Multiple View Geometry

Last updated: 30-05-2020

Szeliski R., Computer Vision - Algorithms and Applications, Springer, 201, Ch. 12.

Solem, J. E., 2012. Programming Computer Vision with Python: Tools and algorithms for analyzing images. "O'Reilly Media, Inc.“, Ch. 5.

Dr. Zulfiqar Habib, Professor
http://vig.cuilahore.edu.pk

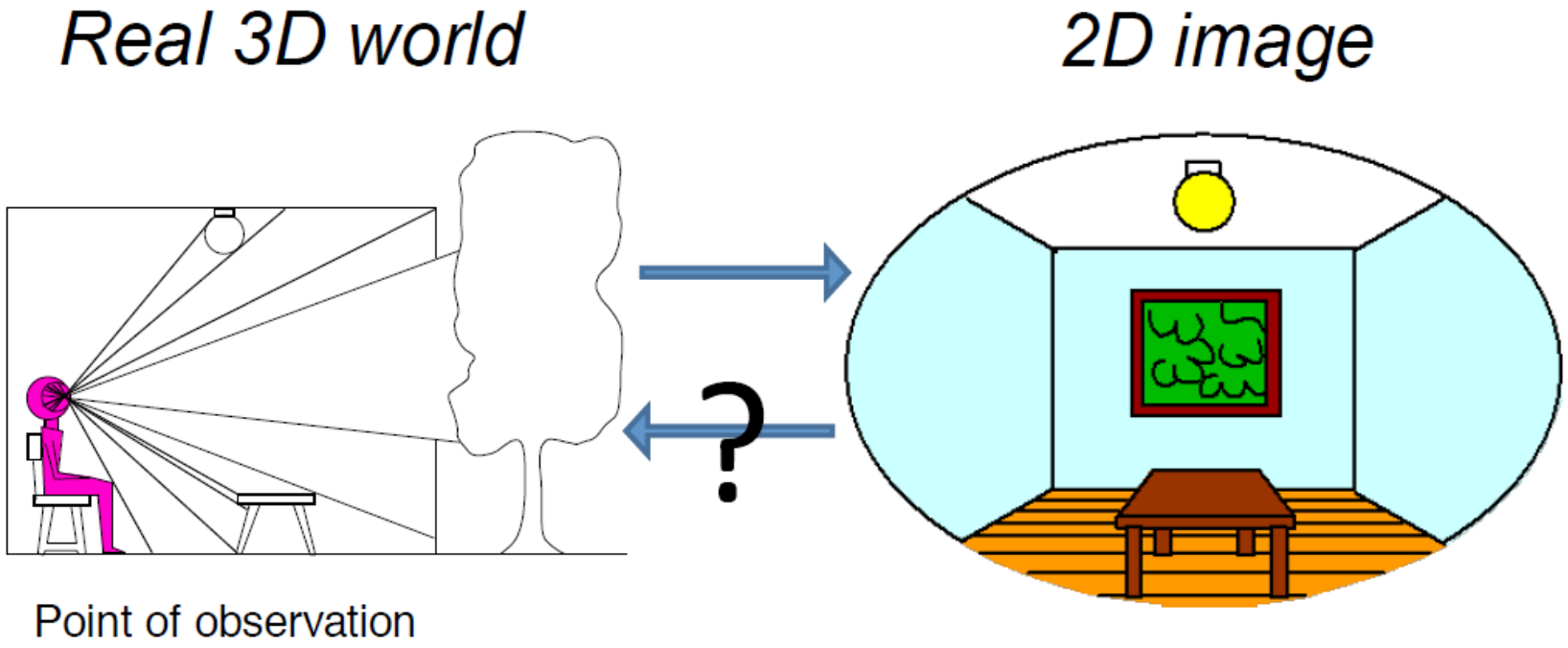
Topics

- Stereo vision
- Depth estimation
- Structure from motion (3D reconstruction)

Recovering 3D from Images

How can we automatically compute 3D geometry from images?

What cues in the image provide 3D information?



Introduction…

- Humans have the natural ability to perceive 3D world.
- When conventional cameras capture the pictures, the depth information is lost due to the projection of a scene onto the 2D image plane.

Introduction

- Recovering the 3D structure of a scene from a single image has become an important research problem in the present years due to its wide range of applications.

3D Printing

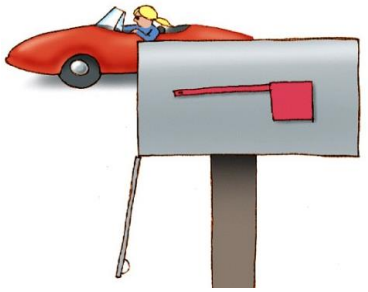
Robotic surgery

3D games/ movies

Self Driving cars / Robot navigation

Monocular Depth Cues…

- Interposition
  - When one object partly blocks your view of another, you perceive the partially blocked object as farther away

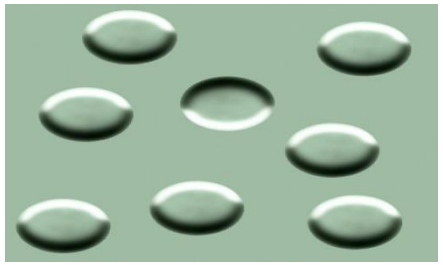


- Linear perspective
  - Parallel lines that are known to be the same distance apart appear to grow closer together, or converge, as they recede into the distance

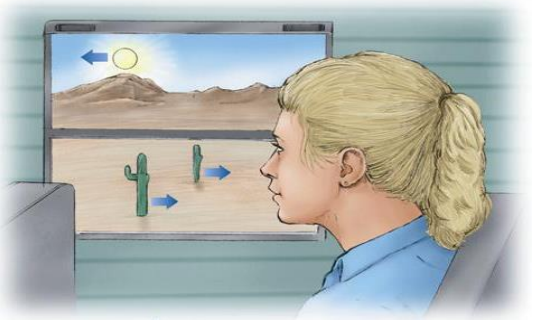
- Texture Gradients
  - The texture of objects tend to become smoother as the object gets farther away, suggesting that more detailed textured objects are closer.

Monocular Depth Cues

- Relative Size
  - Larger objects are perceived as being closer to the viewer, and smaller objects as being farther away



- Light and Shadow
  - When objects are darker or shaded they are perceived as farther away.
- Motion Parallax
  - When you ride in a moving vehicle and look at the side window, the objects you see appear outside move in opposite direction
  - Objects seem to be moving in different speeds-the ones that are closer to you seem to move faster, whereas objects far behind seem to move slower



Types of Existing Approaches

Manual: Currently used in high-quality cinematic 2D-to-3D conversion workflows.

Interactive (Semi-automatic): A combination of user-input and computer vision algorithms.

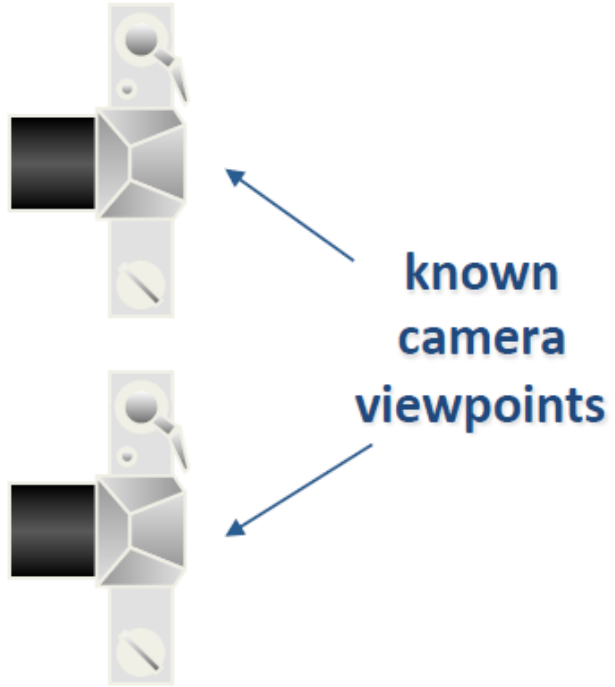
Automatic: Entirely based on computer vision algorithms or convolutional neural networks.

