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

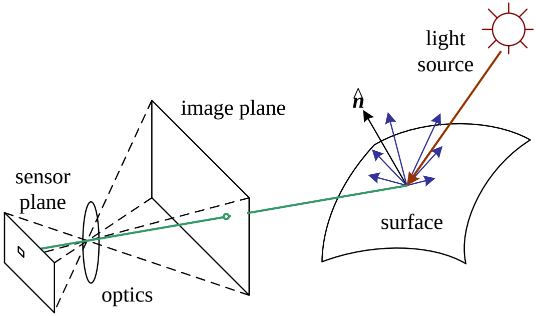


Image from Szeliski 2010

Image Formation
Pinhole Camera Model

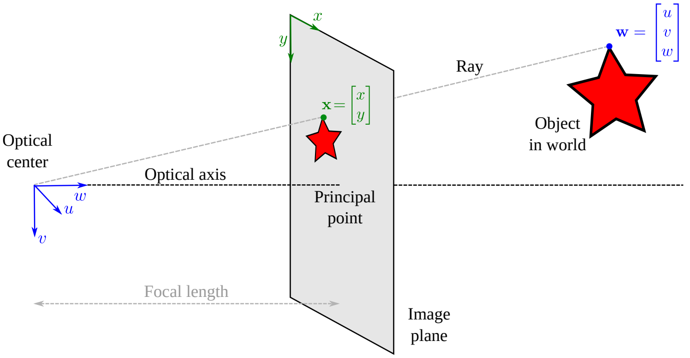


Image adapted from Prince 2012

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**

$$\lambda\tilde{\mathbf{x}} = \begin{pmatrix}\mathbf{\Lambda} & \mathbf{0}\end{pmatrix}\tilde{\mathbf{w}}$$

$$\lambda\begin{pmatrix}x\\y\\1\end{pmatrix} = \begin{pmatrix}f & 0 & p_x & 0\\0 & f & p_y & 0\\0 & 0 & 1 & 0\end{pmatrix}\begin{pmatrix}u\\v\\w\\1\end{pmatrix}$$

