南京航空航天大學
NANJING UNIVERSITY OF AERONAUTICS AND ASTRONAUTICS

# 机器视觉测量与建模

## Machine vision based surveying and modelling


李明磊
南京航空航天大学 电子信息工程学院
E-mail: minglei_li@nuaa.edu.cn



## 6. 双目立体视觉

6.1 双目视觉系统介绍
6.2 密集匹配
6.3 结构光三维成像

## 6.1 双目视觉系统介绍

双目系统成本低、精度高
越来越普及

大疆 Mavic Pro 上搭载的 FlightAutonomy 2.0机身上集成了前、后、下三个方向的双目视觉传感器

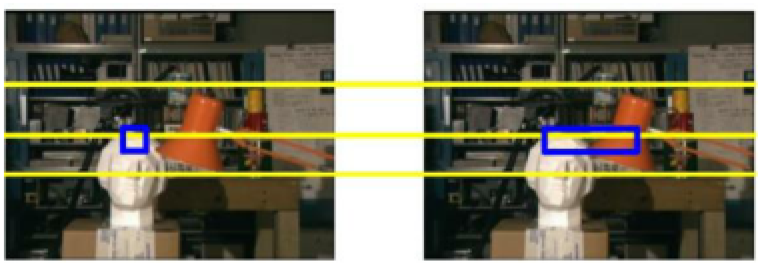
海康产品

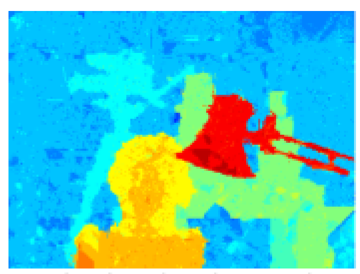
Commercial (binocular) stereo cameras


(Thanks to Slides from Levoy, Rusinkiewicz, Bouguet, Perona)

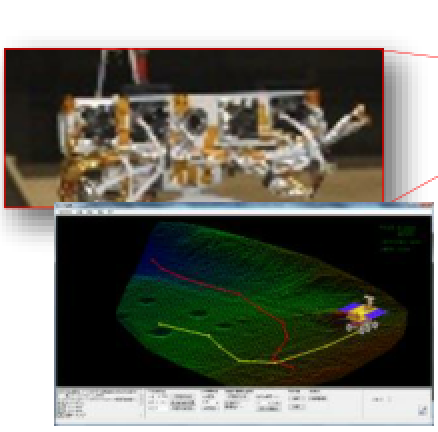



## 6.1 双目视觉系统介绍





disparity for depth sensing



中国嫦娥四号 月球车

2024年6月嫦娥6号采样返回支持

双目立体视觉技术，解决探测器的月面地形重建需求

## Real-time stereo sensing

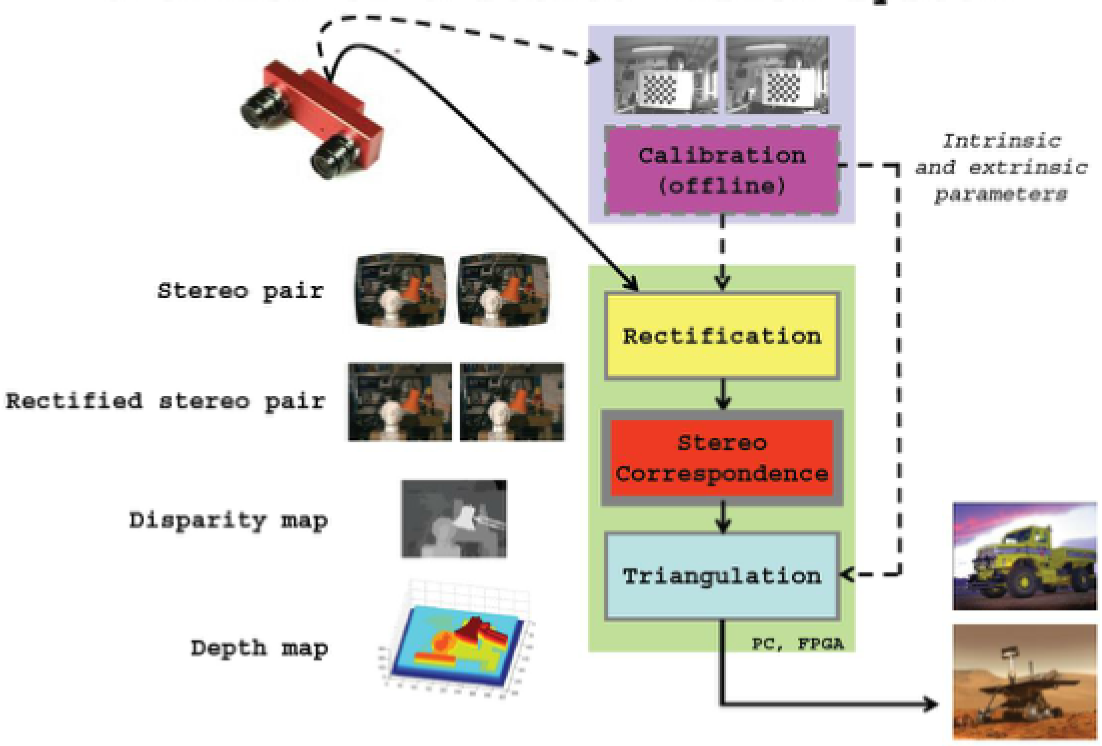
Nomad robot searches for meteorites in Antartica
http://www.frc.ri.cmu.edu/projects/meteorobot/index.html


Pre-collision braking

播放



## Overview of a stereo vision system



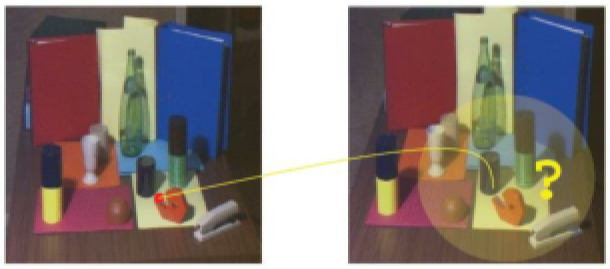
# 6.2 密集匹配

1. 双目校正 Stereo rectification
2. 立体匹配 Stereo matching
3. 三角测量 Triangulation



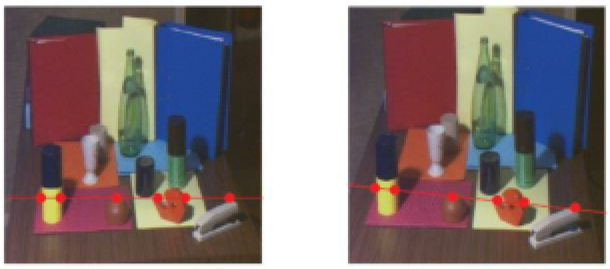
## 1、双目校正 Stereo rectification

**How to solve the correspondence problem ?**




Reference (R) Target (T)