Stereo Reconstruction

The Stereo Problem

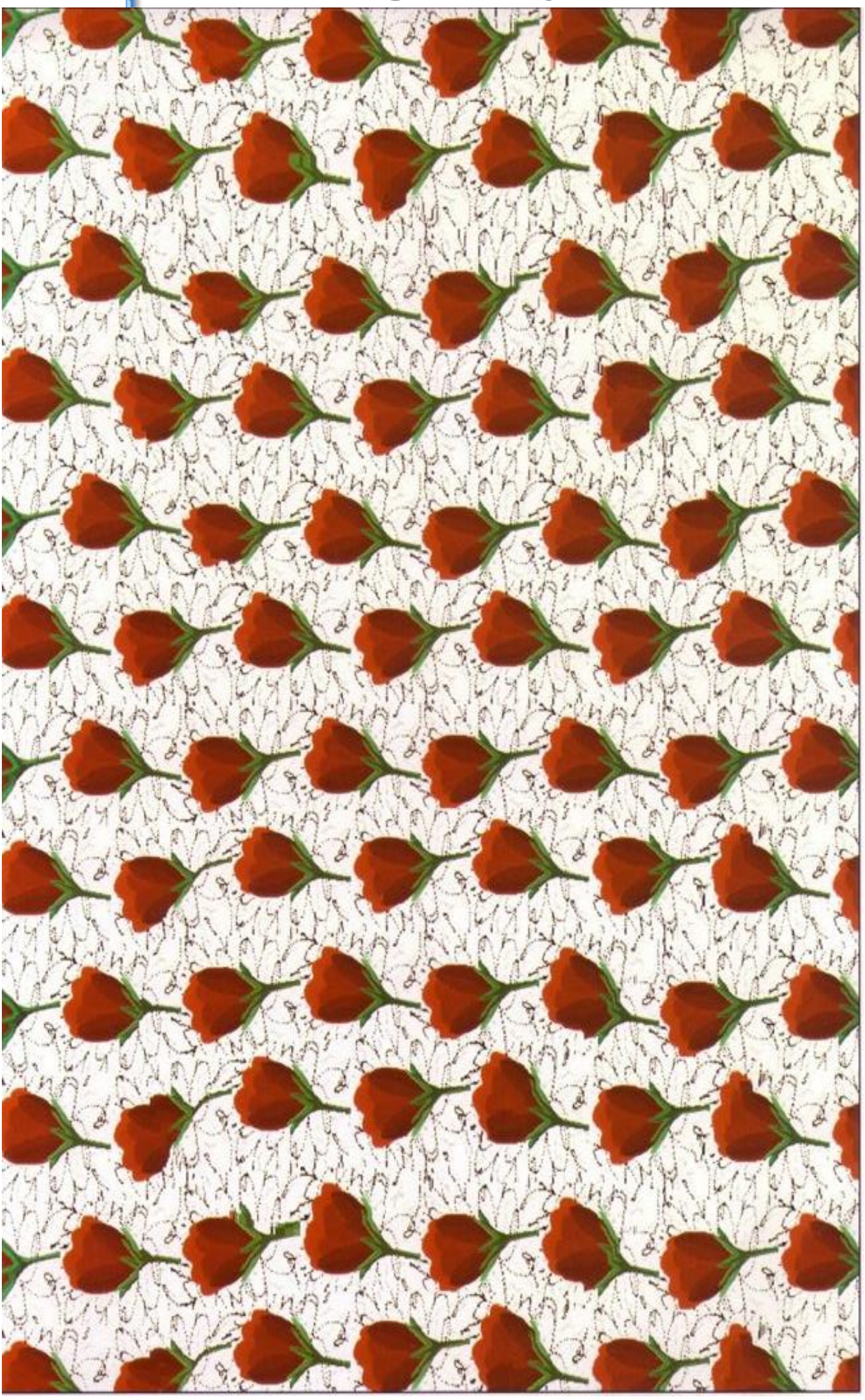
– Shape from two (or more) images

– Biological motivation

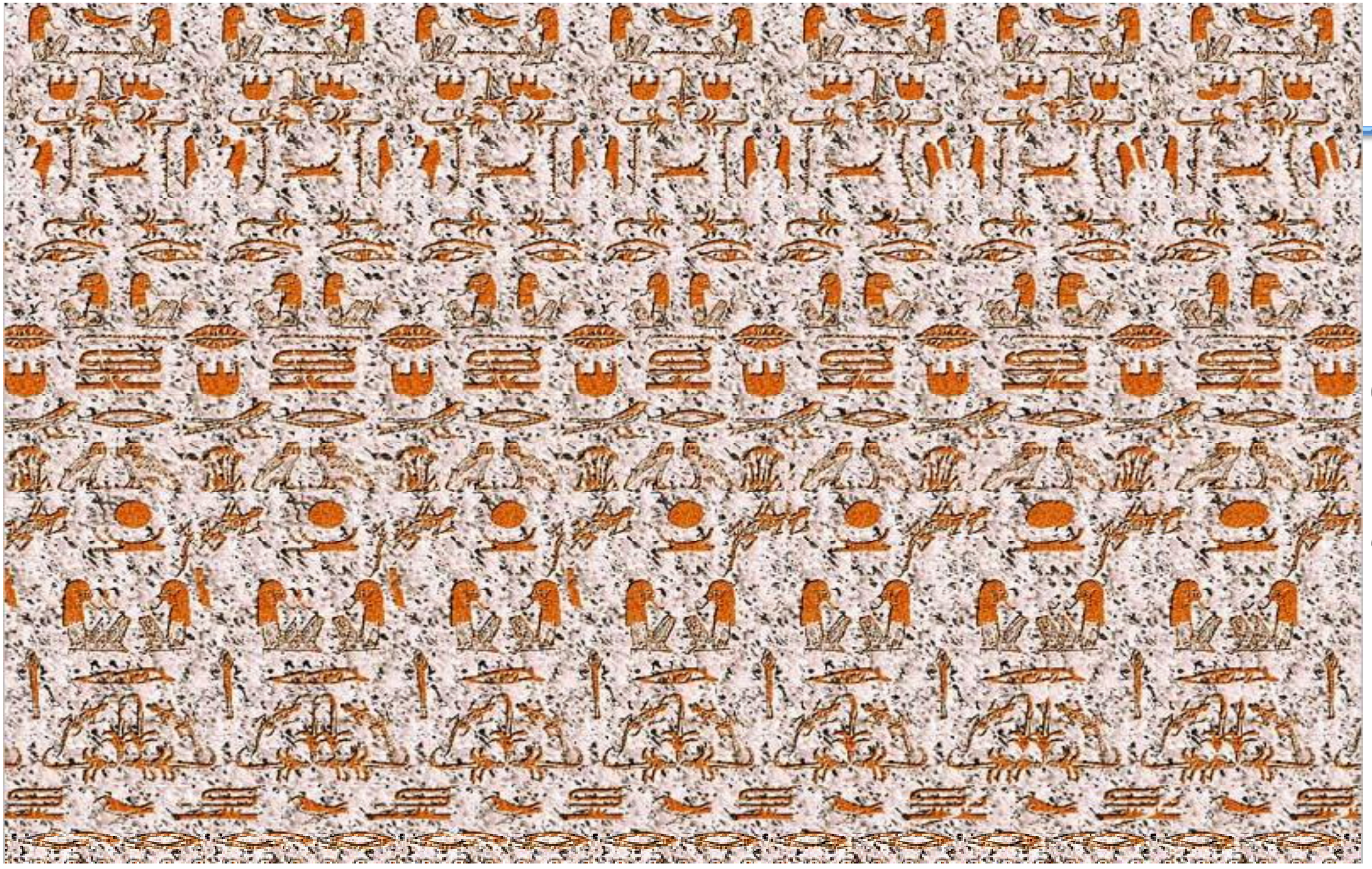


Why do we have two eyes?

