High Speed Camera & IMUs on Mobile Devices

Instructor - Simon Lucey

16-623 - Designing Computer Vision Apps
Today

• CCD vs CMOS cameras.
• Rolling Shutter Epipolar Geometry
• Inertial Measurement Units (IMU)
Let us begin by considering a mathematical description of the imaging process through this idealized camera. We will consider issues like lens distortion subsequently.

The pinhole camera or the projective camera as it is known images the scene by applying a perspective projection to it. In the following we shall refer to scene coordinates with upper case roman letters, \( \{X, Y, Z, \ldots\} \). Image coordinates will be referred to using lower case roman letters, \( \{x, y, z, \ldots\} \).

Vectors shall be denoted by boldfaced symbols, e.g., \( \mathbf{x} \) or \( \mathbf{X} \). (In class, when writing on the blackboard, I will put a tilde underneath the corresponding symbols to denote a vector.)

The scene is three dimensional, whereas the image is located in a two dimensional plane. Hence the perspective projection maps the 3D space to a 2D plane.

\[
\begin{align*}
\mathbf{x} &= f \frac{X}{Z} \\
y &= f \frac{Y}{Z}
\end{align*}
\]

Here, \( f \) is the focal length of the camera, i.e., the distance between the image plane and the pinhole.

The process is illustrated in figure 2.

(Taken from Forsyth & Ponce)
Pinhole Camera

Let us begin by considering a mathematical description of the imaging process through this idealized camera. We will consider issues like lens distortion subsequently.

The pinhole camera or the projective camera as it is known images the scene by applying a perspective projection to it. In the following we shall refer to scene coordinates with upper case roman letters, \{X, Y, Z, \ldots\}. Image coordinates will be referred to using lower case roman letters, \{x, y, z, \ldots\}.

Vectors shall be denoted by boldfaced symbols, e.g., \(\mathbf{X}\) or \(\mathbf{x}\). (In class, when writing on the blackboard, I will put a tilde underneath the corresponding symbols to denote a vector.)

The scene is three dimensional, whereas the image is located in a two dimensional plane. Hence the perspective projection maps the 3D space to a 2D plane.

\[
\begin{align*}
X & = f X Z \\
y & = f Y Z
\end{align*}
\]

here, \(f\) is the focal length of the camera, i.e., the distance between the image plane and the pinhole.

The process is illustrated in figure 2.

(Taken from Forsyth & Ponce)
Digital Cameras

• All digital cameras rely on the photoelectric effect to create electrical signal from light.
• CCD (charge coupled device) and CMOS (complementary metal oxide semiconductor) are the two most common image sensors found in digital cameras.
• Both invented in the late 60s early 70s.

(Taken from https://www.teledynedalsa.com/imaging/knowledge-center/appnotes/ccd-vs-cmos/)
CCD versus CMOS

• CMOS and CCD imagers differ in the way that signals are converted from signal charge.

• CMOS imagers are inherently more parallel than CCDs.

• Consequently, high speed CMOS imagers can be designed to have much lower noise than high speed CCDs.

(Taken from https://www.teledynedalsa.com/imaging/knowledge-center/appnotes/ccd-vs-cmos/)
CCD versus CMOS

- CCD used to be the image sensor of choice as it gave far superior images with the fabrication technology available.
- CMOS was of interest with the advent of mobile phones.
  - CMOS promised lower power consumption.
  - Lowered fabrication costs (reuse mainstream logic and memory device fabrication).
- An enormous amount of investment was made to develop and fine-tune CMOS imagers.
- As a result, we witnessed great improvements in image quality, even as pixel sizes shrank.
- In the case of high volume consumer area imagers, CMOS imagers outperform CCDs based on almost every performance parameter.

(Taken from https://www.teledynedalsa.com/imaging/knowledge-center/appnotes/ccd-vs-cmos/)
New Developments - iPhone 7

• Apple just released the iPhone 7 with new dual lens camera.
• Rumored that advances in the camera are based on the 2015 acquisition of Linx (Israeli startup).
• Image quality “closest” attempt yet to DSLR on mobile device.