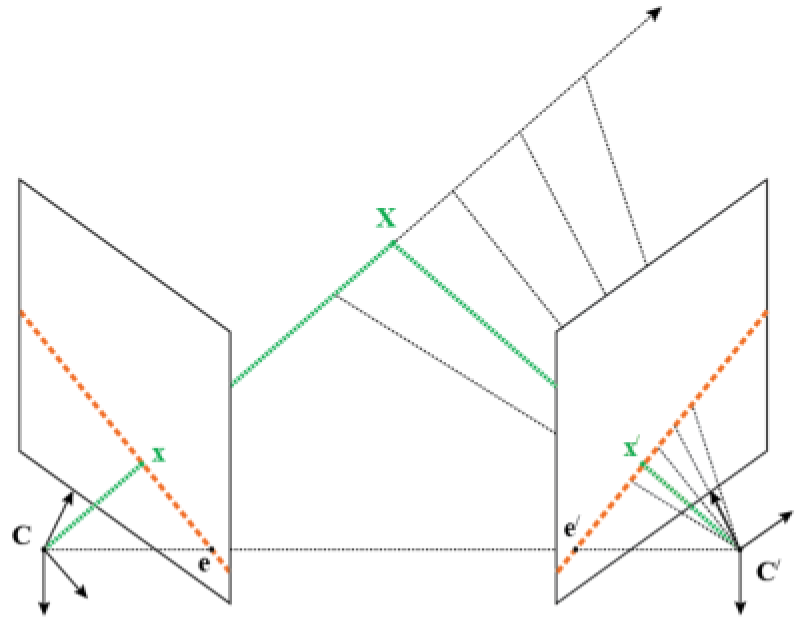
2D search domain ?



Reference (R) Target (T)

No!! Thanks to the epipolar constraint

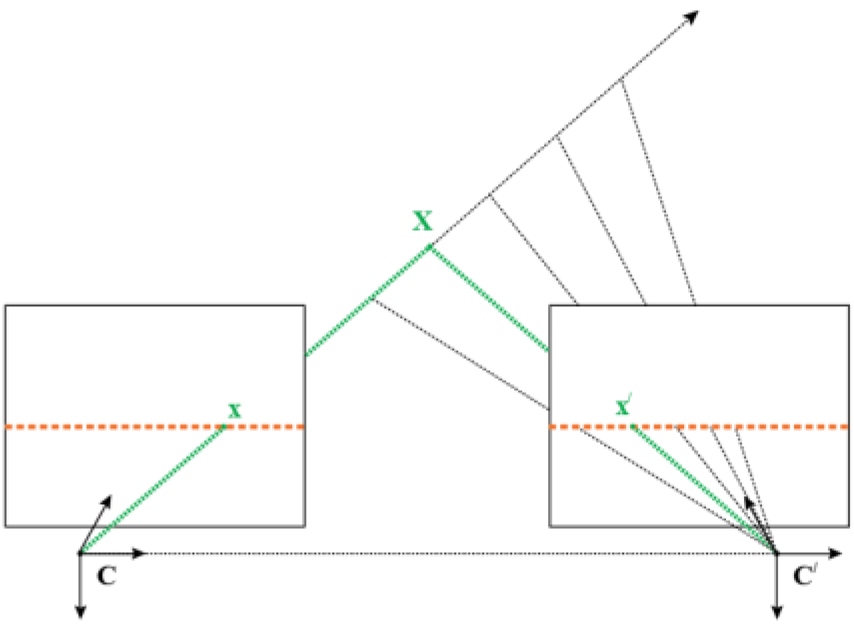
## 极线校正 Rectified Epipolar Geometry




Given x in the left image, reduces the search for x' to the epipolar line in the right image corresponding to x (1D search space)



## 极线校正 Rectified Epipolar Geometry




Speeds up and simplifies the search by warping the images such that correspondences lie on the same horizontal scan line

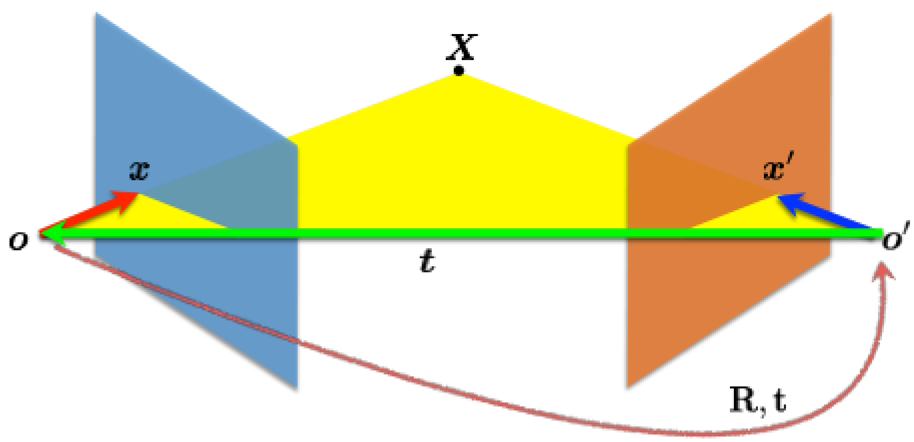
极线校正 Rectified Epipolar Geometry

极线校正
前后示例

*How can you make the epipolar lines horizontal?*

*Use stereo rectification*






$$x' = \mathbf{R}(x - t)$$

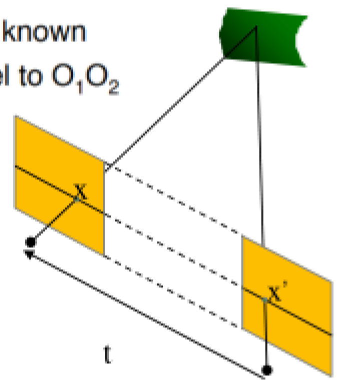
回顾

$$a \times b = [a]_\times b = \boxed{\begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix}} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

Skew symmetric
反对称矩阵

*When are epipolar lines horizontal?*

$K_1=K_2$ = known
x parallel to $O_1O_2$




The image of a 3D point will always be on the same horizontal line

$$\boxed{Tv = Tv'}$$

y coordinate is always the same!