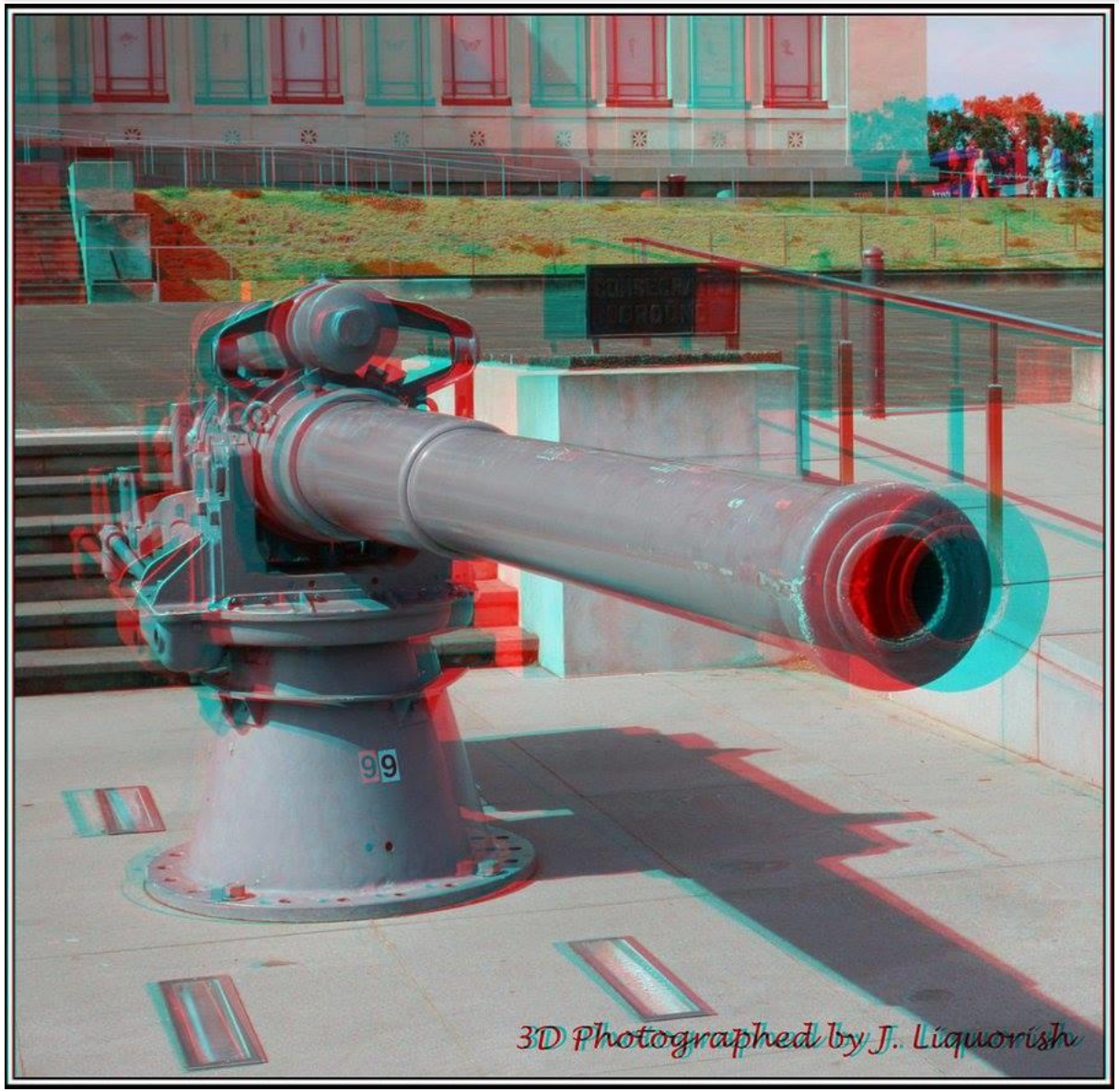
Stereograms
Magic Eye



Stereograms
Magic Eye



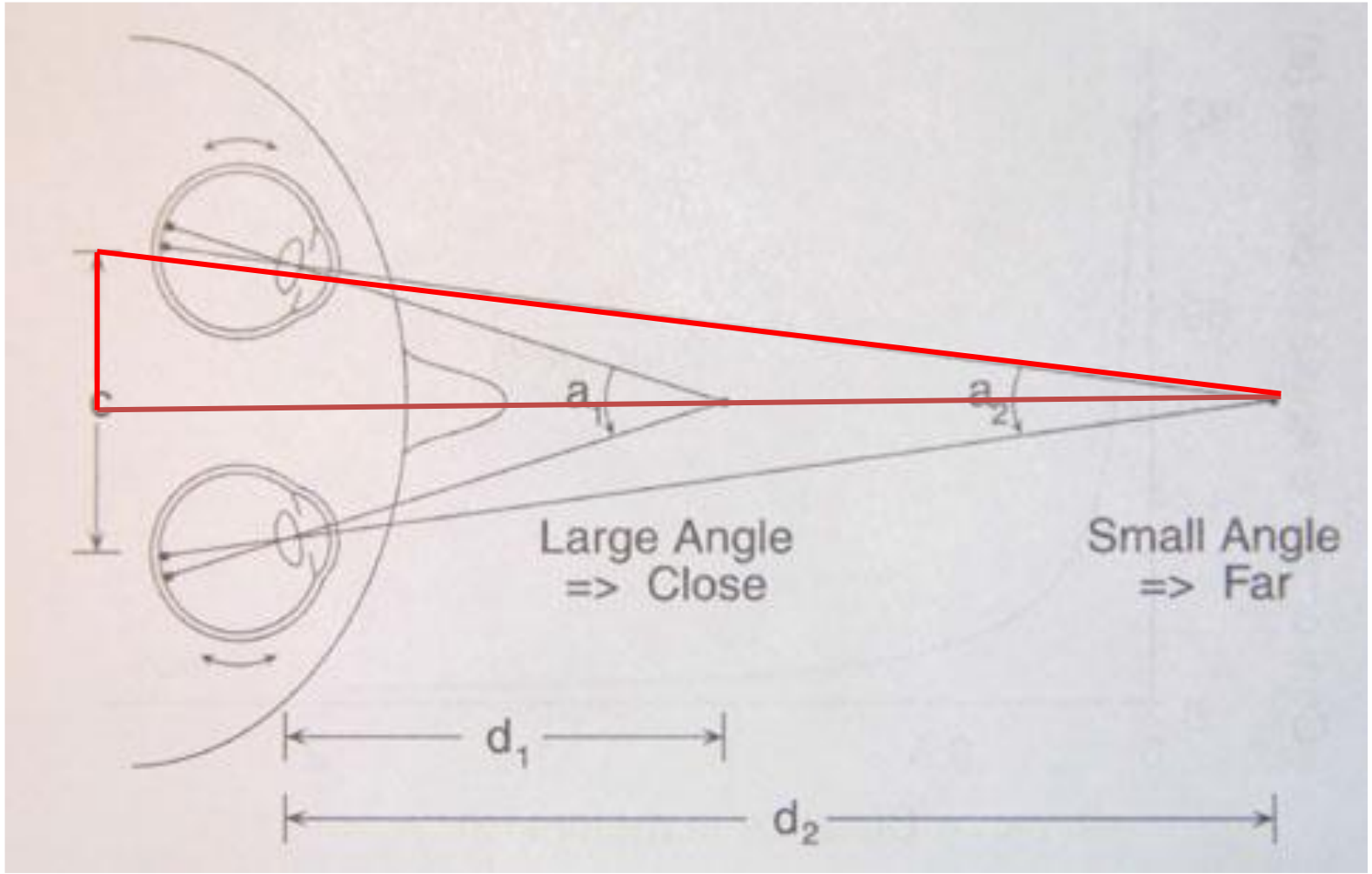
Stereograms Anaglyphs



Depth Estimation

From Convergence

$$d = \frac{c}{2tan(a/2)}$$



Human performance: up to 6-8 feet

Example: Depth Estimation From Stereo Video

Depth Estimation From Stereo Video

This example shows how to detect people in video taken with a calibrated stereo camera and determine their distances from the camera.

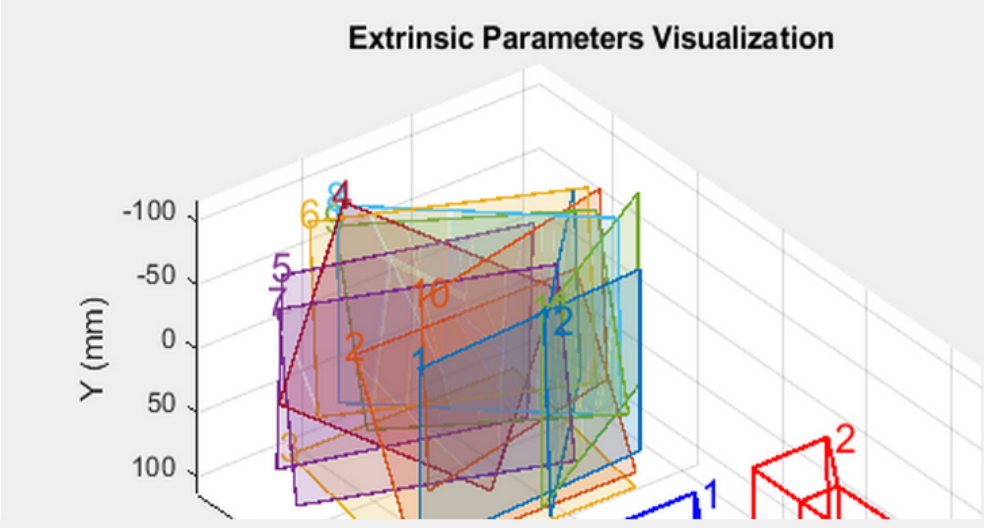
Open Live Script

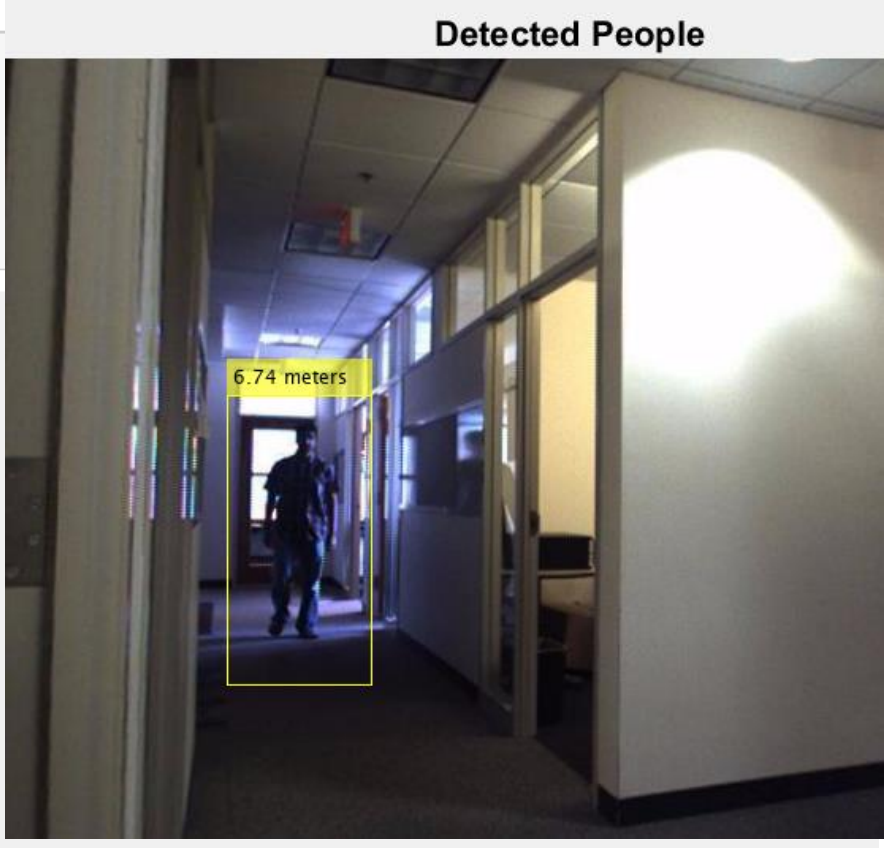
Load the Parameters of the Stereo Camera

Load the `stereoParameters` object, which is the result of calibrating the camera using either the `stereoCameraCalibrator` app or the `estimateCameraParameters` function.

```
% Load the stereoParameters object.
load('handshakeStereoParams.mat');

% Visualize camera extrinsics.
showExtrinsics(stereoParams);
```