Taken from: http://vrscout.com/news/apple-duel-camera-iphone-for-augmented-reality/
Today

- CCD vs CMOS cameras.
- Rolling Shutter Epipolar Geometry
- Inertial Measurement Units (IMU)
Rolling Shutter Effect

Rolling shutter cameras sequentially expose rows.

\[ t_r + t_{id} = \frac{1}{\text{frames per second}} \]
Global versus Rolling Shutter

In each frame and of a rolling shutter camera, where at a different time. Fig. 2 illustrates the image capture moment model, tracked feature points in consecutive frames that uses visual and inertial measurements. In our motion estimation method for rolling-shutter camera considered in SLAM before. Our algorithm is the first EKF estimation between consecutive frames can be measured [9].

transformation under pure rotational motion, the relative for the estimation of the camera motion. Based on the fact not use the accelerometer readings as orientation measurement. Fortunately, camera shake and rolling shutter effects are still ill-posed since it is impossible to obtain depth information. Even if we can obtain accurate camera from inertial measurement sensors. The readings of acceler gravity is approximately the only source in the accelerometer readings to estimate the camera rotation estimated by integrating the gyroscope readings (angular velocity).

sufficient to produce satisfactory videos. have shown that taking only camera rotations into account is deriving [7] have been applied to approximate the stabilization.

Our 3-D orientation estimation is also based on EKF, but considering [16] have been applied to approximate the stabilization.

A. State Vector and Dynamic Bayesian Network

Generally, camera motion can be described by position and orientation parameters. The camera position is usually modeled by a point in 3D space, while the orientation is typically represented by a rotation matrix or a quaternion.

IV. Online Motion Estimation

Our online motion estimation is based on EKF. Due to

Taking from: Jia and Evans “Probabilistic 3-D Motion Estimation for Rolling Shutter Video Rectification from Visual and Inertial Measurements” MMSP 2012.
Global versus Rolling Shutter

In each frame and therefore, our algorithm can be classified as a relative motion estimation method for rolling-shutter cameras. Simultaneous localization and mapping (SLAM) in robotics that matched feature points can be related by a homographic matrix caused by rotation are sometimes non-negligible. Thus we do our measurement model is quite different from [5]. We find as measurements of the camera rotation.

Gravity is approximately the only source in the accelerometer. Users usually try to hold the camera in a steady position so that they are used to compute the camera rotations for the inertial measurement sensors. The readings of accelerometers capture not only linear acceleration of cameras, but the orientations. Another recent approach [5] uses both gyroscope estimated by integrating the gyroscope readings (angular velocity) and accelerometer readings. Camera shake and rolling shutter effects are due to the fact that the idle time is large enough so that the homography matrix is.

Fig. 2. Rolling shutter cameras sequentially expose rows.

\[ u' \rightarrow h(u, t) \rightarrow u \]

Assume the intrinsic camera matrix is \( \mathbf{K} \), the equations can be written as:

\[ \mathbf{u}' = \mathbf{H}(u, t) \mathbf{u} \]

Our 3-D orientation estimation is also based on EKF, but its measurement model is quite different from [5]. We find as measurements of the camera rotation.

\[ \mathbf{u}' \rightarrow \mathbf{H}(u, t) \rightarrow \mathbf{u} \]

Fortunately, camera shake and rolling shutter effects are.

Taken from: Jia and Evans “Probabilistic 3-D Motion Estimation for Rolling Shutter Video Rectification from Visual and Inertial Measurements” MMSP 2012.
Rolling-Shutter Effect

- A drawback to CMOS sensors is the “rolling-shutter effect”.
- CMOS captures images by scanning one line of the frame at a time.
- If anything is moving fast, then it will lead to weird distortions in still photos, and to rather odd effects in video.
- Check out the following video taken with the iPhone 4 CCD camera.
- CCD-based cameras often use a “global” shutter to circumvent this problem.

Taken from: http://www.wired.com/2011/07/iphones-rolling-shutter-captures-amazing-slo-mo-guitar-string-vibrations/
Rolling-Shutter Effect

- A drawback to CMOS sensors is the “rolling-shutter effect”.
- CMOS captures images by scanning one line of the frame at a time.
- If anything is moving fast, then it will lead to weird distortions in still photos, and to rather odd effects in video.
- Check out the following video taken with the iPhone 4 CCD camera.
- CCD-based cameras often use a “global” shutter to circumvent this problem.