**只有当右像相对于左像的转换满足:**

$$\boxed{R = I \qquad t = (T, 0, 0)}$$

Prove:

$$E = t \times R = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -T \\ 0 & T & 0 \end{bmatrix}$$

This always has to hold:

$$x^T E x' = 0$$

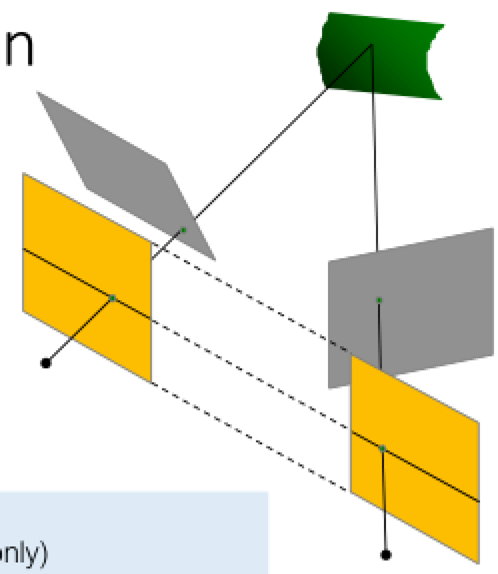
Write out the constraint

$$\begin{pmatrix} u & v & 1 \end{pmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -T \\ 0 & T & 0 \end{bmatrix} \begin{pmatrix} u' \\ v' \\ 1 \end{pmatrix} = 0 \qquad \begin{pmatrix} u & v & 1 \end{pmatrix} \begin{pmatrix} 0 \\ -T \\ Tv' \end{pmatrix} = 0$$



# Stereo Rectification

*What is stereo rectification?*



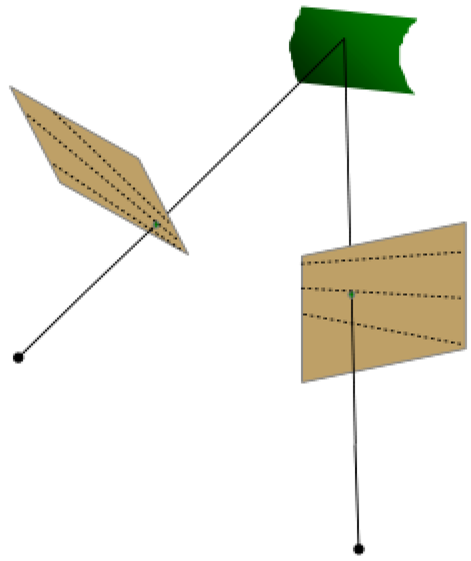
Reproject image planes onto a common plane parallel to the line between camera centers

Need 2 Homography Matrices $H_{3\times3}$ , one for each input image reprojection

1. **Rotate** the right camera by **R**
   (aligns camera coordinate system orientation only)
2. Rotate (**rectify**) the left camera so that the epipole is at infinity
3. Rotate (**rectify**) the right camera so that the epipole is at infinity
4. Adjust the **scale**


Ref. C. Loop and Z. Zhang. Computing Rectifying Homographies for Stereo Vision.Computer Vision and Pattern Recognition, 1999.

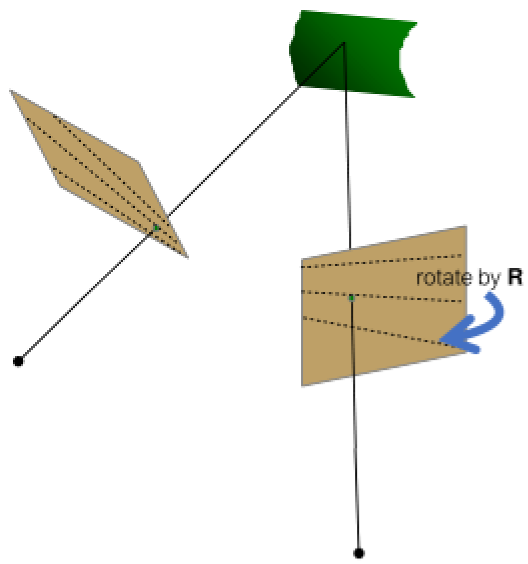
## Stereo Rectification:



1. Compute **E** to get **R**
2. Rotate right image by **R**
3. Rotate both images by $\mathbf{R}_{rect}$
4. Scale both images by **H**

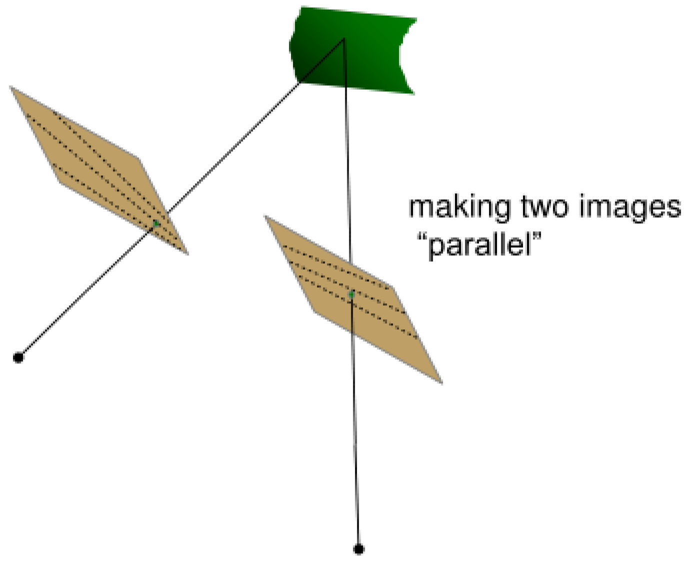


## Stereo Rectification:




1. Compute **E** to get **R**
2. Rotate right image by **R**
3. Rotate both images by $\mathbf{R}_{rect}$
4. Scale both images by **H**

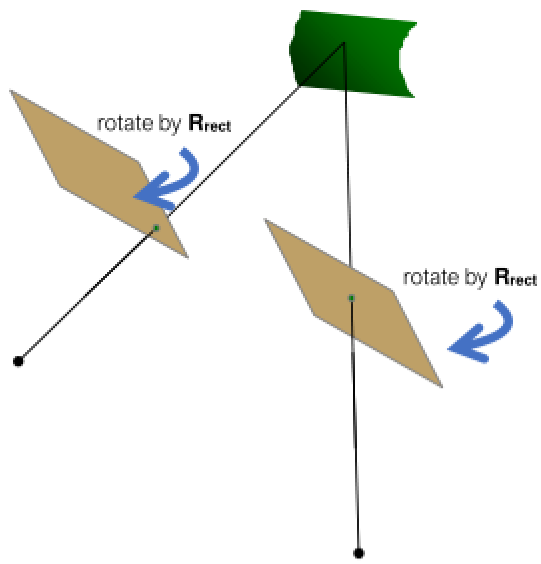
## Stereo Rectification:




1. Compute **E** to get **R**
2. Rotate right image by **R**
3. Rotate both images by $\mathbf{R}_{rect}$
4. Scale both images by **H**