World and camera coordinate systems generally differ

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

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

We obtain the full **pinhole camera model**

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

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

Image by John Kratz / flickr

In **stereo reconstruction** we have

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

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

- Accomplished via triangulation

Computing Scene Geometry
Stereo

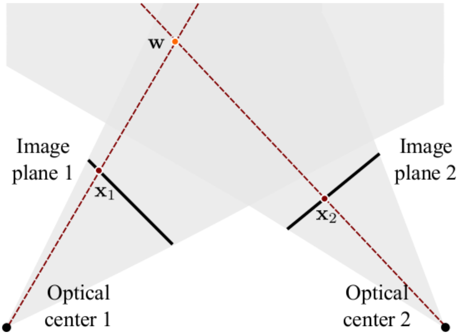


Image from Prince 2012

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

$\mathbf{x}_1$ must lie on the **epipolar line**

- Given by $\mathbf{x}_0$ and the camera parameters

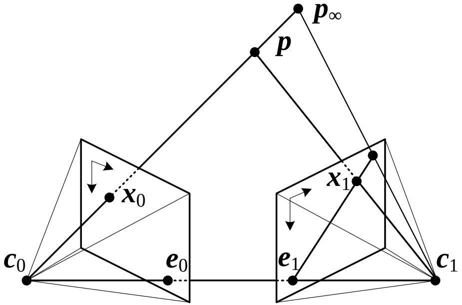


Image from Szeliski 2010

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

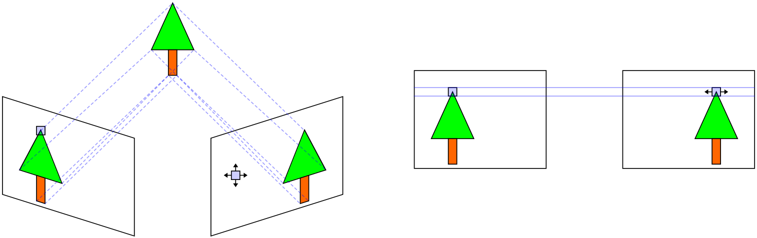


Image adapted from wikipedia.org

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

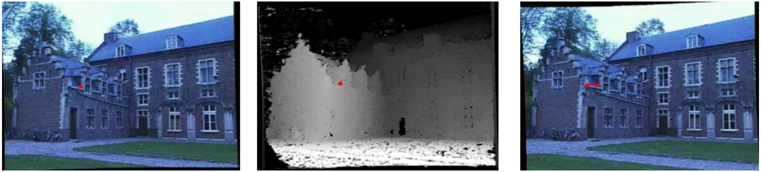


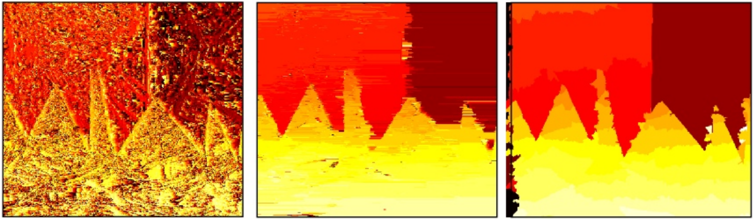
Image from Guido Gerig's slides

Computing Scene Geometry
Stereo

Raw disparity maps are noisy

Quality can be improved by encouraging smoothness

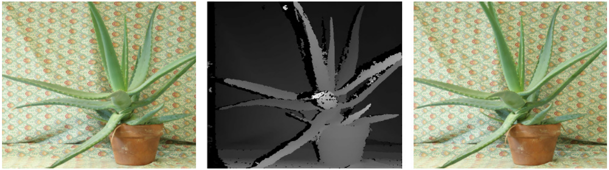
Accomplished via graphical models (e.g. MRFs)



Images from Prince 2012

Stereo matching in Python using OpenCV

```
imgL = cv2.pyrDown(cv2.imread('left.jpg'))
imgR = cv2.pyrDown(cv2.imread('right.jpg'))
stereo = cv2.StereoSGBM(...) # args depend on images
disparity = stereo.compute(imgL, imgR)
```



Limitations of image-based (passive) stereo

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

Computing Scene Geometry
Depth Sensors

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

Released by Microsoft for Xbox 360 in late 2010

Fastest selling consumer electronics device to date

Image from wikipedia.org

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=oOCMr0D7BqY

Released by Microsoft in late 2013

Image from wikipedia.org

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, $t_1 - t_0 = 2w/c$

Image from https://www.youtube.com/watch?v=JaOlUa57BWs

There are several libraries for accessing Kinect data

- Kinect SDK (http://microsoft.com/en-us/kinectforwindows/)
- OpenNI2 (https://github.com/occipital/openni2)
- libfreenect2 (https://github.com/OpenKinect/libfreenect2)

Access depth maps using OpenNI2 & OpenCV

```
using namespace openni;

OpenNI::initialize();
Device dev; dev.open(ANY_DEVICE); // open any connected sensor
VideoStream s; s.create(dev, SENSOR_DEPTH); s.start();

VideoFrameRef fr; s.readFrame(&fr); // read frame

// convert for use with OpenCV
cv::Mat_<ushort> f(fr.getHeight(), fr.getWidth());
const DepthPixel *px = (const DepthPixel*) fr.getData();
std::memcpy(f.data, px, fr.getHeight() * fr.getWidth() * sizeof(DepthPixel));
```

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

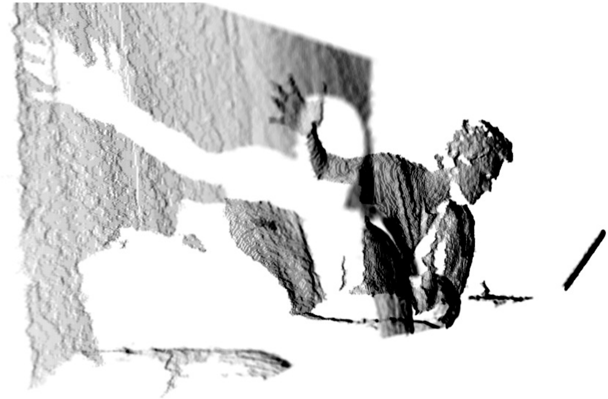


Image from `creativeapplications.net`

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

Interested in 3D vision?

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

Bibliography

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

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