Rolling Shutter Effect = “Aliasing”

- Rolling Shutter Effect is an example of a broader phenomena regularly studied in Signal Processing called “Aliasing”.
- Common phenomenon
  - Wagon wheels rolling the wrong way in movies.
Rolling Shutter Effect = “Aliasing”

• Rolling Shutter Effect is an example of a broader phenomena regularly studied in Signal Processing called “Aliasing”.

• Common phenomenon
  • Wagon wheels rolling the wrong way in movies.
Rectifying Rolling Shutter

- What do you think the camera motion was here?

Taken from: Hanning et al., “Stabilizing Cell Phone Video using Inertial Measurement Sensors” in ICCV 2011 Workshop.
High-Frame Rate Cameras

• Another way around this is to create higher-frame rate cameras.

• Increasingly seeing faster and faster CMOS cameras.

• Opening up other exciting opportunities in computer vision.

• However, really fast motions still need an understanding of the rolling shutter effect.
High-Frame Rate Cameras

• Another way around this is to create higher-frame rate cameras.
• Increasingly seeing faster and faster CMOS cameras.
• Opening up other exciting opportunities in computer vision.
• However, really fast motions still need an understanding of the rolling shutter effect.
Rectifying Rolling Shutter

• Result from rectification,

Taken from: Hanning et al., “Stabilizing Cell Phone Video using Inertial Measurement Sensors” in ICCV 2011 Workshop.
## Reminder: Cheat Sheet

<table>
<thead>
<tr>
<th>Description</th>
<th>Hartley &amp; Zisserman</th>
<th>Prince</th>
</tr>
</thead>
<tbody>
<tr>
<td>3D Point</td>
<td>X</td>
<td>W</td>
</tr>
<tr>
<td>2D Point</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Rotation matrix</td>
<td>R</td>
<td>Ω</td>
</tr>
<tr>
<td>Intrinsics matrix</td>
<td>K</td>
<td>Λ</td>
</tr>
<tr>
<td>Homography matrix</td>
<td>H</td>
<td>Φ</td>
</tr>
<tr>
<td>translation vector</td>
<td>t</td>
<td>τ</td>
</tr>
</tbody>
</table>
Reminder: The Essential Matrix

First camera:
\[ \lambda_1 \tilde{x}_1 = w \]

Second camera:
\[ \lambda_2 \tilde{x}_2 = \Omega w + \tau \]

Substituting:
\[ \lambda_2 \tilde{x}_2 = \lambda_1 \Omega \tilde{x}_1 + \tau \]

This is a mathematical relationship between the points in the two images, but it’s not in the most convenient form.

Adapted from: Computer vision: models, learning and inference. Simon J.D. Prince
Reminder: The Essential Matrix

\[ \lambda_2 \tilde{x}_2 = \lambda_1 \Omega \tilde{x}_1 + \tau \]

\[ \lambda_2 \tau \times \tilde{x}_2 = \lambda_1 \tau \times \Omega \tilde{x}_1 \]

\[ \tilde{x}_2^T \tau \times \Omega \tilde{x}_1 = 0 \]
The cross product term can be expressed as a matrix

\[
\hat{x}_2^T \tau \times \Omega \hat{x}_1 = 0
\]

Defining:

\[
\tau \times = \begin{bmatrix}
0 & -\tau_z & \tau_y \\
\tau_z & 0 & -\tau_x \\
-\tau_y & \tau_x & 0
\end{bmatrix}
\]

\[
E = \tau \times \Omega
\]

We now have the essential matrix relation

\[
\hat{x}_2^T E \hat{x}_1 = 0
\]
Recently Dai et al. (2016) developed Generalized Epipolar Geometry for Rolling Shutter Camera.

