Introduction to Computer Vision

Week 2, Fall 2010
Instructor: Prof. Ko Nishino
Last Week

- What is Computer Vision
- History of Imaging
  - Camera Obscura
- Pin-hole Camera
- Thin-lens Law
- Aperture
  - Reciprocity of Aperture-Shutter Speed
- Defocus, Depth-of-Field
- Distortion
  - Geometric and Radiometric
How about our eyes?

- Index of refraction: cornea 1.376, aqueous 1.336, lens 1.406-1.386
- Iris is the diaphragm that changes the aperture (pupil)
- Retina is the sensor where the fovea has the highest resolution
Accommodation

Changes the focal length of the lens

shorter focal length
Myopia and Hyperopia

**Normal sight**

- *Focus point (fovea)*

**Nearsightedness** (myopia)

**Farsightedness** (hypermetropia)
Astigmatism

Image out of focus

Corrected image
Image Sensing

Reading: Robot Vision Chapter 2
Human Eye
## Rods and Cones

<table>
<thead>
<tr>
<th>Rods</th>
<th>Cones</th>
</tr>
</thead>
<tbody>
<tr>
<td>Achromatic: one type of pigment</td>
<td>Chromatic: three types of pigment</td>
</tr>
<tr>
<td>Slow response (long integration time)</td>
<td>Fast response (short integration time)</td>
</tr>
<tr>
<td>High amplification</td>
<td>Less amplification</td>
</tr>
<tr>
<td>High sensitivity</td>
<td>Lower absolute sensitivity</td>
</tr>
<tr>
<td>Low acuity</td>
<td>High acuity</td>
</tr>
</tbody>
</table>
Sensors

- Convert light into electric charge

- **CCD (charge coupled device)**
  - Higher dynamic range
  - High uniformity
  - Lower noise

- **CMOS (complementary metal Oxide semiconductor)**
  - Lower voltage
  - Higher speed
  - Lower system complexity
Sensor Readout

- **CCD**

- **CMOS**
Rolling Shutter

- Read out each row sequentially
- No explicit shutter; controlled sequential exposure
- CMOS
- CCD ≠ Rolling Shutter
- Introduces distortion for object moving faster than buffering limit
Coded Rolling Shutter Photography: Flexible Space-Time Sampling

Jinwei Gu

Abstract

We propose a novel readout architecture called coded rolling shutter for complementary metal-oxide semiconductor (CMOS) image sensors. Rolling shutter has traditionally been considered as a disadvantage to image quality since it often introduces skew artifact. In this paper, we show that by controlling the readout timing and the exposure length for each row, the row-wise exposure discrepancy in rolling shutter can be exploited to flexibly sample the 3D space-time volume of scene appearance, and can thus be advantageous for computational photography. The required controls can be readily implemented in standard CMOS sensors by altering the logic of the control unit.

We propose several coding schemes and applications: (1) coded readout allows us to better sample time dimension for high-speed photography and optical flow based applications; and (2) row-wise control enables capturing motion-blur free high dynamic range images from a single shot. While a prototype chip is currently in development, we demonstrate the benefits of coded rolling shutter via simulation using images of real scenes.

1. Introduction

CMOS image sensors are rapidly overtaking CCD sensors in a variety of imaging systems, from digital still and video cameras to mobile phone cameras to surveillance and web cameras. In order to maintain high fill-factor and readout speed, most CMOS image sensors are equipped with column-parallel readout circuits, which simultaneously read all pixels in a row into a line-memory. The readout proceeds row-by-row, sequentially from top to bottom. This is called rolling shutter. Rolling shutter has traditionally been considered detrimental to image quality, because pixels in different rows are exposed to light at different times, which often causes skew and other image artifacts, especially for moving objects [11, 13, 6].

From the perspective of sampling the space-time volume of a scene, however, we argue that the exposure discrepancy in rolling shutter can actually be exploited using computational photography to achieve new imaging functionalities and features. In fact, a few recent studies have demonstrated the use of conventional rolling shutter for kinematics and object pose estimation [1, 2, 3].

In this paper, we propose a novel readout architecture for CMOS image sensors called coded rolling shutter. We show that by controlling the readout timing and exposure length for each row of the pixel array, we can flexibly sample the 3D space-time volume of a scene and take photographs that effectively encode temporal scene appearance within a single 2D image. These coded images are useful for many applications, such as skew compensation, high-speed photography, and high dynamic range imaging.