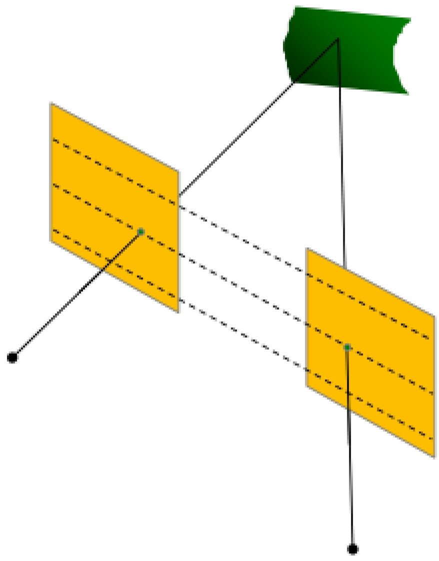


## Stereo Rectification:




1. Compute **E** to get **R**
2. Rotate right image by **R**
3. Rotate both images by $\mathbf{R}_{rect}$
4. Scale both images by **H**

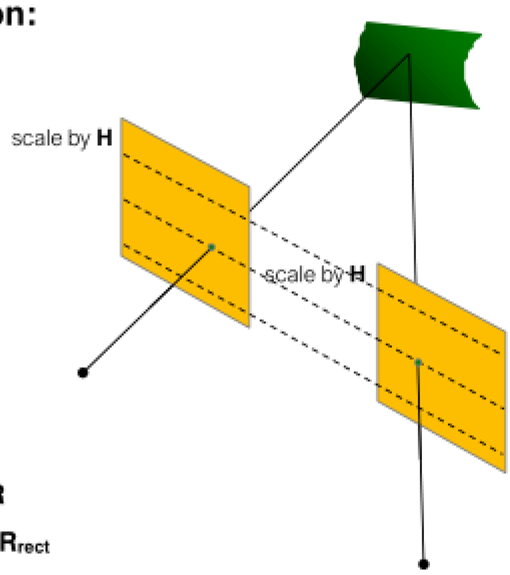
### Stereo Rectification:



1. Compute **E** to get **R**
2. Rotate right image by **R**
3. Rotate both images by $\mathbf{R}_{\text{rect}}$
4. Scale both images by **H**

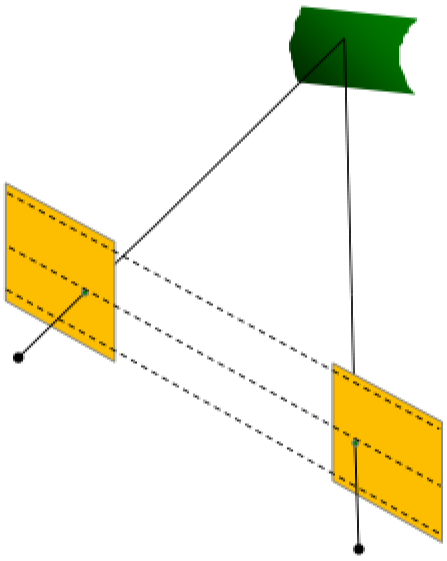


### Stereo Rectification:




1. Compute **E** to get **R**
2. Rotate right image by **R**
3. Rotate both images by $\mathbf{R}_{\text{rect}}$
4. Scale both images by **H**

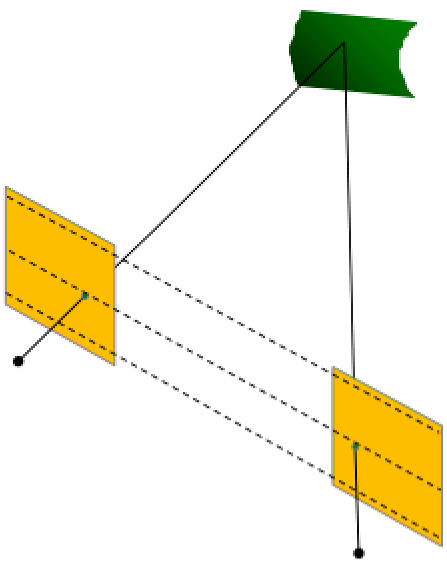
**Stereo Rectification:**



1. Compute **E** to get **R**
2. Rotate right image by **R**
3. Rotate both images by $\mathbf{R}_{rect}$
4. Scale both images by **H**



**Stereo Rectification:**




1. Compute **E** to get **R**
2. Rotate right image by **R**
3. Rotate both images by $\mathbf{R}_{rect}$
4. Scale both images by **H**

## **Step 1:** Compute **E** to get **R**

SVD: $\mathbf{E} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\top$ Let $\mathbf{W} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

2 possible rotations

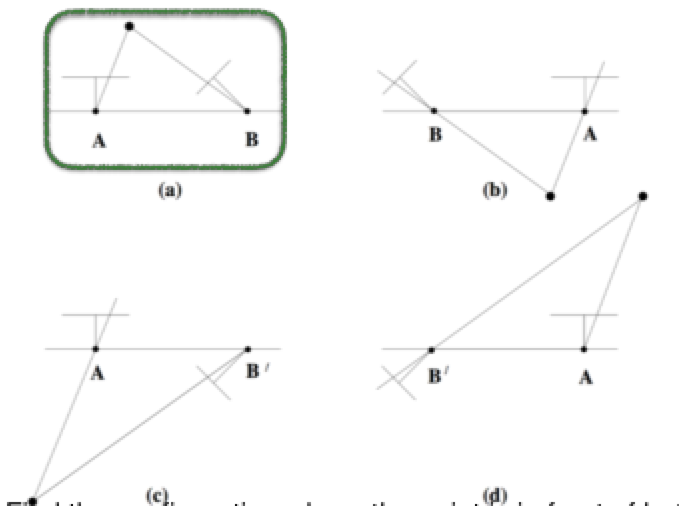
$\mathbf{R}_1 = \mathbf{U}\mathbf{W}\mathbf{V}^\top \quad \mathbf{R}_2 = \mathbf{U}\mathbf{W}^\top\mathbf{V}^\top$

2 possible translations

$\mathbf{T}_1 = U_3 \quad \mathbf{T}_2 = -U_3$

We get 4 solutions:

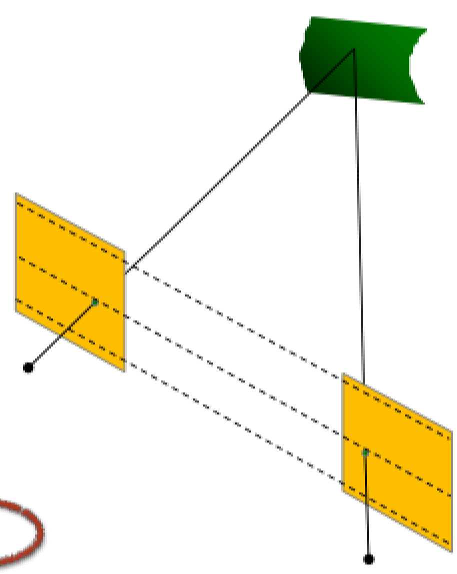
| | |
|---|---|
| $\mathbf{R}_1 = \mathbf{U}\mathbf{W}\mathbf{V}^\top$<br>$\mathbf{T}_1 = U_3$ | $\mathbf{R}_1 = \mathbf{U}\mathbf{W}\mathbf{V}^\top$<br>$\mathbf{T}_2 = -U_3$ |
| $\mathbf{R}_2 = \mathbf{U}\mathbf{W}^\top\mathbf{V}^\top$<br>$\mathbf{T}_2 = -U_3$ | $\mathbf{R}_2 = \mathbf{U}\mathbf{W}^\top\mathbf{V}^\top$<br>$\mathbf{T}_1 = U_3$ |