Assuming linear rolling shutter,

\[ \lambda_1 \tilde{x}_1 = w + \nu_1 d_1 \]
\[ \lambda_2 \tilde{x}_2 = \Omega w + \tau + \nu_2 d_2 \]

\( \nu \rightarrow \) index to the scan line in the image
\( d_i \rightarrow 3D \) velocity for i-th viewpoint

Epipolar Geometry for Rolling Shutter

- Results in a different essential matrix for every possible combination of $\nu_1$ and $\nu_2$.

\[ E(\nu_1, \nu_2) = (\tau + \nu_2 d_2 - \nu_1 \Omega d_1) \times \Omega \]
Epipolar Geometry for Rolling Shutter

- Results in a different essential matrix for every possible combination of $\nu_1$ and $\nu_2$.

$$E(\nu_1, \nu_2) = (\tau + \nu_2 d_2 - \nu_1 \Omega d_1) \times \Omega$$

How many degrees of freedom?

## Epipolar Geometry for Rolling Shutter

<table>
<thead>
<tr>
<th>Camera Model</th>
<th>Essential Matrix</th>
<th>Monomials</th>
<th>Degree-of-freedom</th>
<th>Linear Algorithm</th>
<th>Non-linear Algorithm</th>
<th>Motion Parameters</th>
</tr>
</thead>
</table>
| Perspective camera    | \[
|                       | \begin{bmatrix} f_{11} & f_{12} & f_{13} \\
|                       | f_{21} & f_{22} & f_{23} \\
|                       | f_{31} & f_{32} & f_{33} \end{bmatrix} | (u_1, v_1, 1)     | 3^2 = 9          | 8-point             | 5-point            | R, t              |
| Linear push broom     | \[
|                       | \begin{bmatrix} 0 & 0 & f_{13} & f_{14} \\
|                       | 0 & 0 & f_{23} & f_{24} \\
|                       | f_{31} & f_{32} & f_{33} & f_{34} \\
|                       | f_{41} & f_{42} & f_{43} & f_{44} \end{bmatrix} | (u_i v_i, u_i, v_i, 1) | 12 = 4^2 - 2^2 | 11-point           | 11-point           | R, t, d_1, d_2    |
| Linear rolling shutter | \[
|                         | \begin{bmatrix} 0 & 0 & f_{13} & f_{14} & f_{15} \\
|                         | 0 & 0 & f_{23} & f_{24} & f_{25} \\
|                         | f_{31} & f_{32} & f_{33} & f_{34} & f_{35} \\
|                         | f_{41} & f_{42} & f_{43} & f_{44} & f_{45} & f_{46} \\
|                         | f_{51} & f_{52} & f_{53} & f_{54} & f_{55} & f_{56} \end{bmatrix} | (u_i^2 v_i u_i^2, u_i, v_i, 1) | 21 = 5^2 - 2^2 | 20-point           | 11-point           | R, t, d_1, d_2    |
| Uniform push broom    | \[
|                         | \begin{bmatrix} 0 & 0 & f_{13} & f_{14} & f_{15} & f_{16} \\
|                         | 0 & 0 & f_{23} & f_{24} & f_{25} & f_{26} \\
|                         | f_{31} & f_{32} & f_{33} & f_{34} & f_{35} & f_{36} \\
|                         | f_{41} & f_{42} & f_{43} & f_{44} & f_{45} & f_{46} & f_{47} \\
|                         | f_{51} & f_{52} & f_{53} & f_{54} & f_{55} & f_{56} & f_{57} \\
|                         | f_{61} & f_{62} & f_{63} & f_{64} & f_{65} & f_{66} & f_{67} & f_{68} & f_{69} & f_{70} & f_{71} & f_{72} & f_{73} \end{bmatrix} | (u_i^2 v_i^2 u_i^2, u_i, v_i, 1) | 32 = 6^2 - 2^2 | 31-point           | 17-point           | R, t, w_1, w_2, d_1, d_2 |
| Uniform rolling shutter | \[
|                         | \begin{bmatrix} 0 & 0 & f_{13} & f_{14} & f_{15} & f_{16} & f_{17} \\
|                         | 0 & 0 & f_{23} & f_{24} & f_{25} & f_{26} & f_{27} \\
|                         | f_{31} & f_{32} & f_{33} & f_{34} & f_{35} & f_{36} & f_{37} \\
|                         | f_{41} & f_{42} & f_{43} & f_{44} & f_{45} & f_{46} & f_{47} & f_{48} & f_{49} & f_{50} & f_{51} & f_{52} & f_{53} & f_{54} & f_{55} & f_{56} & f_{57} \\
|                         | f_{61} & f_{62} & f_{63} & f_{64} & f_{65} & f_{66} & f_{67} & f_{68} & f_{69} & f_{70} & f_{71} & f_{72} & f_{73} & f_{74} & f_{75} & f_{76} & f_{77} \end{bmatrix} | (u_i^3 v_i^3 u_i^3, u_i^2, v_i^2, u_i, v_i, 1) | 45 = 7^2 - 2^2 | 44-point           | 17-point           | R, t, w_1, w_2, d_1, d_2 |