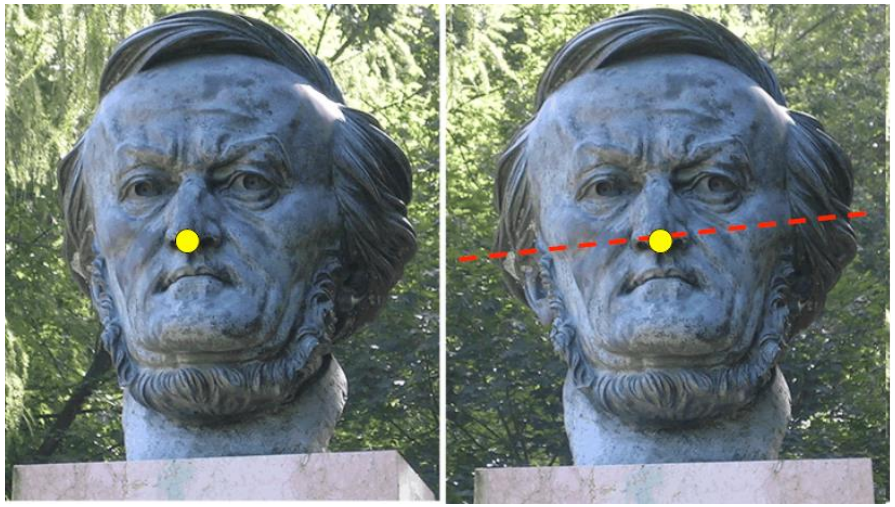




Epipolar Geometry

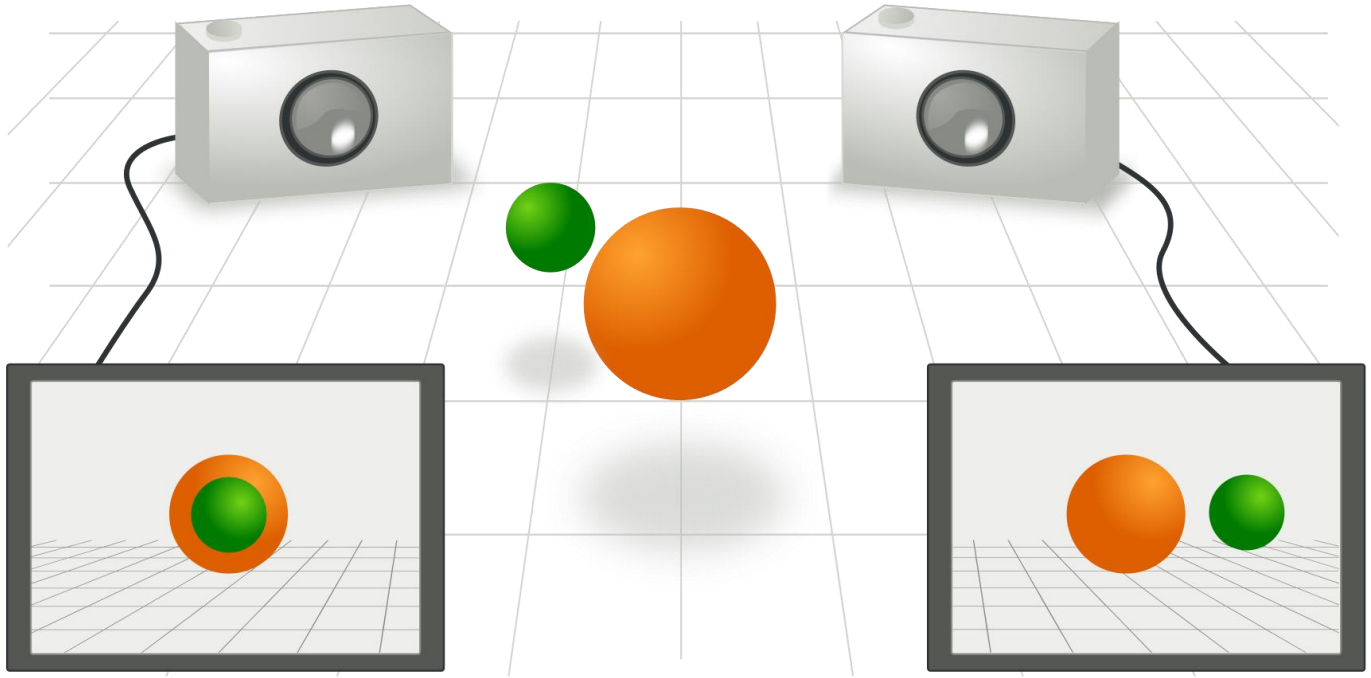
Epipolar geometry is the geometry of stereo vision. When two cameras view a 3D scene from two distinct positions, there are a number of geometric relations between the 3D points and their projections onto the 2D images that lead to constraints between the image points. These relations are derived based on the assumption that the cameras can be approximated by the pinhole camera model.

- Two views of the same object
- Suppose I know the camera positions and camera matrices
- Given a point on left image, how can I find the corresponding point on right image?



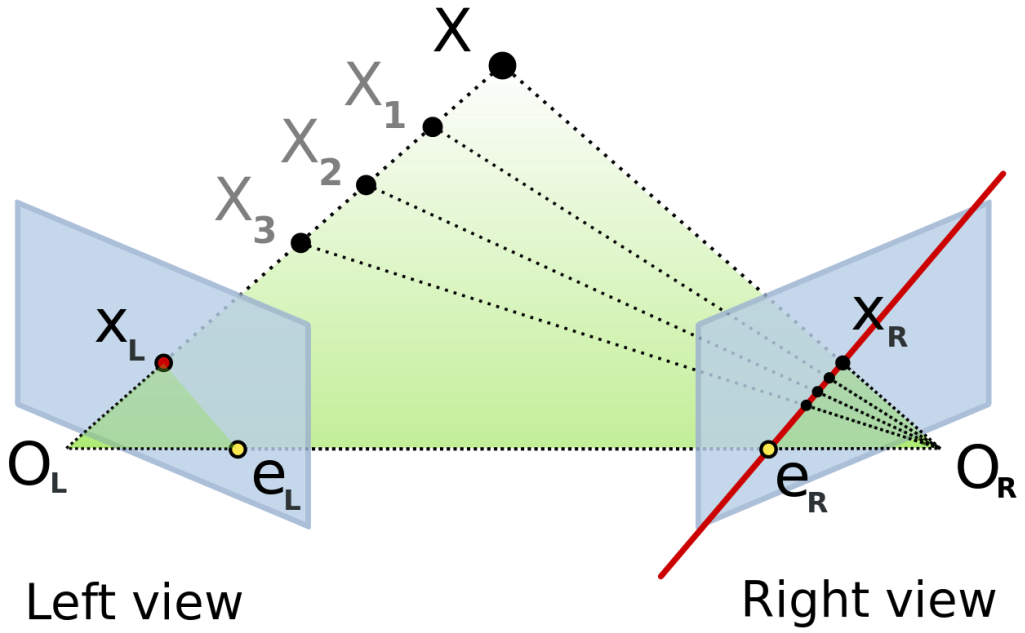
Epipolar Geometry

- Two cameras take a picture of the same scene from different points of view.
- The epipolar geometry then describes the relation between the two resulting views.



Epipolar Geometry

The figure depicts two pinhole cameras looking at point **X**. In real cameras, the image plane is actually behind the focal center, and produces an image that is symmetric about the focal center of the lens. Here, however, the problem is simplified by placing a *virtual image plane* in front of the focal center i.e. optical center of each camera lens to produce an image not transformed by the symmetry.



O_L and **O_R**: the centers of symmetry of the two cameras lenses.

X: the point of interest in both cameras.

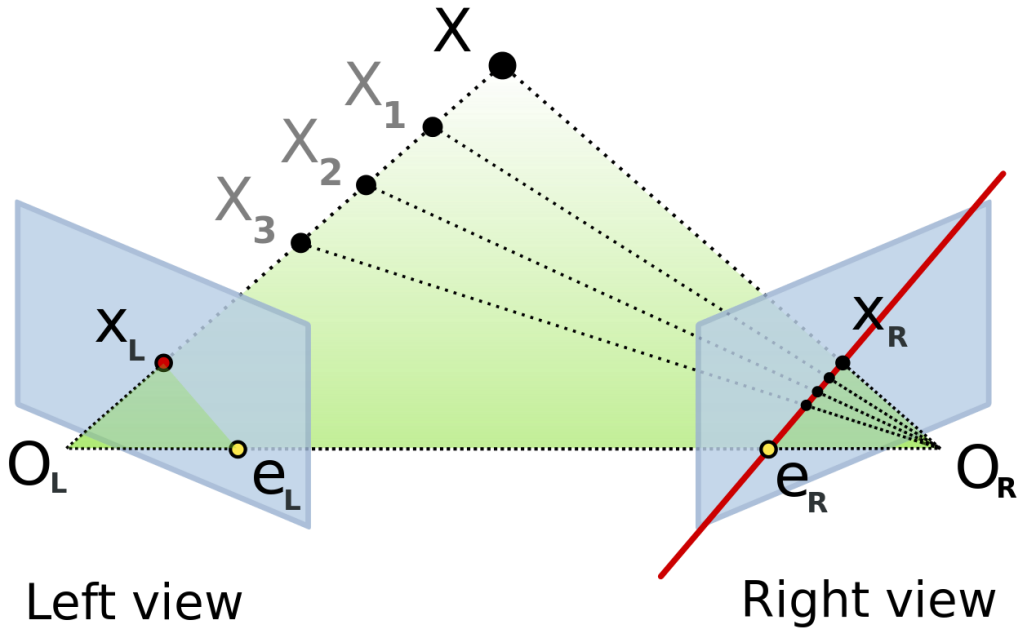
x_L and **x_R**: are the projections of point **X** onto the image planes.

Epipolar Geometry

Each camera captures a 2D image of the 3D world.

This conversion from 3D to 2D is referred to as a perspective projection and is described by the pinhole camera model.

It is common to model this projection operation by rays that emanate from the camera, passing through its focal center. Each emanating ray corresponds to a single point in the image.



$\mathbf{O}_L$ and $\mathbf{O}_R$: the centers of symmetry of the two cameras lenses.

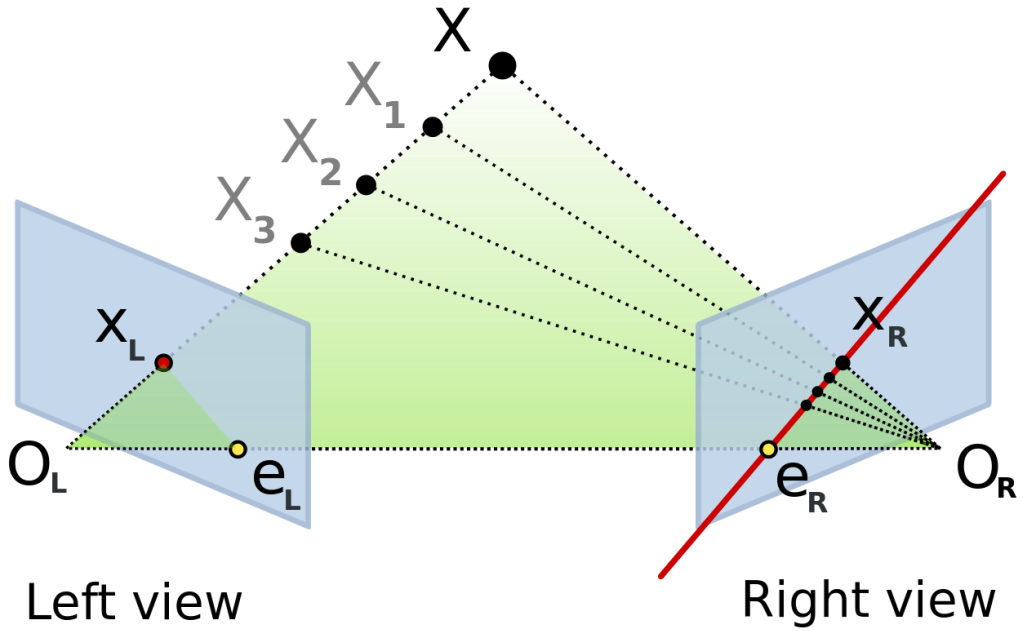
$\mathbf{X}$: the point of interest in both cameras.