Find the configuration where the point is in front of both cameras

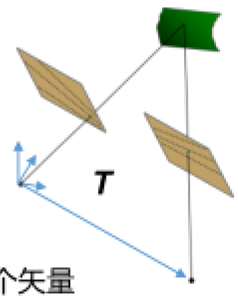


**Stereo Rectification:**



1. Compute **E** to get **R**
2. Rotate right image by **R**
3. Rotate both images by $\mathbf{R}_{\text{rect}}$
4. Scale both images by **H**

## 构造法




Building $\mathbf{R}_{\text{rect}}$　使满足，映射后极点**e**在无穷远处

$$\text{构造一个 } R_{\text{rect}} = \begin{bmatrix} \boldsymbol{r}_1^\top \\ \boldsymbol{r}_2^\top \\ \boldsymbol{r}_3^\top \end{bmatrix}$$

- 给定***E***，做SVD分解。
- 从本质矩阵***E***中可以算出平移向量***T***
- ***T***是从左图像的像空间坐标系发出的一个矢量

$$\boldsymbol{r}_1 = \frac{T}{||T||}$$

构造：让 $r1$ 和 translation vector ***T*** 射影空间中一致
（从左图像的光连接e1的矢量和***T*** 共线）

$$\boldsymbol{r}_2 = \frac{1}{\sqrt{T_x^2 + T_y^2}} \begin{bmatrix} -T_y & T_x & 0 \end{bmatrix}$$

矢量 [-Ty, Tx, 0] 与 e1 正交

$$\boldsymbol{r}_3 = \boldsymbol{r}_1 \times \boldsymbol{r}_2$$

cross product of e and the direction vector of the optical axis → orthogonal vector

$\boldsymbol{r}_1$　$\boldsymbol{r}_2$　$\boldsymbol{r}_3$ 是正交的

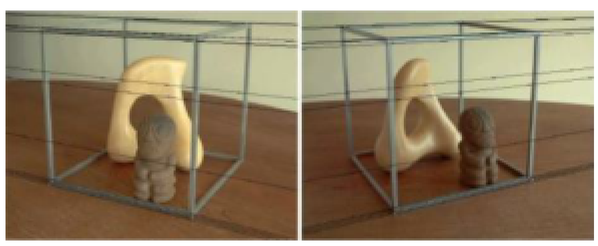
then
$$R_{\text{rect}}\boldsymbol{e}_1 = \begin{bmatrix} \boldsymbol{r}_1^\top \boldsymbol{e}_1 \\ \boldsymbol{r}_2^\top \boldsymbol{e}_1 \\ \boldsymbol{r}_3^\top \boldsymbol{e}_1 \end{bmatrix} = \begin{bmatrix} ? \\ ? \\ ? \end{bmatrix}$$

$$R_{\text{rect}}\boldsymbol{e}_1 = \begin{bmatrix} \boldsymbol{r}_1^\top \boldsymbol{e}_1 \\ \boldsymbol{r}_2^\top \boldsymbol{e}_1 \\ \boldsymbol{r}_3^\top \boldsymbol{e}_1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

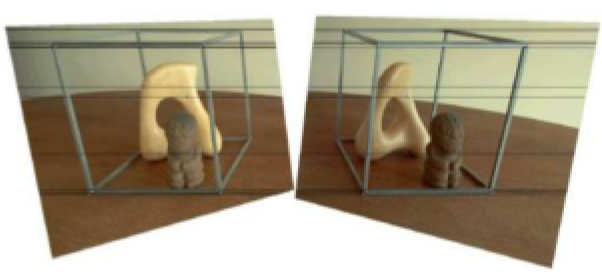
在 x方向上的 infinity

*Where is this point located on the image plane?*　满足：极点在无穷远处



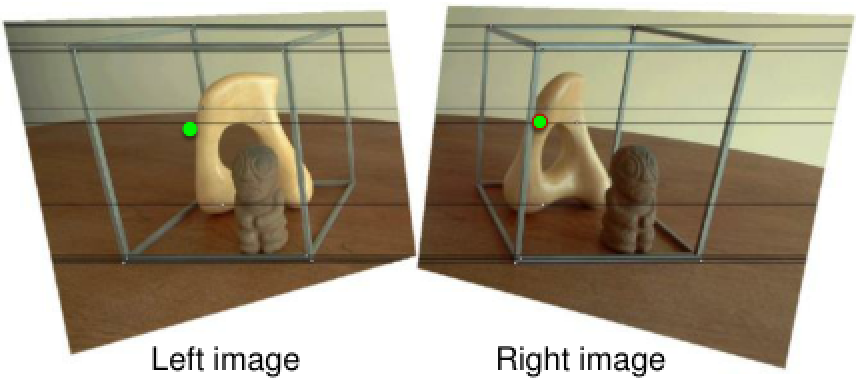


*What do we do after rectifying the two image planes?*



Stereo matching

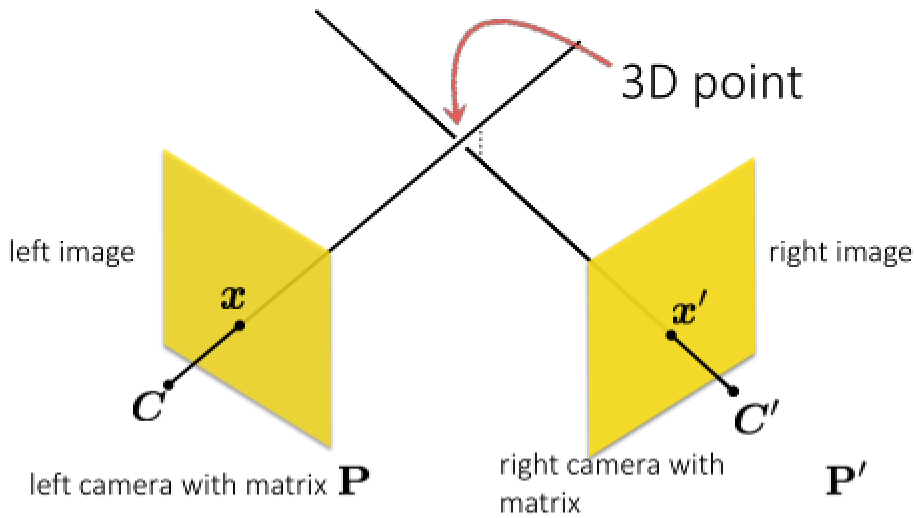
# How would you reconstruct 3D points?




