

Computer Vision Systems Programming VO

3D Vision

Christopher Pramerdorfer

Computer Vision Lab, Vienna University of Technology

Topics

Image formation

3D data acquisition



Images from wikipedia.org, createiveapplications.net

Motivation

Many CV applications rely on knowledge of scene geometry

This lecture covers

- ▶ How scene geometry and images are related
- ▶ How this relation can be used to recover scene geometry

Image Formation

Pinhole Camera Model

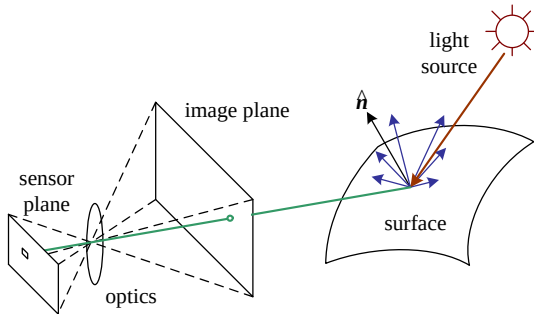


Image from Szeliski 2010

Image Formation

Pinhole Camera Model

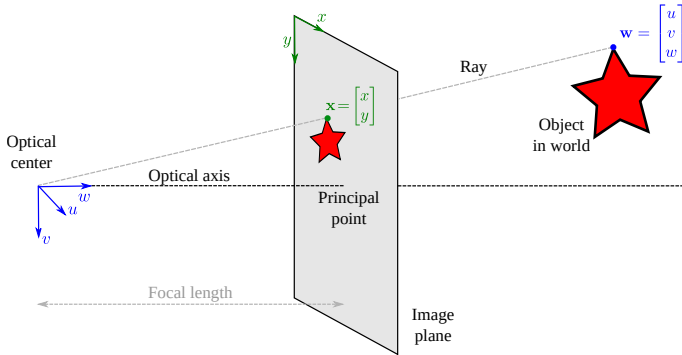


Image adapted from Prince 2012

Image Formation

Pinhole Camera Model

We obtain $x = fu/w + p_x$, $y = fv/w + p_y$

- ▶ f : focal length in pixels
- ▶ p_x, p_y : image coordinates of the principal point

This mapping is linear in **homogeneous coordinates**

$$\lambda \tilde{\mathbf{x}} = (\mathbf{\Lambda} \quad \mathbf{0}) \tilde{\mathbf{w}}$$
$$\lambda \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} u \\ v \\ w \\ 1 \end{pmatrix}$$

Image Formation

Pinhole Camera Model

World and camera coordinate systems generally differ

- Transform \mathbf{w} to camera coordinates before projection

$$\mathbf{w}' = \mathbf{\Omega} \mathbf{w} + \boldsymbol{\tau}$$
$$\begin{pmatrix} u' \\ v' \\ w' \end{pmatrix} = \begin{pmatrix} \omega_{11} & \omega_{12} & \omega_{13} \\ \omega_{21} & \omega_{22} & \omega_{23} \\ \omega_{31} & \omega_{32} & \omega_{33} \end{pmatrix} \begin{pmatrix} u \\ v \\ w \end{pmatrix} + \begin{pmatrix} \tau_u \\ \tau_v \\ \tau_w \end{pmatrix}$$

Image Formation

Pinhole Camera Model

We obtain the full **pinhole camera model**

$$\lambda \tilde{\mathbf{x}} = (\mathbf{\Lambda} \quad \mathbf{0}) \begin{pmatrix} \mathbf{\Omega} & \boldsymbol{\tau} \\ \mathbf{0}^\top & 1 \end{pmatrix} \tilde{\mathbf{w}}$$

Standard camera model in CV

- ▶ Usually together with radial distortion correction

Approximation to actual image formation

- ▶ In practice \mathbf{w} is not mapped to a single \mathbf{x}

Computing Scene Geometry

We can obtain \mathbf{w} by inverting the pinhole camera model

- ▶ But we don't know w

To this end, we must

- ▶ Utilize information from multiple images
- ▶ Use sensors that capture w directly

Computing Scene Geometry

Stereo



Image by John Kratz / flickr

Computing Scene Geometry

Stereo

In **stereo reconstruction** we have

- ▶ Point correspondences $\{(\mathbf{x}_1, \mathbf{x}_2)\}$ in two images
- ▶ Taken with calibrated cameras (known $\mathbf{\Lambda}, \mathbf{\Omega}, \tau$)

Goal is to estimate corresponding world coordinates \mathbf{w}

- ▶ Accomplished via triangulation

Computing Scene Geometry

Stereo

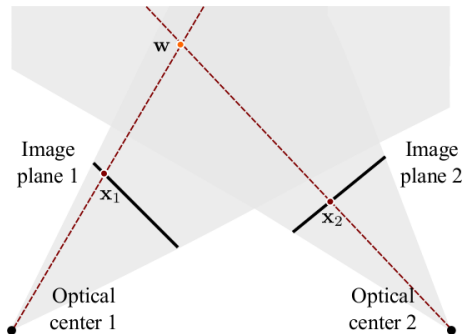


Image from Prince 2012

Computing Scene Geometry

Stereo

The challenge is finding correspondences

We typically want

- ▶ Many correspondences
- ▶ High accuracy and low noise

Usually accomplished via

- ▶ Dense feature matching along epipolar lines
- ▶ Followed by local or global optimization