$\mathbf{x}_L$ and $\mathbf{x}_R$: are the projections of point $\mathbf{X}$ onto the image planes.

Epipolar Geometry

Epipole or Epipolar Point

Since the optical centers of the cameras lenses are distinct, each center projects onto a distinct point into the other camera's image plane. These two image points, denoted by $\mathbf{e}_L$ and $\mathbf{e}_R$, are called *epipoles* or *epipolar points*. Both epipoles $\mathbf{e}_L$ and $\mathbf{e}_R$ in their respective image planes and both optical centers $\mathbf{O}_L$ and $\mathbf{O}_R$ lie on a single 3D line.



$\mathbf{O}_L$ and $\mathbf{O}_R$: the centers of symmetry of the two cameras lenses.

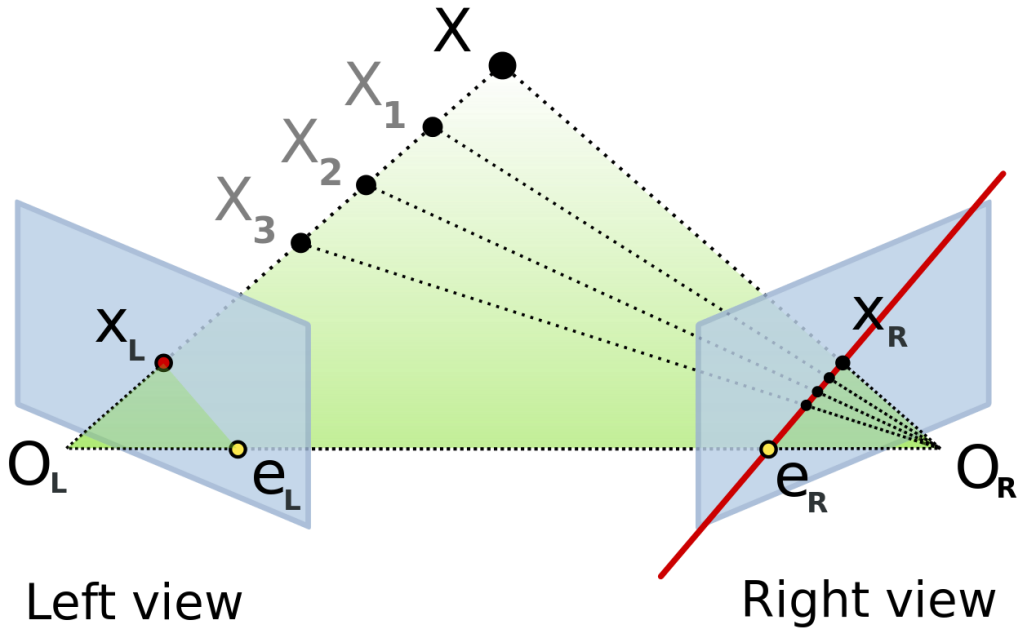
X: the point of interest in both cameras.

$\mathbf{x}_L$ and $\mathbf{x}_R$: are the projections of point **X** onto the image planes.

Epipolar Geometry

Epipolar Line

The line $\mathbf{O}_L$–$\mathbf{X}$ is seen by the left camera as a point because it is directly in line with that camera's lens optical center. However, the right camera sees this line as a line in its image plane. That line ($\mathbf{e}_R$–$\mathbf{x}_R$) in the right camera is called an *epipolar line*. Symmetrically, the line $\mathbf{O}_R$–$\mathbf{X}$ seen by the right camera as a point is seen as epipolar line $\mathbf{e}_L$–$\mathbf{x}_L$ by the left camera.



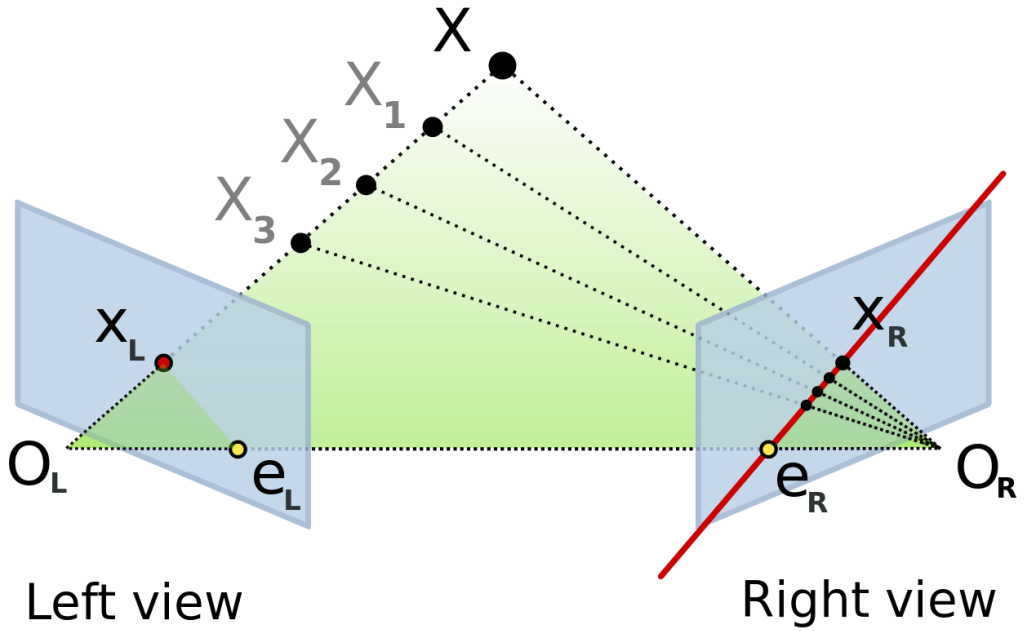
$\mathbf{O}_L$ and $\mathbf{O}_R$: the centers of symmetry of the two cameras lenses.

$\mathbf{X}$: the point of interest in both cameras.

$\mathbf{x}_L$ and $\mathbf{x}_R$: are the projections of point $\mathbf{X}$ onto the image planes.

Epipolar Geometry

An epipolar line is a function of the position of point **X** in the 3D space, i.e. as **X** varies, a set of epipolar lines is generated in both images. Since the 3D line $\mathbf{O}_L$–**X** passes through the optical center of the lens $\mathbf{O}_L$, the corresponding epipolar line in the right image must pass through the epipole $\mathbf{e}_R$ (and correspondingly for epipolar lines in the left image). All epipolar lines in one image contain the epipolar point of that image. In fact, any line which contains the epipolar point is an epipolar line since it can be derived from some 3D point **X**.



$\mathbf{O}_L$ and $\mathbf{O}_R$: the centers of symmetry of the two cameras lenses.

X: the point of interest in both cameras.

$\mathbf{x}_L$ and $\mathbf{x}_R$: are the projections of point **X** onto the image planes.

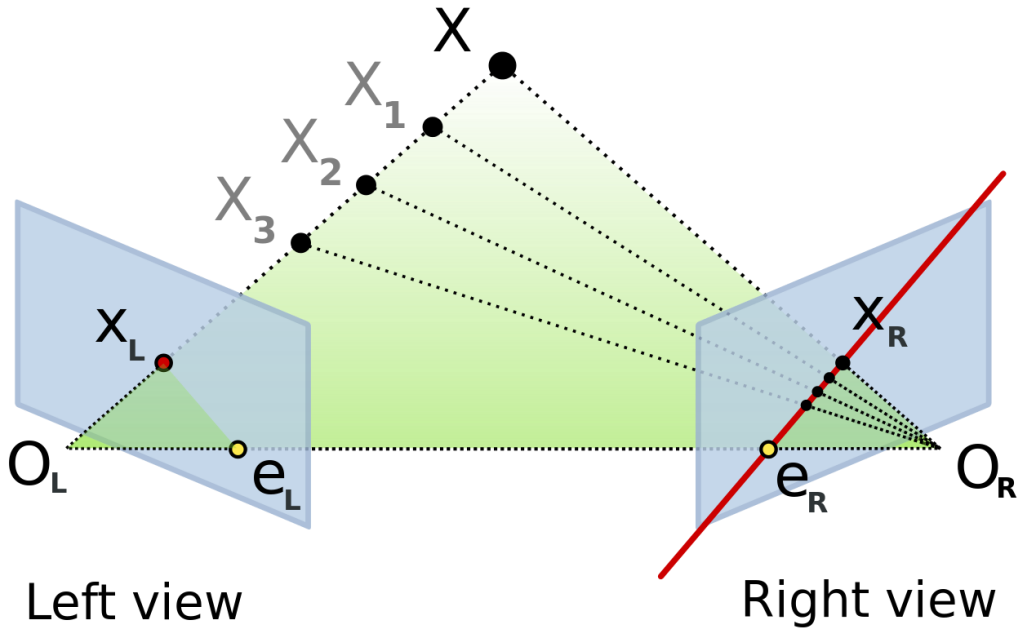
Epipolar Geometry

Epipolar Plane

As an alternative visualization, consider the points **X**, $\mathbf{O}_L$ & $\mathbf{O}_R$ that form a plane called the *epipolar plane*.

The epipolar plane intersects each camera's image plane where it forms lines—the epipolar lines.

All epipolar planes and epipolar lines intersect the epipole regardless of where **X** is located.