1. 在一个图像中选择点
2. 在第二张图中形成该点的极线
3. 沿直线寻找匹配点
4. 进行三角测量

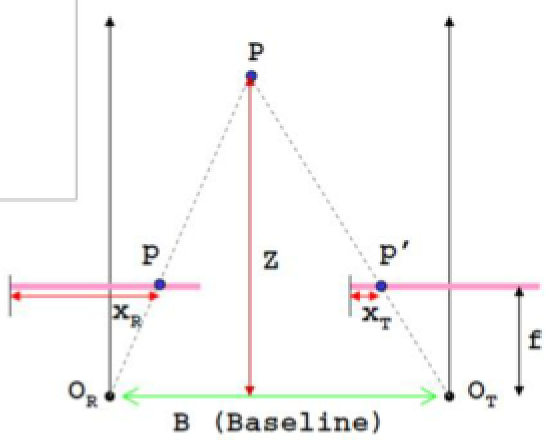


# 三角测量
Triangulation



## 三角测量
## Triangulation

### Disparity and depth




With the stereo rig in standard form and by considering similar triangles ($PO_RO_T$ and Ppp'):

与深度Z成反比

$$\frac{b}{Z}=\frac{(b+x_T)-x_R}{Z-f} \Rightarrow Z=\frac{b\cdot f}{x_R-x_T}=\frac{b\cdot f}{d}$$

视差
(Disparity)

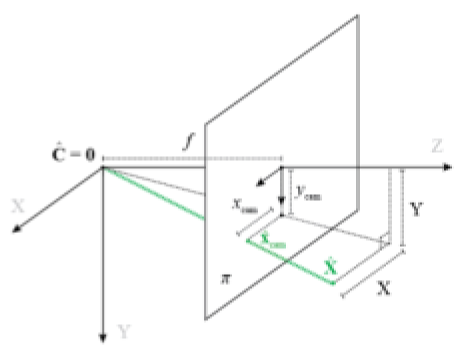
$x_R - x_T$ is the **disparity**

Stefano Mattoccia



## 三角测量
## Triangulation

Given the disparity map, the baseline and the Focal length (calibration): triangulation computes the position of the correspondence in the 3D space

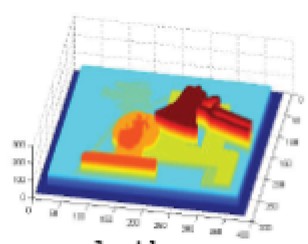




disparity map

$$Z=\frac{b\cdot f}{d}$$

$$X=Z\frac{x_R}{f}$$

$$Y=Z\frac{y_R}{f}$$

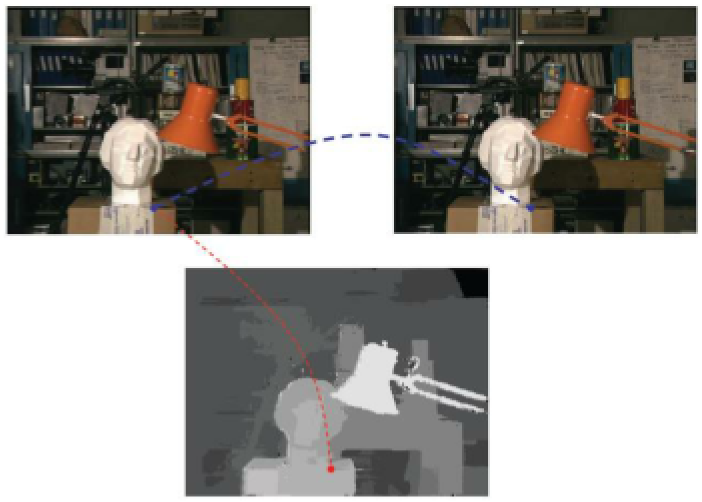


depth map

# Stereo matching

## Stereo correspondence

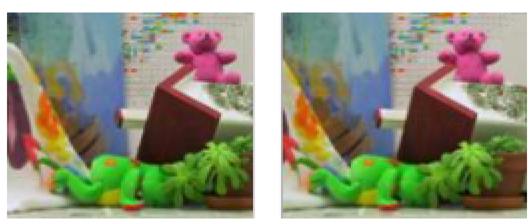
Aims at finding homologous points in the stereo pair.



disparity map

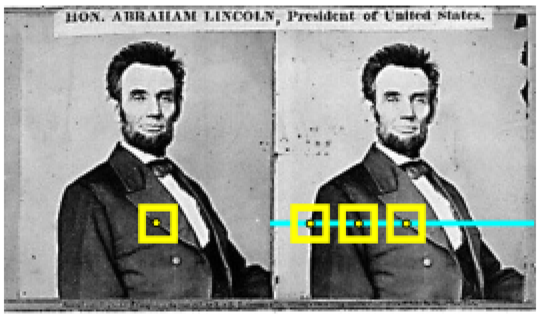


## Stereo matching



Depth Estimation via Stereo Matching

## Stereo matching



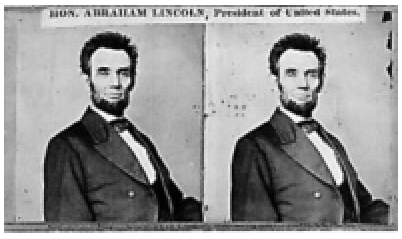

1. Rectify images
   (make epipolar lines horizontal)
2. For each $\boxed{\text{pixel}}$
   a. Find epipolar line
   b. Scan line for best match ← How would you do this?
   c. Compute depth from disparity