As shown in Fig. 1, the controls of row-wise readout and exposure can be readily implemented in standard CMOS image sensors by altering the logic of the address generator unit without any further hardware modification. For conventional rolling shutter, the address generator is simply a shift register which scans all the rows and generates row-reset (RST) and row-select (SEL) signals. For coded rolling shutter, new logics can be implemented to generate the desired RST and SEL signals for coded readout and exposure, as shown in Fig. 2. Since the address generator belongs to the control unit of CMOS image sensors [9, 17], it is easy to design and implement new logics in the address generator using high level tools.

We have begun the process of developing the prototype sensor. We expect to have a fully programmable coded rolling shutter sensor in 18 months. Meanwhile, in this paper, we demonstrated coding schemes and their applications.

Figure 1. The address generator in CMOS image sensors is used to implement coded rolling shutter with desired row-reset and row-select patterns for flexible space-time sampling.

(a) CMOS image sensor architecture

(b) Timing for rolling shutter

Figure by Jinwei Gu
Rolling Shutter Distortion
Sensing Brightness

Pixel intensity $q(\lambda)_{Si}$ light (photons)

Quantum Efficiency

$$q(\lambda) = \frac{\text{generated electron flux}}{\text{photon flux of wavelength } \lambda}$$

Pixel intensity: $I = k(\text{generated electron flux})$

For monochromatic light $(\lambda = \lambda_i)$ with flux $P_i$:

$$I = kq(\lambda_i)P_i$$

However, incoming light can vary in wavelength $\lambda$
Sensing Brightness

Incoming light has a spectral distribution $p(\lambda)$

So the pixel intensity becomes

$$I = k \int_{-\infty}^{\infty} q(\lambda)p(\lambda) d\lambda$$
Sensing Color

 Assume we have an image
  - We know the pixel value $I$
  - We know our camera parameters $k, q(\lambda)$

Can we tell the color of the scene?
(Can we recover the spectral distribution $p(\lambda)$)

\[ I = k \int_{-\infty}^{\infty} q(\lambda)p(\lambda)d\lambda \]

Use a filter $f_i(\lambda)$ Where $f_i(\lambda) = \delta(\lambda - \lambda_i) = \begin{cases} 1 & \lambda = \lambda_i \\ 0 & \text{otherwise} \end{cases}$

then

\[ I = k \int_{-\infty}^{\infty} q(\lambda)p(\lambda)f(\lambda_i)d\lambda = kq(\lambda_i)p(\lambda_i) \]
How do we sense color?

Do we have infinite number of filters?

Three filters of different spectral responses
Sensing Color

- Tristimulus (trichromatic) values \((I_R, I_G, I_B)\)

Camera’s spectral response functions: \(h_R(\lambda), h_G(\lambda), h_B(\lambda)\)

\[
I_R = k \int_{-\infty}^{\infty} h_R(\lambda) p(\lambda) d\lambda
\]

\[
I_G = k \int_{-\infty}^{\infty} h_G(\lambda) p(\lambda) d\lambda
\]

\[
I_B = k \int_{-\infty}^{\infty} h_B(\lambda) p(\lambda) d\lambda
\]
Sensing Color

beam splitter

light

3 CCD

Bayer pattern

Foveon X3™
Today

How does a scene map to its image?
- Projective Geometry
- Homography

Assign Project 1
- Fun with homography
Camera Models and Projective Geometry

Reading: Zisserman and Mundy book appendix, Simoncelli notes
Modeling Projection

$f'$: effective focal length (will be $d$ from next slide)
Modeling Projection

- The coordinate system
  - We will use the pin-hole model as an approximation
  - Put the optical center (Center Of Projection) at the origin
  - Put the image plane (Projection Plane) in front of the COP
  - The camera looks down the negative z axis
Perspective Projection

- Compute intersection with PP of ray from \((x, y, z)\) to COP
- Derived using similar triangles
  \[ (x, y, z) \rightarrow (-d\frac{x}{z}, -d\frac{y}{z}, -d) \]
- We get the projection by throwing out the last coordinate:
  \[ (x, y, z) \rightarrow (-d\frac{x}{z}, -d\frac{y}{z}) \]
Homogeneous Coordinates

- Is this a linear transformation?
  - no—division by $z$ is nonlinear

- Trick: add one more coordinate:

  $\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$  \[ \Rightarrow \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \]

  homogeneous image
  coordinates

  homogeneous scene
  coordinates

- Converting \textit{from} homogeneous coordinates

  $\begin{bmatrix} x \\ y \\ w \end{bmatrix} \Rightarrow (x/w, y/w) \quad \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \Rightarrow (x/w, y/w, z/w)$
Perspective Projection

- Projection is a matrix multiplication using homogeneous coordinates:

\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & -1/d & 0 \\
0 & 0 & 1 & 0
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
z \\
1
\end{bmatrix}
= \begin{bmatrix}
x \\
y \\
z/d \\
1
\end{bmatrix}
\Rightarrow (-d\frac{x}{z}, -d\frac{y}{z})
\]

- This is known as *perspective projection*
  - The matrix is the **projection matrix**
  - Can also formulate as a 4x4 matrix

\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & -1/d & 0
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
z \\
1
\end{bmatrix}
= \begin{bmatrix}
x \\
y \\
z \\
-z/d
\end{bmatrix}
\Rightarrow (-d\frac{x}{z}, -d\frac{y}{z})
\]

divide by third coordinate

divide by fourth coordinate
Perspective Projection

- How does scaling the projection matrix change the transformation?