Computing Scene Geometry

Stereo

x_1 must lie on the **epipolar line**

- ▶ Given by x_0 and the camera parameters

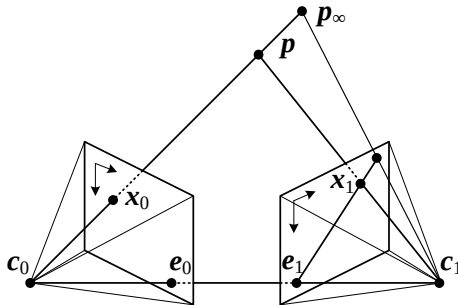


Image from Szeliski 2010

Computing Scene Geometry

Stereo

Images are **rectified** before correspondence search

- ▶ Relation between x -offset (**disparity** d) and w , $d = fb/w$
- ▶ b is the distance between the cameras

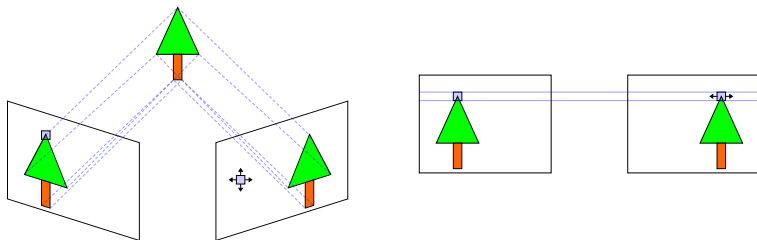


Image adapted from wikipedia.org

Computing Scene Geometry

Stereo

Dense matching on rectified images results in a **disparity map**

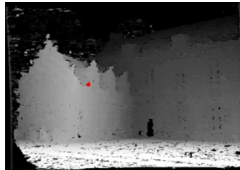


Image from Guido Gerig's slides

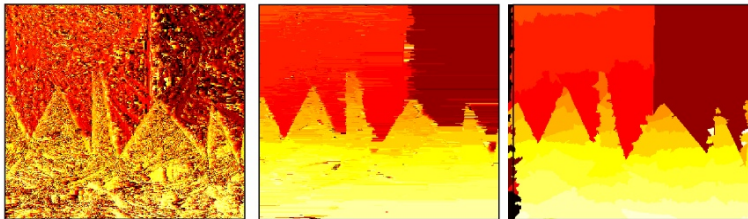
Computing Scene Geometry

Stereo

Raw disparity maps are noisy

Quality can be improved by encouraging smoothness

Accomplished via graphical models (e.g. MRFs)



Images from Prince 2012

Computing Scene Geometry

Stereo

Limitations of image-based (passive) stereo

- ▶ No correspondences in regions without texture
- ▶ Relies on proper illumination (no dark living rooms)
- ▶ Computational complexity

Alternatively, we can use sensors that capture w directly

- ▶ Usually together with brightness or color

These **depth sensors**

- ▶ Do not rely on texture
- ▶ Work under dim or dark conditions
- ▶ Save computational resources

Computing Scene Geometry

Depth Sensors – Kinect v1

Released by Microsoft for Xbox 360 in late 2010

Fastest selling consumer electronics device to date



Image from wikipedia.org

Computing Scene Geometry

Depth Sensors – Kinect v1

Works by replacing one sensor with an IR source

- Projects a speckle pattern onto objects

<https://www.youtube.com/watch?v=t5joFtzEYpo>

Pattern is observed using an IR sensor

Stereo (epipolar) geometry still applies

- Shift between patterns corresponds to d respectively w

Does not work in sunlight due to IR radiation

Computing Scene Geometry

Depth Sensors – Kinect v1



Image from <https://www.youtube.com/watch?v=o0CMr0D7BqY>

Computing Scene Geometry

Depth Sensors – Kinect v2

Released by Microsoft in late 2013



Image from wikipedia.org

Computing Scene Geometry

Depth Sensors – Kinect v2

Works based on the time of flight principle

- ▶ A light pulse is emitted at time t_0
- ▶ Pulse is reflected and observed at time t_1
- ▶ w proportional to delay, $t_1 - t_0 = 2w/c$

Computing Scene Geometry

Depth Sensors – Kinect v2

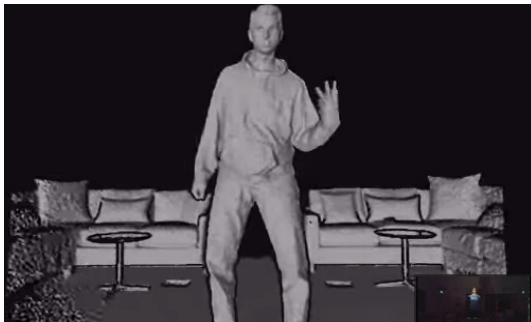


Image from <https://www.youtube.com/watch?v=Ja01Ua57BWs>

Computing Scene Geometry

We now have an image encoding w (**depth map**)

- ▶ Can be generated from disparity map as shown above

We can use this information to obtain points in the world \mathbf{w}

- ▶ By inverting the pinhole camera model
- ▶ Resulting in a **point cloud**

Computing Scene Geometry



Image from creativeapplications.net

Summary

We have covered

- ▶ How the scene geometry and images are related
- ▶ Means for estimating point distances w
- ▶ How scene geometry can be recovered on this basis

This information enables interesting CV applications

- ▶ Next, we will go over some examples

3D Vision Lecture

Interested in 3D vision?

- ▶ There is an own VO (183.129) and UE (183.130)

Prince, S.J.D. (2012). **Computer Vision: Models Learning and Inference**. Cambridge University Press.

Szeliski, Richard (2010). **Computer vision: algorithms and applications**. Springer.