$\mathbf{O}_L$ and $\mathbf{O}_R$: the centers of symmetry of the two cameras lenses.

X: the point of interest in both cameras.

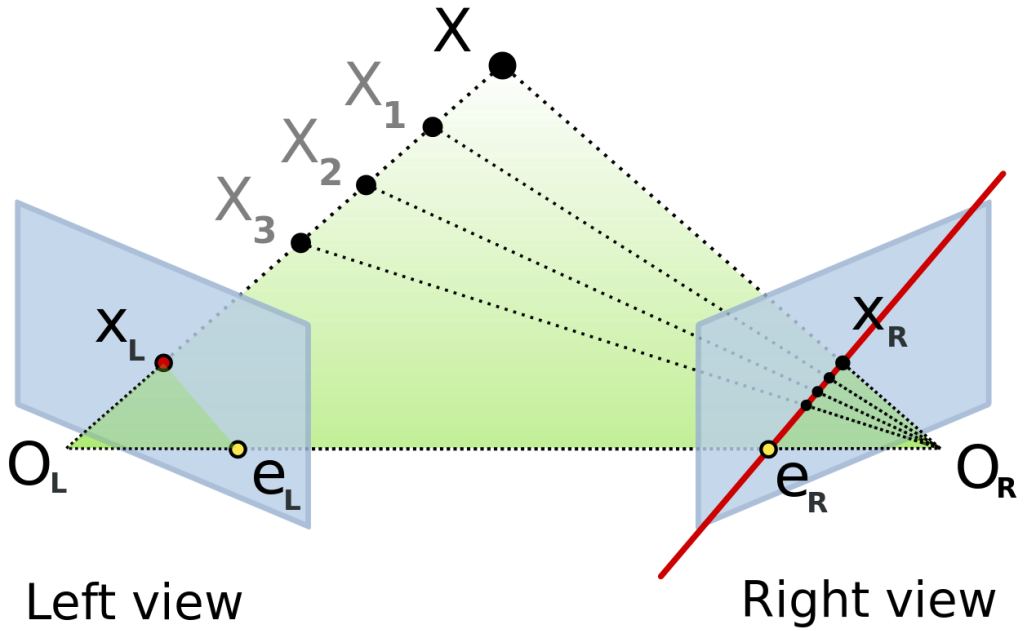
$\mathbf{x}_L$ and $\mathbf{x}_R$: are the projections of point **X** onto the image planes.

Epipolar Geometry

Epipolar Constraint & Triangulation

If the relative position of the two cameras is known, this leads to two important observations:

1. Assume the projection point $\mathbf{x}_L$ is known, and the epipolar line $\mathbf{e}_R$–$\mathbf{x}_R$ is known and the point **X** projects into the right image, on a point $\mathbf{x}_R$ which must lie on this particular epipolar line. This means that for each point observed in one image the same point must be observed in the other image on a known epipolar line.



$\mathbf{O}_L$ and $\mathbf{O}_R$: the centers of symmetry of the two cameras lenses.

X: the point of interest in both cameras.

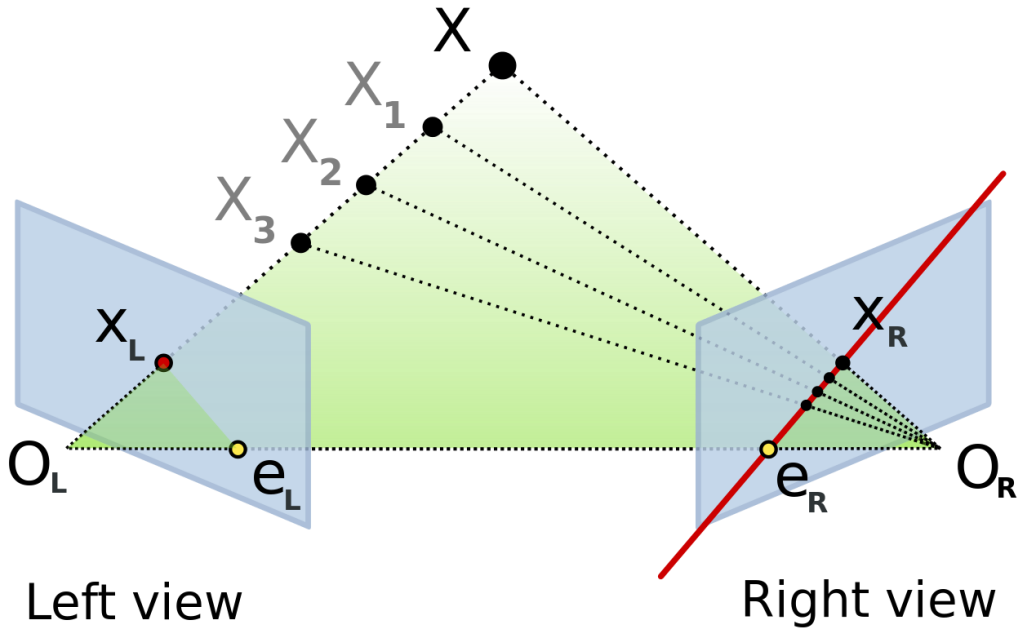
$\mathbf{x}_L$ and $\mathbf{x}_R$: are the projections of point **X** onto the image planes.

Epipolar Geometry

This provides an *epipolar constraint*:

The projection of X on the right camera plane $\mathbf{x}_R$ must be contained in the $\mathbf{e}_R$–$\mathbf{x}_R$ epipolar line. All points X e.g. $\mathbf{X}_1$, $\mathbf{X}_2$, $\mathbf{X}_3$ on the $\mathbf{O}_L$–$\mathbf{X}_L$ line will verify that constraint.

It means that it is possible to test if two points correspond to the same 3D point. Epipolar constraints can also be described by the essential matrix or the fundamental matrix between the two cameras.



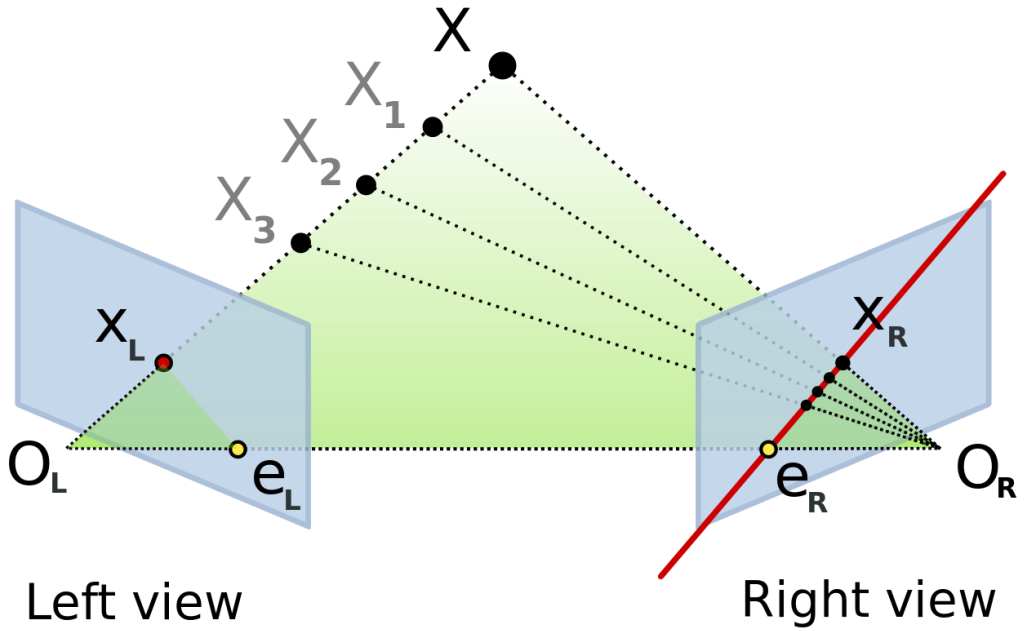
$\mathbf{O}_L$ and $\mathbf{O}_R$: the centers of symmetry of the two cameras lenses.

X: the point of interest in both cameras.

$\mathbf{x}_L$ and $\mathbf{x}_R$: are the projections of point **X** onto the image planes.

Epipolar Geometry

2. If the points $\mathbf{x}_L$ and $\mathbf{x}_R$ are known, their projection lines are also known. If the two image points correspond to the same 3D point **X** the projection lines must intersect precisely at **X**. This means that **X** can be calculated from the coordinates of the two image points, a process called *triangulation*.



$\mathbf{O}_L$ and $\mathbf{O}_R$: the centers of symmetry of the two cameras lenses.

X: the point of interest in both cameras.

$\mathbf{x}_L$ and $\mathbf{x}_R$: are the projections of point **X** onto the image planes.

Example: Epipolar Lines

« Documentation Home

« Computer Vision Toolbox

« Camera Calibration and 3-D Vision

« Structure From Motion

epipolarLine

ON THIS PAGE

Syntax

Description

Examples

Input Arguments

Output Arguments

Extended Capabilities

See Also

epipolarLine

Compute epipolar lines for stereo images

collapse all in page

Syntax

```
lines = epipolarLine(F,points)
lines = epipolarLine(F',points)
```

Description

example

`lines = epipolarLine(F,points)` returns an *M*-by-3 matrix, `lines`. The matrix represents the computed epipolar lines in image `I2` corresponding to the `points` in image `I1`. The input `F` represents the fundamental matrix that maps points in `I1` to epipolar lines in image `I2`.

`lines = epipolarLine(F',points)` The matrix represents the computed epipolar lines in image `I1` corresponding to the `points` in image `I2`.

Examples

collapse all

Compute Fundamental Matrix

This example shows you how to compute the fundamental matrix. It uses the least median of squares method to find the inliers.