$$Z = \frac{bf}{d}$$



## Stereo matching

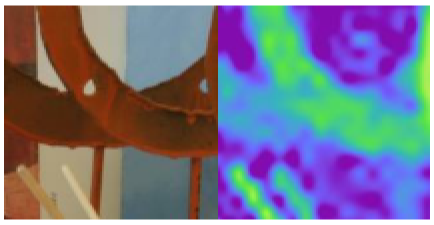
*When are correspondences difficult?*



**Textureless regions**

**Repeated patterns**

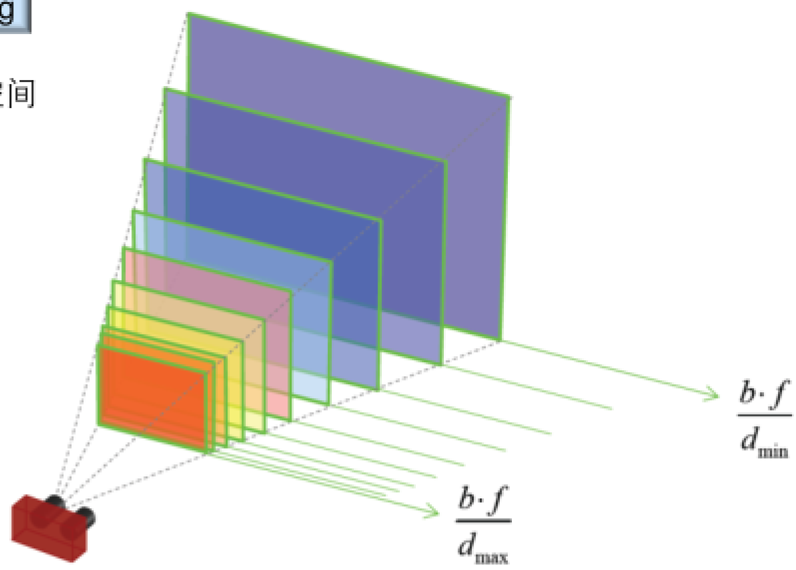
**Specularities**
**镜面反射**



**Depth discontinuities**

## Stereo matching

视差空间



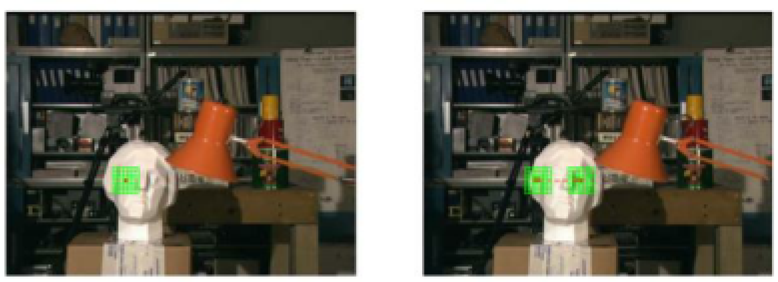

- Depth measured by a stereo vision system is discretized into parallel planes (one for each disparity value)
- A better (virtual) discretization can be achieved with subpixel techniques (see Disparity Refinements)

Stefano Mattoccia

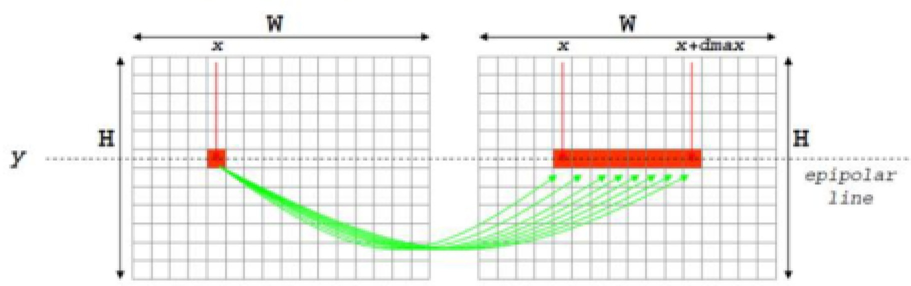


## Stereo matching

The simplest (naive and unused) local approach:



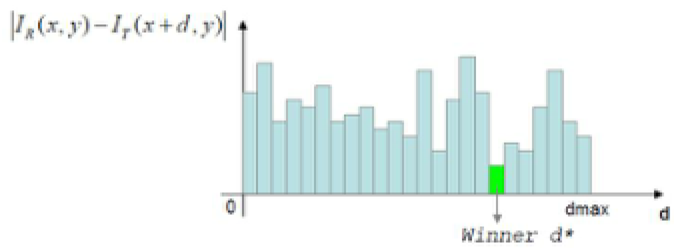
Reference (R) Target (T)




Reference (R) Target (T)

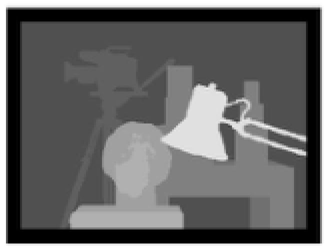
## Stereo matching

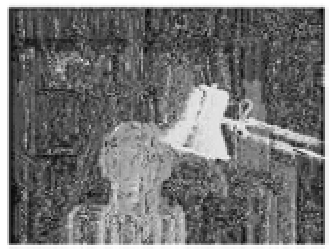
- matching cost (1): pixel-based absolute difference between pixel intensities
- disparity computation (3): Winner Takes All (WTA)





Reference


Groundtruth


Result
(disappointing)



## Stereo matching

- 由匹配算法计算的原始视差图包含异常值，需要检测和校正
- 由于视差映射通常在离散的像素级计算，因此需要更精确的视差分配

优化视差图的技术

a) **Graph Cuts**
b) **Belief Propagation**
c) **Cooperative optimization**
d) **Dynamic Programming (DP)**

视差寻优问题，经典的代价能量方程：

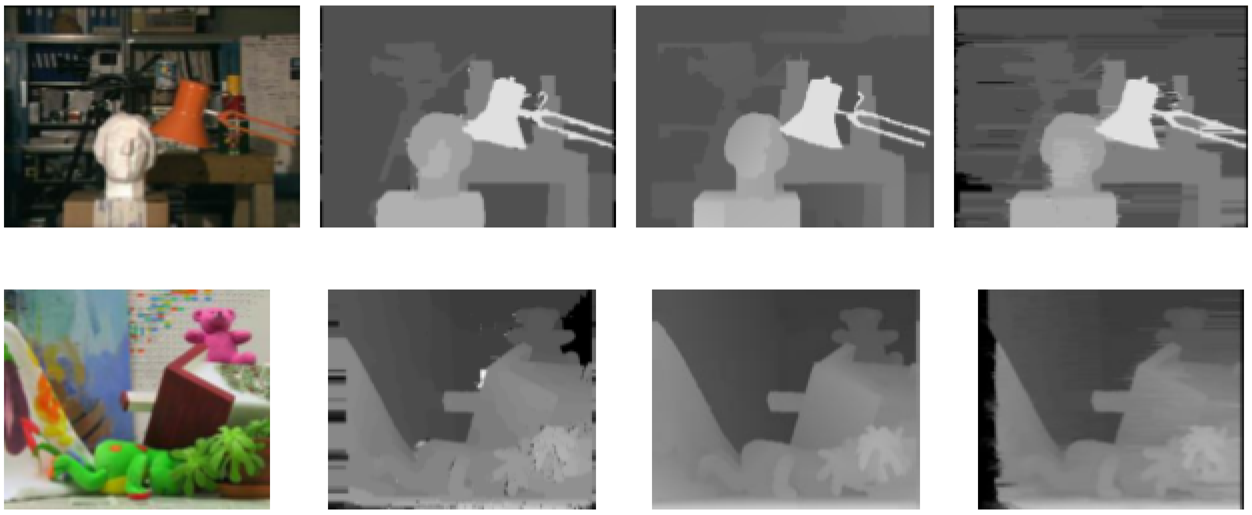
$$E(d) = E_{data}(d) + E_{smooth}(d)$$

- The data term $E_{data}$ measure how well the assignment fits to the stereo pair (in terms of overall matching cost). Several approaches rely on simple pixel-based cost functions but effective support aggregation strategies have been successfully adopted
- The smoothness/regularization $E_{smooth}$ term explicitly enforces piecewise assumptions (continuity) about the scene. This term penalizes disparity variations and large variation are allowed only at (unknown) depth borders. Plausibility of depth border is often related to edges.