Accessing the Camera in iOS

```c
// ViewController.m
// Camera_AvFoundation
//
// Created by Simon Lucey on 9/7/16.
// Copyright © 2016 CMU_16623. All rights reserved.
//

#import "ViewController.h"
#include <iostream>

@interface ViewController()
@end

@implementation ViewController

- (void)viewDidLoad {
    [super viewDidLoad];
    // Do any additional setup after loading the view, typically from a nib.

    // Initialize the view
    imageView_ = [[UIImageView alloc] initWithFrame:CGRectMake(0.0, 0.0, self.view.frame.size.width, self.view.frame.size.height)];
    [self.view addSubview:imageView_];

    // Initialize the video camera
    self.videoCamera = [[CvVideoCamera alloc] initWithParentView:imageView_];
    self.videoCamera.delegate = self;
    self.videoCamera.defaultAVCaptureDevicePosition = AVCaptureDevicePositionFront;
    self.videoCamera.defaultAVCaptureSessionPreset = AVCaptureSessionPreset640x480;
    self.videoCamera.defaultAVCaptureVideoOrientation = AVCaptureVideoOrientationPortrait;
    self.videoCamera.defaultFPS = 30;

    // Finally show the output
    [videoCamera start];
    isCapturing = YES;
}
```

Accessing the Camera in iOS

```c
// ViewController.m
// Camera_AvFoundation
// Created by Simon Lucey on 9/7/16.
// Copyright © 2016 CMU_16623. All rights reserved.
//
#import "ViewController.h"
#include <iostream>

@interface ViewController();
@end

@implementation ViewController

-(void)viewDidLoad {
    [super viewDidLoad];
    // Do any additional setup after loading the view, typically from a nib.

    // Initialize the view
    imageView_ = [[[UIImageView alloc] initWithFrame:CGRectMake(0.0, 0.0, self.view.frame.size.width, self.view.frame.size.height)] autorelease];
    [self.view addSubview:imageView_];

    // Initialize the video camera
    self.videoCamera = [[[CvVideoCamera alloc] initWithParentView:imageView_] autorelease];
    self.videoCamera.delegate = self;
    self.videoCamera.defaultAVCaptureDevicePosition = AVCaptureDevicePositionFront;
    self.videoCamera.defaultAVCaptureSessionPreset = AVCaptureSessionPreset640x480;
    self.videoCamera.defaultAVCaptureVideoOrientation = AVCaptureVideoOrientationPortrait;
    self.videoCamera.defaultFPS = 30;

    // Finally show the output
    [videoCamera start];
    isCapturing = YES;
}
```
Accessing the Camera in iOS

```swift
// ViewController.m
// Camera_AvFoundation
//
// Created by Simon Lucey on 9/7/16.
// Copyright © 2016 CMU_16623. All rights reserved.
//
#import "ViewController.h"
#include <iostream>

@interface ViewController
@end

@implementation ViewController
-
(void)viewDidLoad {
    [super viewDidLoad];
    // Do any additional setup after loading the view, typically from a nib.

    // Initialize the view
    imageView_ = [[UIImageView alloc] initWithFrame.CGRectMake(0.0, 0.0, self.view.frame.size.width, self.view.frame.size.height)];
    [self.view addSubview:imageView_];

    // Initialize the video camera
    self.videoCamera = [[CvVideoCamera alloc] initWithParentView:imageView_];
    self.videoCamera.delegate = self;
    self.videoCamera AVCaptureDevicePosition = AVCaptureDevicePositionFront;
    self.videoCamera AVCaptureSessionPreset = AVCaptureSessionPreset640x480;
    self.videoCamera AVCaptureVideoOrientation = AVCaptureVideoOrientationPortrait;
    self.videoCamera.defaultFPS = 30;

    // Finally show the output
    [videoCamera start];
    isCapturing = YES;
}
```
Today

