thus, it is affine, and in homogeneous coordinates

$$\hat{\mathbf{x}} = \begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ 0 & 0 & 0 & 1 \end{bmatrix} \hat{\mathbf{p}}$$

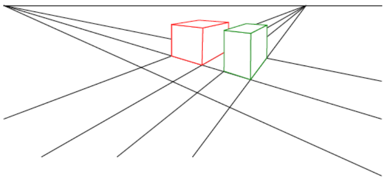
[Source: N. Snavely]

Variants of Orthographic

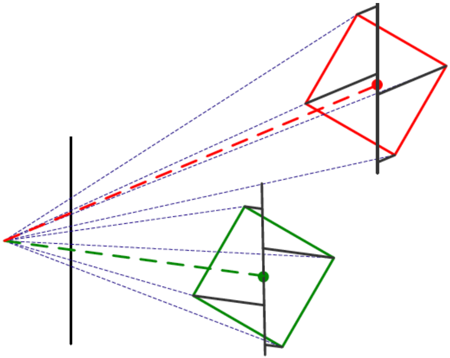
- **Affine projection**, also called **paraperspective**
- Object points are again first projected onto a local reference parallel to the image plane.
- However, rather than being projected orthogonally to this plane, they are projected parallel to the line of sight to the object center
- This is follow by projection onto the final image plane, which amounts to a scaling
- Thus, it is affine, and in homogeneous coordinates

$$\hat{\mathbf{x}} = \begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ 0 & 0 & 0 & 1 \end{bmatrix} \hat{\mathbf{p}}$$

[Source: N. Snavely]



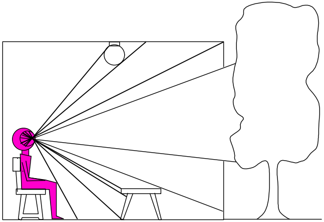
3D World



Paraperspective

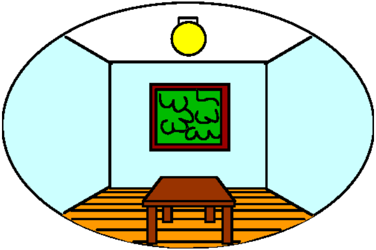
Dimensionality Reduction Machine (3D to 2D)

3D world



Point of observation

2D image



What have we lost?

- Angles
- Distances (lengths)

Slide by A. Efros

Projection properties

- **Many-to-one**: any points along same ray map to same point in image
- Points → points

Projection properties

- **Many-to-one**: any points along same ray map to same point in image
- Points $\rightarrow$ points
- Lines $\rightarrow$ lines

Projection properties

- **Many-to-one**: any points along same ray map to same point in image
- Points $\rightarrow$ points
- Lines $\rightarrow$ lines
- But line through focal point projects to a point. Why?

Projection properties

- **Many-to-one**: any points along same ray map to same point in image
- Points $\rightarrow$ points
- Lines $\rightarrow$ lines
- But line through focal point projects to a point. Why?
- Planes $\rightarrow$ planes

Projection properties

- **Many-to-one**: any points along same ray map to same point in image
- Points $\rightarrow$ points
- Lines $\rightarrow$ lines
- But line through focal point projects to a point. Why?
- Planes $\rightarrow$ planes
- But plane through focal point projects to line. Why?

[Source: N. Snavely]

Projection properties

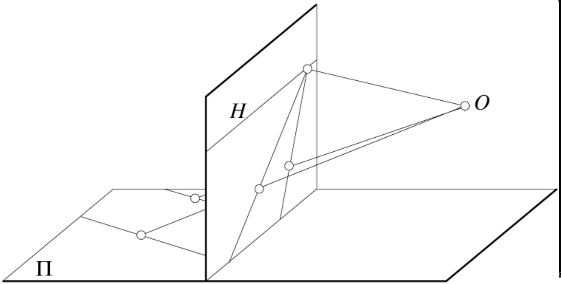
- **Many-to-one**: any points along same ray map to same point in image
- Points $\rightarrow$ points
- Lines $\rightarrow$ lines
- But line through focal point projects to a point. Why?
- Planes $\rightarrow$ planes
- But plane through focal point projects to line. Why?