Open Live Script

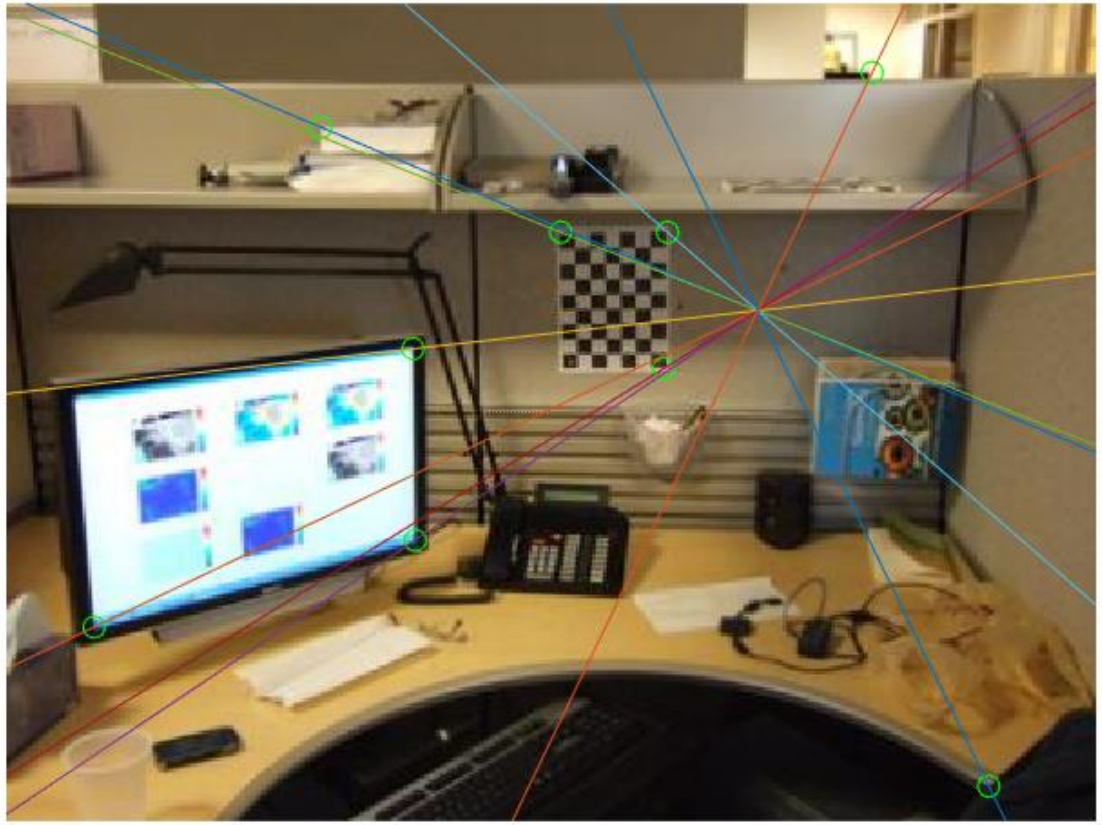
The points, matched_points1 and matched_points2, have been putatively matched.

```
load stereoPointPairs
[fLMedS,inliers] = estimateFundamentalMatrix(matchedPoints1,...
    matchedPoints2,'NumTrials',4000);
```

Examples: Epipolar Lines

Inliers and Epipolar Lines in First Image

Inliers and Epipolar Lines in Second Image



Structure from Motion

Structure from motion (SfM) is the process of estimating the 3D structure of a scene from a set of 2D images.

Applications:

- 3D scanning
- 3D printing
- Geosciences
- Cultural heritage
- Biological vision
- Augmented reality, etc.

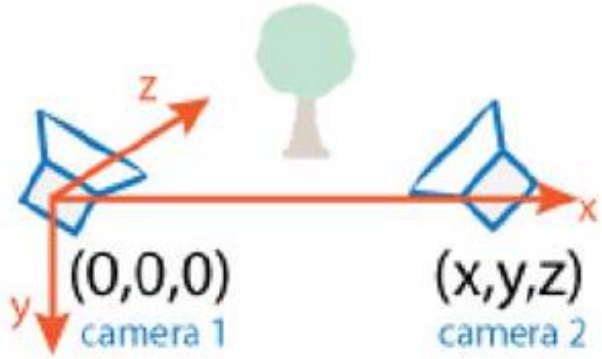
Structure from Motion

If the images are taken with a single calibrated camera, then the 3-D structure and camera motion can only be recovered *up to scale*. *Up to scale* means that you can rescale the structure and the magnitude of the camera motion and still maintain observations. For example, if you put a camera close to an object, you can see the same image as when you enlarge the object and move the camera far away. If you want to compute the actual scale of the structure and motion in world units, you need additional information, such as:

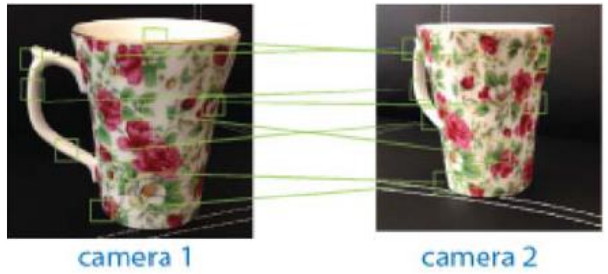
- The size of an object in the scene
- Information from another sensor, for example, an odometer.

Structure from Motion from Two Views

For the simple case of structure from two stationary cameras or one moving camera, one view must be considered camera 1 and the other one camera 2.



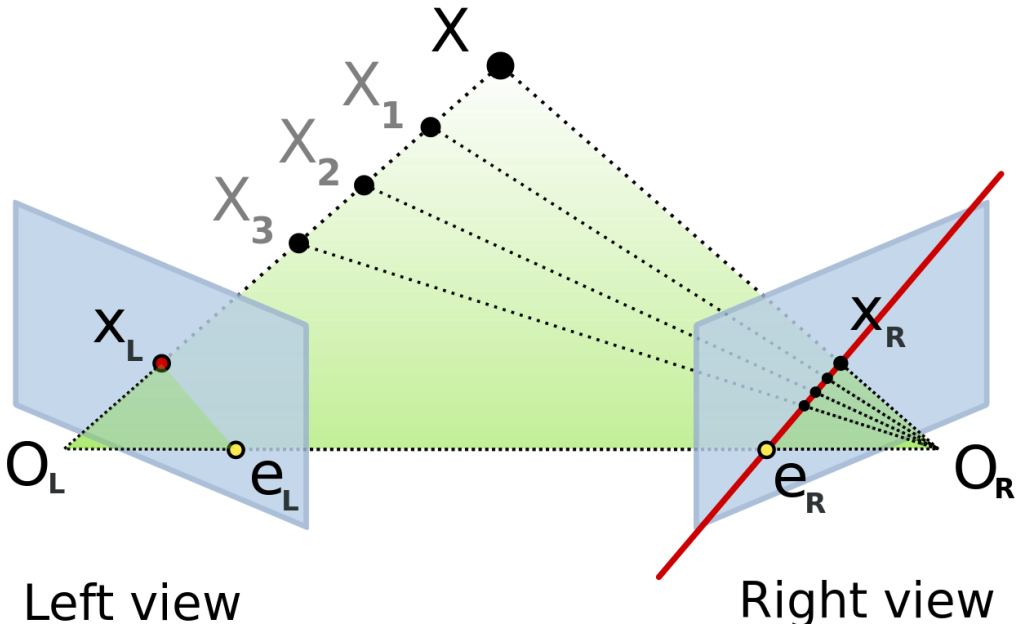
In this scenario, the algorithm assumes that camera 1 is at the origin and its optical axis lies along the *z*-axis.



1. SfM requires point correspondences between images. Find corresponding points either by matching features or tracking points from image 1 to image 2.

Structure from Motion from Two Views

2. To find the pose of the second camera relative to the first camera, you must compute the fundamental matrix. Use the corresponding points found in the previous step for the computation. The fundamental matrix describes the epipolar geometry of the two cameras. It relates a point in one camera to an epipolar line in the other camera.



Structure from Motion from Two Views

3. Find the orientation and the location of the second camera in the coordinate system of the first camera. The location can only be computed up to scale, so the distance between two cameras is normalized to 1 unit.
4. Determine the 3-D locations of the matched points. Because the pose is up to scale, when you compute the structure, it has the right shape but not the actual size.
5. Finally display the reconstruction, and visualize the camera poses

Example: 3D Reconstruction

Structure From Motion From Two Views

Structure from motion (SfM) is the process of estimating the 3-D structure of a scene from a set of 2-D images. This example shows you how to estimate the poses of a calibrated camera from two images, reconstruct the 3-D structure of the scene up to an unknown scale factor, and then recover the actual scale factor by detecting an object of a known size.

Open Script

Overview

This example shows how to reconstruct a 3-D scene from a pair of 2-D images taken with a camera calibrated using the Camera Calibrator app. The algorithm consists of the following steps:

1. Match a sparse set of points between the two images. There are multiple ways of finding point correspondences between two images. This example detects corners in the first image using the `detectMinEigenFeatures` function, and tracks them into the second image using `vision.PointTracker`. Alternatively you can use `extractFeatures` followed by `matchFeatures`.
2. Estimate the fundamental matrix using `estimateFundamentalMatrix`.
3. Compute the motion of the camera using the `relativeCameraPose` function.
4. Match a dense set of points between the two images. Re-detect the point using `detectMinEigenFeatures` with a reduced `'MinQuality'` to get more points. Then track the dense points into the second image using `vision.PointTracker`.
5. Determine the 3-D locations of the matched points using `triangulate`.
6. Detect an object of a known size. In this scene there is a globe, whose radius is known to be 10cm. Use `pcfitsphere` to find the globe in the point cloud.
7. Recover the actual scale, resulting in a metric reconstruction.