\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & -1/d & 0
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
z \\
1
\end{bmatrix}
=\begin{bmatrix}
x \\
y \\
-z/d \\
1
\end{bmatrix}
\Rightarrow (-d\frac{x}{z}, -d\frac{y}{z})
\]

\[
\begin{bmatrix}
-d & 0 & 0 & 0 \\
0 & -d & 0 & 0 \\
0 & 0 & 1 & 0
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
z \\
1
\end{bmatrix}
=\begin{bmatrix}
-dx \\
-dy \\
-z \\
1
\end{bmatrix}
\Rightarrow (-d\frac{x}{z}, -d\frac{y}{z})
\]
Orthographic Projection

- Special case of perspective projection
  - Distance from the COP to the PP (effective focal length) is infinite

- Also called “parallel projection”: \((x, y, z) \rightarrow (x, y)\)

- What’s the projection matrix?

\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
z \\
1
\end{bmatrix}
= \begin{bmatrix}
x \\
y \\
z \\
1
\end{bmatrix} \Rightarrow (x, y)
Weak-Perspective Projecton

- Scaled orthographic

\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1/d \\
0 & 0 & 0 & 1 \\
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
z \\
1 \\
\end{bmatrix}
= \begin{bmatrix}
x \\
y \\
z \\
1/d \\
\end{bmatrix} \Rightarrow (dx, dy)
\]
Affine Projection

- Also called “paraperspective”

\[
\begin{bmatrix}
  a & b & c & d \\
  e & f & g & h \\
  0 & 0 & 0 & 1 \\
  0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
  x \\
  y \\
  z \\
  1
\end{bmatrix}
\]
Camera Parameters
Camera Parameters

A camera is described by several parameters

- Translation $T$ of the optical center from the origin of world coords
- Rotation $R$ of the image plane
- focal length $f$, principle point $(x'_c, y'_c)$, pixel size $(s_x, s_y)$
- blue parameters are called “extrinsics,” red are “intrinsics”

**Projection equation**

$$
\mathbf{x} = \begin{bmatrix} s_x \\ s_y \\ s \end{bmatrix} = \begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \Pi \mathbf{X}
$$

- The projection matrix models the cumulative effect of all parameters
- Useful to decompose into a series of operations

$$
\Pi = \begin{bmatrix} -f s_x & 0 & x'_c \\ 0 & -f s_y & y'_c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R_{3x3} & 0_{3x1} \\ 0_{1x3} & I_{3x3} \end{bmatrix} \begin{bmatrix} T_{3x1} \end{bmatrix}
$$

- identity matrix
- intrinsics
- projection
- rotation
- translation
Camera Calibration

- Goal: estimate the camera parameters
  - Version 1: solve for projection matrix
    \[
    \begin{bmatrix}
    wx \\
    wy \\
    w
    \end{bmatrix}
    =
    \begin{bmatrix}
    * & * & * & * \\
    * & * & * & * \\
    * & * & * & * \\
    \end{bmatrix}
    \begin{bmatrix}
    X \\
    Y \\
    Z \\
    1
    \end{bmatrix}
    = \Pi X
    \]
  - Version 2: solve for camera parameters separately
    - intrinsics (focal length, principle point, pixel size)
    - extrinsics (rotation angles, translation)
    - radial distortion
Estimating the Projection Matrix

- Place a known object in the scene
  - identify correspondence between image and scene
  - compute mapping from scene to image

\[
\begin{bmatrix}
  u_i \\
  v_i \\
  1
\end{bmatrix}
= \begin{bmatrix}
  m_{00} & m_{01} & m_{02} & m_{03} \\
  m_{10} & m_{11} & m_{12} & m_{13} \\
  m_{20} & m_{21} & m_{22} & m_{23}
\end{bmatrix}
\begin{bmatrix}
  X_i \\
  Y_i \\
  Z_i \\
  1
\end{bmatrix}
\]
Direct Linear Calibration