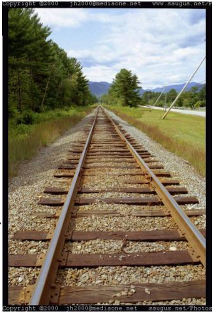
[Source: N. Snavely]

Projection properties

Parallel lines converge at a vanishing point

- Each direction in space has its own vanishing point
- But parallels parallel to the image plane remain parallel





[Source: N. Snavely]

Camera Parameters

How many numbers do we need to describe a camera?

- We need to describe its **pose in the world**

Camera Parameters

How many numbers do we need to describe a camera?

- We need to describe its **pose in the world**
- We need to describe its **internal parameters**

Camera Parameters

How many numbers do we need to describe a camera?

- We need to describe its **pose in the world**
- We need to describe its **internal parameters**
- How many then?

[Source: N. Snavely]

Camera Parameters

How many numbers do we need to describe a camera?

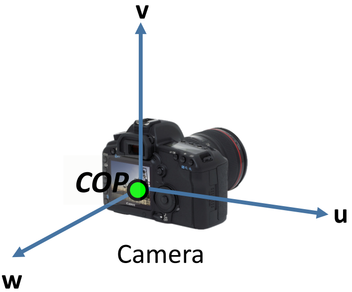
- We need to describe its **pose in the world**
- We need to describe its **internal parameters**
- How many then?

[Source: N. Snavely]

Which coordinate system to use?

Two important coordinate systems:

- World coordinate system
- Camera coordinate system



[Source: N. Snavely]

Camera parameters

To project a point (x, y, z) in world coordinates into a camera

- Transform (x, y, z) into camera coordinates, we need to know

Camera parameters

To project a point (x, y, z) in world coordinates into a camera

- Transform (x, y, z) into camera coordinates, we need to know
  - Camera **position** (in world coordinates)

Camera parameters

To project a point (x, y, z) in world coordinates into a camera

- Transform (x, y, z) into camera coordinates, we need to know
  - Camera **position** (in world coordinates)
  - Camera **orientation** (in world coordinates)

Camera parameters

To project a point (x, y, z) in world coordinates into a camera

- Transform (x, y, z) into camera coordinates, we need to know
  - Camera **position** (in world coordinates)
  - Camera **orientation** (in world coordinates)
- We then project into the image plane
  - Need to know **camera intrinsics**

Camera parameters

To project a point (x, y, z) in world coordinates into a camera

- Transform (x, y, z) into camera coordinates, we need to know
  - Camera **position** (in world coordinates)
  - Camera **orientation** (in world coordinates)
- We then project into the image plane
  - Need to know **camera intrinsics**
- These can all be described with matrices

[Source: N. Snavely]

Camera parameters

To project a point (x, y, z) in world coordinates into a camera

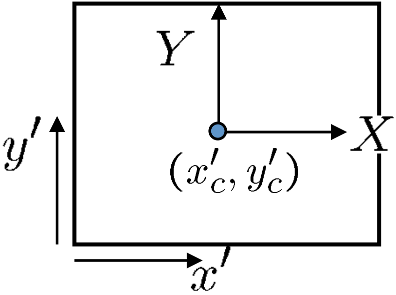
- Transform (x, y, z) into camera coordinates, we need to know
  - Camera **position** (in world coordinates)
  - Camera **orientation** (in world coordinates)
- We then project into the image plane
  - Need to know **camera intrinsics**
- These can all be described with matrices

[Source: N. Snavely]

More on camera parameters

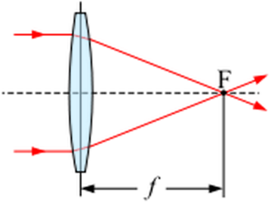
A camera is described by several parameters

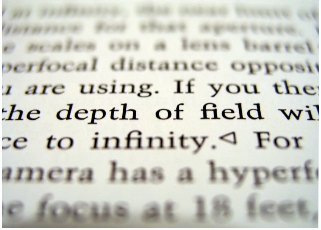
- Translation T of the optical center from the origin of world coords
- Rotation R of the image plane
- Focal length f, principle point (x'_c, y'_c), pixel size (s_x, s_y)
- Which parameters are **extrinsics** and which **intrinsics**?



Focal Length

- Distance over which initially collimated rays (i.e., parallel) are brought to a focus.