• CCD vs CMOS cameras.
• Rolling Shutter Epipolar Geometry
• Inertial Measurement Units (IMU)
Inertial Measurement Unit

- Measures a device’s specific force, angular rate & magnetic field.
- Composed of,
  - Accelerometer.
  - Gyroscope.
  - Magnetometer.
- Historically used heavily within navigation and robotic systems.
- More recently have become common place in smart devices.
Accelerometer
Accelerometer
Accelerometer

What can’t you measure?
Gyroscope
IMU Example in iOS

• Good example of using IMU in iOS can be found at,

https://github.com/nscookbook/recipe19

• Or better yet, if you have git installed you can type from the command line.

$ git clone https://github.com/NSCookbook/recipe19.git

• Good tutorial about how code works can be found at,

Accessing the IMU in iOS

```swift
-(void)viewDidLoad
{
    [super viewDidLoad];
    // Do any additional setup after loading the view, typically from a nib.
    currentMaxAccelX = 0;
    currentMaxAccelY = 0;
    currentMaxAccelZ = 0;

    currentMaxRotX = 0;
    currentMaxRotY = 0;
    currentMaxRotZ = 0;

    self.motionManager = [[CMMotionManager alloc] init];
    self.motionManager.accelerometerUpdateInterval = .2;
    self.motionManager.gyroUpdateInterval = .2;

    [self.motionManager startAccelerometerUpdatesToQueue:[NSOperationQueue currentQueue]
        withHandler:^(CMAccelerometerData *accelerometerData, NSError *error) {
            [self outputAccelerationData:accelerometerData.acceleration];
            if(error){
                NSLog(@"%@", error);
            }
        }];

    [self.motionManager startGyroUpdatesToQueue:[NSOperationQueue currentQueue]
        withHandler:^(CMGyroData *gyroData, NSError *error) {
            [self outputRotationData:gyroData.rotationRate];
        }];
```
Accessing the IMU in iOS
Accessing the IMU in iOS
Accessing the IMU in iOS
Mobile Solutions

- Tanskanen et al. - ETH Zurich
- Generates accurate point-cloud using SLAM (PTAM)
- Integrates IMU for scale

P. Tanskanen, K. Kolev, L. Meier, F. Camposeco, O. Saurer, M. Pollefeys: Live metric 3d reconstruction on mobile phones. (ICCV 2013)
Mobile Visual SLAM + IMU

P. Tanskanen, K. Kolev, L. Meier, F. Camposeco, O. Saurer, M. Pollefeys: Live metric 3d reconstruction on mobile phones. (ICCV 2013)
Mobile Visual SLAM + IMU

P. Tanskanen, K. Kolev, L. Meier, F. Camposeco, O. Saurer, M. Pollefeys: Live metric 3d reconstruction on mobile phones. (ICCV 2013)
Mobile Visual SLAM + IMU

P. Tanskanen, K. Kolev, L. Meier, F. Camposeco, O. Saurer, M. Pollefeys: Live metric 3d reconstruction on mobile phones. (ICCV 2013)
Estimated pupil distance vs. Seconds of data used

C. Ham, S. Singh, and S. Lucey: Handwaving away scale. (ECCV 2014)
C. Ham, S. Singh, and S. Lucey: Handwaving away scale. (ECCV 2014)
C. Ham, S. Singh, and S. Lucey: Handwaving away scale. (ECCV 2014)
C. Ham, S. Singh, and S. Lucey: Handwaving away scale. (ECCV 2014)
Mobile Platform Issues

- IMU and Camera time stamped differently

**IMU (System timestamps)**
- 1045 ns
- 1145 ns

**Camera (Relative timestamps)**
- 0 ns
- 100 ns

\[ \Delta t \]
Auto-Correlation

Unaligned Accelerometer Signals

Cross-correlation of Signals

C. Ham, S. Singh, and S. Lucey: Handwaving away scale. (ECCV 2014)