\[
\begin{bmatrix}
u_i \\ v_i \\ 1
\end{bmatrix} \mapsto \begin{bmatrix}
m_{00} & m_{01} & m_{02} & m_{03} \\ m_{10} & m_{11} & m_{12} & m_{13} \\ m_{20} & m_{21} & m_{22} & m_{23}
\end{bmatrix} \begin{bmatrix}
X_i \\ Y_i \\ Z_i \\ 1
\end{bmatrix}
\]

\[
u_i = \frac{m_{00}X_i + m_{01}Y_i + m_{02}Z_i + m_{03}}{m_{20}X_i + m_{21}Y_i + m_{22}Z_i + m_{23}}
\]

\[
v_i = \frac{m_{10}X_i + m_{11}Y_i + m_{12}Z_i + m_{13}}{m_{20}X_i + m_{21}Y_i + m_{22}Z_i + m_{23}}
\]

\[
u_i(m_{20}X_i + m_{21}Y_i + m_{22}Z_i + m_{23}) = m_{00}X_i + m_{01}Y_i + m_{02}Z_i + m_{03}
\]

\[
v_i(m_{20}X_i + m_{21}Y_i + m_{22}Z_i + m_{23}) = m_{10}X_i + m_{11}Y_i + m_{12}Z_i + m_{13}
\]

\[
\begin{bmatrix}
X_i & Y_i & Z_i & 1 & 0 & 0 & 0 & 0 & -u_iX_i & -u_iY_i & -u_iZ_i & -u_i \\
0 & 0 & 0 & 0 & X_i & Y_i & Z_i & 1 & -v_iX_i & -v_iY_i & -v_iZ_i & -v_i
\end{bmatrix}
\begin{bmatrix}
m_{00} \\ m_{01} \\ m_{02} \\ m_{03} \\ m_{10} \\ m_{11} \\ m_{12} \\ m_{13} \\ m_{20} \\ m_{21} \\ m_{22} \\ m_{23}
\end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}
\]
Direct Linear Calibration

\[
\begin{bmatrix}
X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & 0 & -u_1X_1 & -u_1Y_1 & -u_1Z_1 & -u_1 \\
0 & 0 & 0 & 0 & X_1 & Y_1 & Z_1 & 1 & -v_1X_1 & -v_1Y_1 & -v_1Z_1 & -v_1 \\
& \vdots \\
X_n & Y_n & Z_n & 1 & 0 & 0 & 0 & 0 & -u_nX_n & -u_nY_n & -u_nZ_n & -u_n \\
0 & 0 & 0 & 0 & X_n & Y_n & Z_n & 1 & -v_nX_n & -v_nY_n & -v_nZ_n & -v_n
\end{bmatrix}
\begin{bmatrix}
m_{00} \\
m_{01} \\
m_{02} \\
m_{03} \\
m_{10} \\
m_{11} \\
m_{12} \\
m_{13} \\
m_{20} \\
m_{21} \\
m_{22}
\end{bmatrix} =
\begin{bmatrix}
0 \\
0 \\
0 \\
0 \\
0
\end{bmatrix}
\]

\[A_{2n \times 12}\]

\[m_{12} \ 0_{2n}\]

- Total least squares
  - Since \( \mathbf{m} \) is only defined up to scale, solve for unit vector \( \hat{\mathbf{m}} \)
  - Minimize \( \|A\hat{\mathbf{m}}\|^2 \)
    \[\|A\hat{\mathbf{m}}\|^2 = (A\hat{\mathbf{m}})^T A\hat{\mathbf{m}} = \hat{\mathbf{m}}^T A^T A \hat{\mathbf{m}}\]
  - Solution: \( \hat{\mathbf{m}} = \) eigenvector of \( A^T A \) with smallest eigenvalue
  - Works with 6 or more points
Direct Linear Calibration

- Advantage:
  - Very simple to formulate and solve

- Disadvantages:
  - Doesn’t tell you the camera parameters
  - Doesn’t model radial distortion
  - Hard to impose constraints (e.g., known focal length)
  - Doesn’t minimize the right error function

For these reasons, *nonlinear methods* are preferred

- Define error function $E$ between projected 3D points and image positions
  - $E$ is nonlinear function of intrinsics, extrinsics, radial distortion
- Minimize $E$ using nonlinear optimization techniques
  - e.g., variants of Newton’s method (e.g., Levenberg-Marquart)
Alternative: Multi-Plane Calibration

Advantage

- Only requires a plane
- Don’t have to know positions/orientations
- Good code available online!
  - Intel’s OpenCV library: http://www.intel.com/research/mrl/research/opencv/
  - Zhengyou Zhang’s web site: http://research.microsoft.com/~zhang/Calib/

Images courtesy Jean-Yves Bouguet, Intel Corp.