Focal Length

- Can be thought of as **zoom**
- Related to the **field of view**

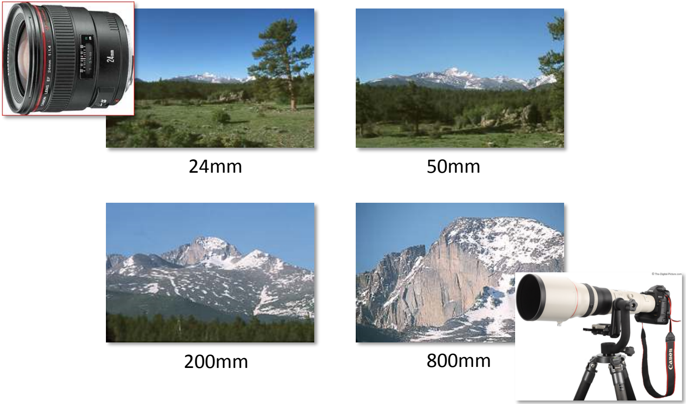


Figure: Image from N. Snavely

Projection Equations

- The projection matrix models the cumulative effect of all intrinsic and extrinsic parameters

$$\mathbf{X} = \begin{bmatrix} ax \\ ay \\ a \end{bmatrix} = \mathbf{P} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

- It can be computed as

$$\mathbf{P} = \underbrace{\begin{bmatrix} -f \cdot s_x & 0 & x'_c \\ 0 & -f \cdot s_y & y'_c \\ 0 & 0 & 1 \end{bmatrix}}_{\text{intrinsics}} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\text{projection}} \underbrace{\begin{bmatrix} \mathbf{R}_{3\times 3} & 0_{3\times 1} \\ 0_{1\times 3} & 1 \end{bmatrix}}_{\text{rotation}} \underbrace{\begin{bmatrix} \mathbf{I}_{3\times 3} & \mathbf{T}_{3\times 1} \\ 0_{1\times 3} & 1 \end{bmatrix}}_{\text{translation}}$$

Projection Equations

- The projection matrix models the cumulative effect of all intrinsic and extrinsic parameters

$$\mathbf{X} = \begin{bmatrix} ax \\ ay \\ a \end{bmatrix} = \mathbf{P} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

- It can be computed as

$$\mathbf{P} = \underbrace{\begin{bmatrix} -f \cdot s_x & 0 & x'_c \\ 0 & -f \cdot s_y & y'_c \\ 0 & 0 & 1 \end{bmatrix}}_{\text{intrinsics}} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\text{projection}} \underbrace{\begin{bmatrix} \mathbf{R}_{3\times3} & 0_{3\times1} \\ 0_{1\times3} & 1 \end{bmatrix}}_{\text{rotation}} \underbrace{\begin{bmatrix} \mathbf{I}_{3\times3} & \mathbf{T}_{3\times1} \\ 0_{1\times3} & 1 \end{bmatrix}}_{\text{translation}}$$

- No standard definition of intrinsics and extrinsics

Projection Equations

- The projection matrix models the cumulative effect of all intrinsic and extrinsic parameters

$$\mathbf{X} = \begin{bmatrix} ax \\ ay \\ a \end{bmatrix} = \mathbf{P} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

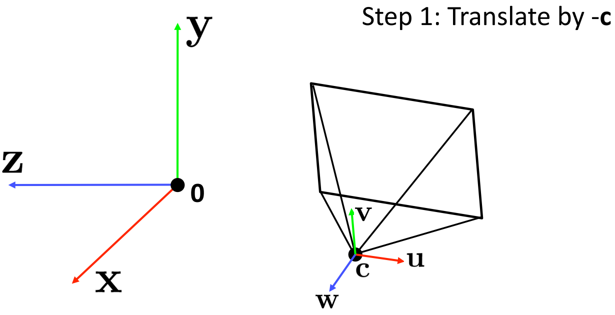
- It can be computed as

$$\mathbf{P} = \underbrace{\begin{bmatrix} -f \cdot s_x & 0 & x'_c \\ 0 & -f \cdot s_y & y'_c \\ 0 & 0 & 1 \end{bmatrix}}_{\text{intrinsics}} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\text{projection}} \underbrace{\begin{bmatrix} \mathbf{R}_{3\times3} & 0_{3\times1} \\ 0_{1\times3} & 1 \end{bmatrix}}_{\text{rotation}} \underbrace{\begin{bmatrix} \mathbf{I}_{3\times3} & \mathbf{T}_{3\times1} \\ 0_{1\times3} & 1 \end{bmatrix}}_{\text{translation}}$$

- No standard definition of intrinsics and extrinsics

Extrinsics

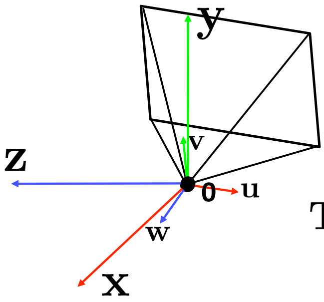
How do we get the camera to canonical form?