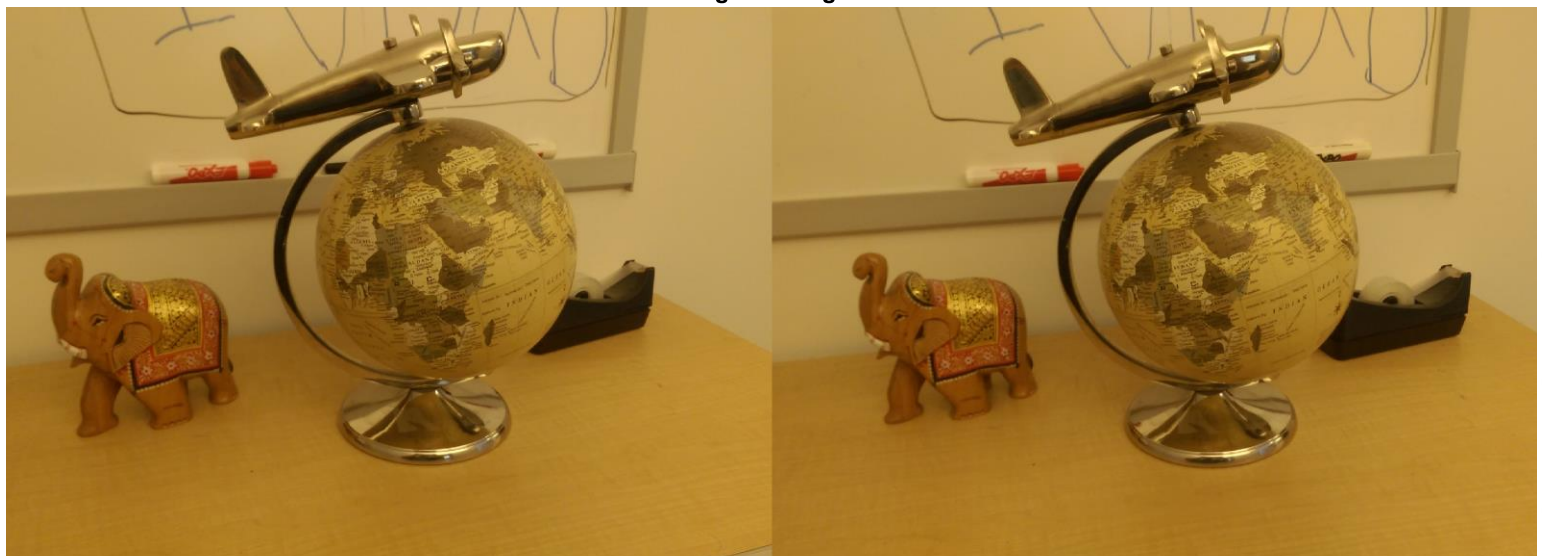
Read a Pair of Images

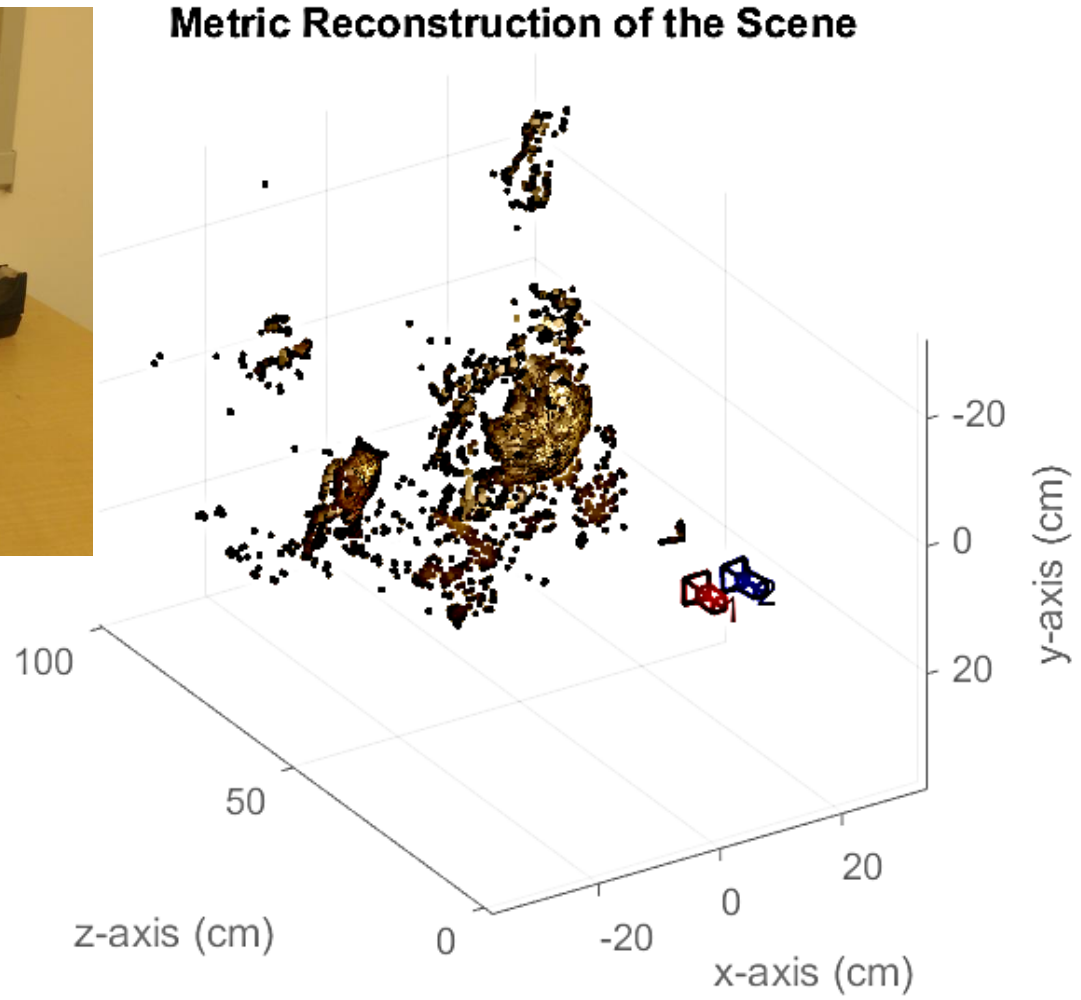
Load a pair of images into the workspace.

```
imageDir = fullfile(toolboxdir('vision'), 'visiondata','upToScaleReconstructionImages');
images = imageDatastore(imageDir);
I1 = readimage(images, 1);
```

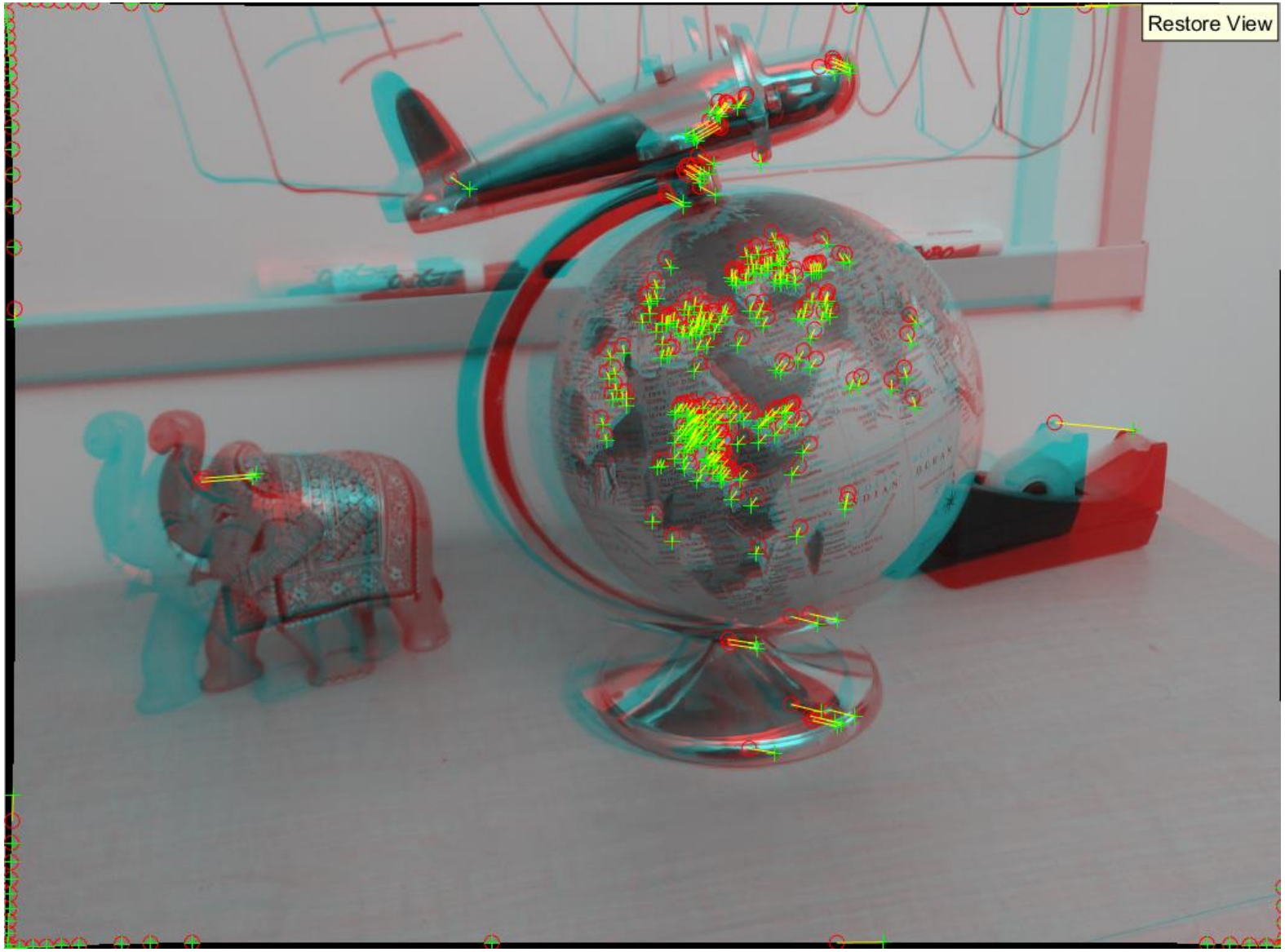
Example: 3D Reconstruction

Original Images





Example: 3D Reconstruction



3D Reconstruction from a Single Image

Online Demo

https://cvl-demos.cs.nott.ac.uk/vrn/