Stereo matching

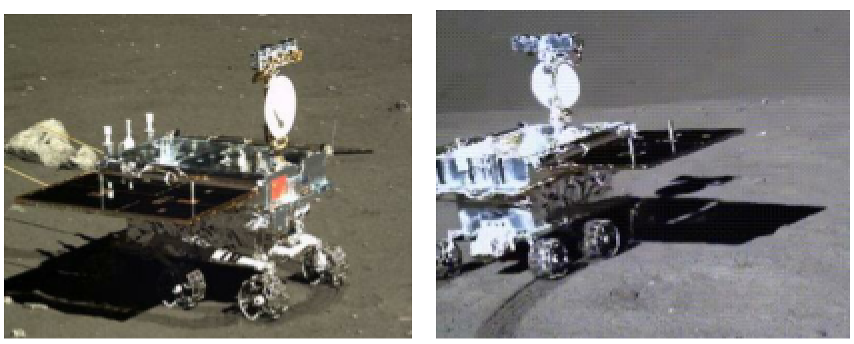
实验效果举例 Middlebury Stereo Evaluation

左视图 Graph cut BP + segmentation DP

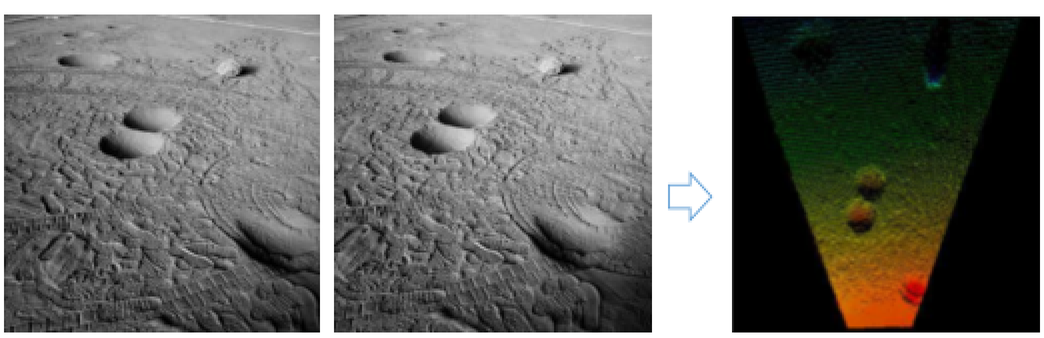


vision.middlebury.edu 由 Daniel Scharstein 和 Richard Szeliski 及其他研究人员维护。Middlebury Stereo Vision Page 主要提供立体匹配算法的在线评价和数据下载服务。它由《A taxonomy and evaluation of dense two-frame stereo correspondence algorithms》这篇文章发展而来。





嫦娥三号月面巡视器



探月工程，巡视器地形三维重建

# References


Basic reading:
•Szeliski textbook
•Boles et al., "Epipolar-plane image analysis: An approach to determining structure from motion," IJCV 1987.
- This classical paper introduces EPIs, and discusses how they can be used to infer depth.
•Lanman and Taubin, "Build Your Own 3D Scanner: Optical Triangulation for Beginners," SIGGRAPH course 2009.
- This very comprehensive course has everything you need to know about 3D scanning using structured light, including details on how to build your own.
•Bouguet and Perona, "3D Photography Using Shadows in Dual-Space Geometry," IJCV 1999.
- This paper introduces the idea of using shadows to do structured light 3D scanning, and shows an implementation using just a camera, desk lamp, and a stick.

Additional reading:
•Gupta et al., "A Practical Approach to 3D Scanning in the Presence of Interreflections, Subsurface Scattering and Defocus," IJCV 2013.
- This paper has a very detailed treatment of standard patterns used for structured light, problems arising due to global illumination, and robust patterns for dealing with these patterns.
•Barron et al., "Fast bilateral-space stereo for synthetic defocus," CVPR 2015.
•Barron and Poole, "The fast bilateral solver," ECCV 2016.
- The above two papers show how to combine edge-aware filtering (and bilateral filtering in particular) with disparity matching for robust stereo. The first paper also shows how the resulting depth maps can be used to create synthetic defocus blur.
•Wanner and Goldluecke, "Globally Consistent Depth Labeling of 4D Light Fields," CVPR 2012.
•Kim et al., "Scene reconstruction from high spatio-angular resolution light fields," SIGGRAPH 2013.
- These two papers show detailed systems for using EPIs to extract depth.
•Levin et al., "Understanding camera trade-offs through a Bayesian analysis of light field projections," ECCV 2008.
- This paper uses EPIs to show how different types of imaging systems (pinhole cameras, plenoptic cameras, stereo pairs, lens-based systems, and so on) relate to each other, and analyze their pros and cons for 3D imaging.




# Bouguet's Algorithm in OpenCV

## stereoRectify

Computes rectification transforms for each head of a calibrated stereo camera.

**C++:** void **stereoRectify**(InputArray **cameraMatrix1**, InputArray **distCoeffs1**, InputArray **cameraMatrix2**, InputArray **distCoeffs2**, Size **imageSize**, InputArray **R**, InputArray **T**, OutputArray **R1**, OutputArray **R2**, OutputArray **P1**, OutputArray **P2**, OutputArray **Q**, int **flags**=CALIB_ZERO_DISPARITY, double **alpha**=-1, Size **newImageSize**=Size(), Rect* **validPixROI1**=0, Rect* **validPixROI2**=0 )

**Parameters:**
- **cameraMatrix1** – First camera matrix.
- **cameraMatrix2** – Second camera matrix.
- **distCoeffs1** – First camera distortion parameters.
- **distCoeffs2** – Second camera distortion parameters.
- **imageSize** – Size of the image used for stereo calibration.
- **R** – Rotation matrix between the coordinate systems of the first and the second cameras.
- **T** – Translation vector between coordinate systems of the cameras.
- **R1** – Output 3x3 rectification transform (rotation matrix) for the first camera.
- **R2** – Output 3x3 rectification transform (rotation matrix) for the second camera.
- **P1** – Output 3x4 projection matrix in the new (rectified) coordinate systems for the first camera.
- **P2** – Output 3x4 projection matrix in the new (rectified) coordinate systems for the second camera.

## initUndistortRectifyMap

Computes the undistortion and rectification transformation map.

**C++:** void **initUndistortRectifyMap**(InputArray **cameraMatrix**, InputArray **distCoeffs**, InputArray **R**, InputArray **newCameraMatrix**, Size **size**, int **m1type**, OutputArray **map1**, OutputArray **map2**)

## remap

Applies a generic geometrical transformation to an image.

**C++:** void **remap**(InputArray **src**, OutputArray **dst**, InputArray **map1**, InputArray **map2**, int **interpolation**, int **borderMode**=BORDER_CONSTANT, const Scalar& **borderValue**=Scalar())