[Source: N. Snavely]

Extrinsics

How do we get the camera to canonical form?



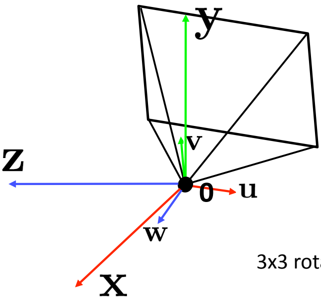
Step 1: Translate by -**c**

How do we represent translation as a matrix multiplication?

$$\mathbf{T} = \begin{bmatrix} \mathbf{I}_{3\times 3} & -\mathbf{c} \\ 0 \quad 0 \quad 0 & 1 \end{bmatrix}$$

[Source: N. Snavely]

How do we get the camera to canonical form?



Step 1: Translate by -**c**

Step 2: Rotate by **R**

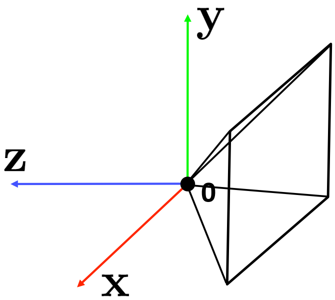
$$\mathbf{R} = \begin{bmatrix} \mathbf{u}^T \\ \mathbf{v}^T \\ \mathbf{w}^T \end{bmatrix}$$

3x3 rotation matrix

[Source: N. Snavely]

Extrinsics

How do we get the camera to canonical form?



Step 1: Translate by **-c**

Step 2: Rotate by **R**

$$\mathbf{R} = \begin{bmatrix} \mathbf{u}^T \\ \mathbf{v}^T \\ \mathbf{w}^T \end{bmatrix}$$

[Source: N. Snavely]

Perspective Projection

$$\underbrace{\begin{bmatrix} -f & 0 & 0 \\ 0 & -f & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{K}} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

(intrinsics) (converts from 3D rays in camera coordinate system to pixel coordinates)

in general, $\mathbf{K} = \begin{bmatrix} -f & s & c_x \\ 0 & -\alpha f & c_y \\ 0 & 0 & 1 \end{bmatrix}$ (upper triangular matrix)

α : **aspect ratio** (1 unless pixels are not square)

s : **skew** (0 unless pixels are shaped like rhombi/parallelograms)

: **principal point** ((0,0) unless optical axis doesn't intersect projection plane at origin)

- Simplifications used in practice

[Source: N. Snavely]

Camera matrix

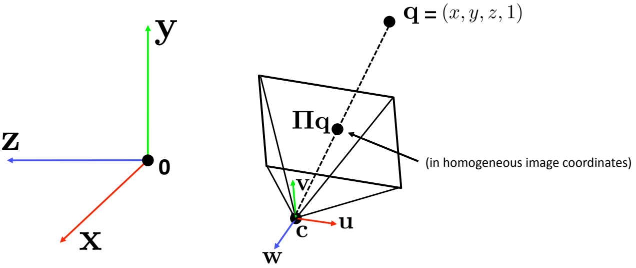
- The projection matrix is defined as

$$\mathbf{P} = \underbrace{\mathbf{K}}_{\text{intrinsics}} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\text{projection}} \underbrace{\underbrace{\begin{bmatrix} \mathbf{R}_{3\times3} & 0_{3\times1} \\ 0_{1\times3} & 1 \end{bmatrix}}_{\text{rotation}} \underbrace{\begin{bmatrix} \mathbf{I}_{3\times3} & \mathbf{T}_{3\times3} \\ 0_{1\times3} & 1 \end{bmatrix}}_{\text{translation}}}_{\begin{bmatrix} \mathbf{R} & -\mathbf{Rc} \end{bmatrix}}$$

- More compactly

$$\mathbf{P} = \mathbf{K} \begin{bmatrix} \mathbf{R} & -\mathbf{Rc} \end{bmatrix} = \mathbf{K} \begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix}$$

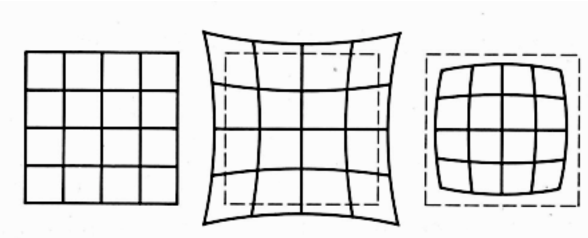
Camera matrix



[Source: N. Snavely]

Radial Distorsion

- Caused by imperfect lenses
- Deviations are most noticeable for rays that pass through the edge of the lens



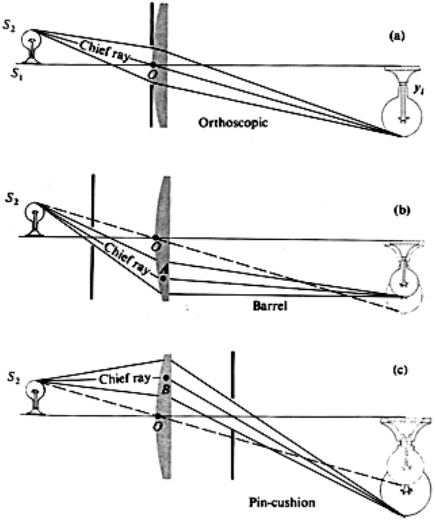
No distortion Pin cushion Barrel

[Source: N. Snavely]

Correcting Radial Distorsion

from Helmut Dersch

Distorsion



[Source: N. Snavely]

Modeling Distorsion

- Project point to normalized image coordinates

$$
\begin{aligned}
x_n &= \frac{x}{z} \\
y_n &= \frac{y}{z}
\end{aligned}
$$

- Apply radial distortion

$$
\begin{aligned}
r^2 &= x_n^2 + y_n^2 \\
x_d &= x_n(1 + \kappa_1 r^2 + \kappa_2 r^4) \\
y_d &= y_n(1 + \kappa_1 r^2 + \kappa_2 r^4)
\end{aligned}
$$

Modeling Distorsion

- Project point to normalized image coordinates

$$\begin{aligned} x_n &= \frac{x}{z} \\ y_n &= \frac{y}{z} \end{aligned}$$

- Apply radial distorsion

$$\begin{aligned} r^2 &= x_n^2 + y_n^2 \\ x_d &= x_n(1 + \kappa_1 r^2 + \kappa_2 r^4) \\ y_d &= y_n(1 + \kappa_1 r^2 + \kappa_2 r^4) \end{aligned}$$

- Apply focal length and translate image center

$$\begin{aligned} x' &= f x_d + x_c \\ y' &= f y_d + y_c \end{aligned}$$

Modeling Distorsion

- Project point to normalized image coordinates

$$x_n = \frac{x}{z}$$
$$y_n = \frac{y}{z}$$

- Apply radial distorsion

$$r^2 = x_n^2 + y_n^2$$
$$x_d = x_n(1 + \kappa_1 r^2 + \kappa_2 r^4)$$
$$y_d = y_n(1 + \kappa_1 r^2 + \kappa_2 r^4)$$

- Apply focal length and translate image center

$$x' = f x_d + x_c$$
$$y' = f y_d + y_c$$

- To model lens distortion use above projection operation instead of standard projection matrix multiplication

Modeling Distorsion

- Project point to normalized image coordinates

$$x_n = \frac{x}{z}$$
$$y_n = \frac{y}{z}$$

- Apply radial distorsion

$$r^2 = x_n^2 + y_n^2$$
$$x_d = x_n(1 + \kappa_1 r^2 + \kappa_2 r^4)$$
$$y_d = y_n(1 + \kappa_1 r^2 + \kappa_2 r^4)$$

- Apply focal length and translate image center

$$x' = f x_d + x_c$$
$$y' = f y_d + y_c$$

- To model lens distortion use above projection operation instead of standard projection matrix multiplication

Next class ... more on panoramas