Outline

• Introduction
• System Overview
• Camera Calibration
• Marker Tracking
• Pose Estimation of Markers
• Conclusion
Outline

• Introduction
• System Overview
• Camera Calibration
• Marker Tracking
• Pose Estimation of Markers
• Conclusion
R. T. Azuma[1] define an Augmented Reality (AR) system to have the following properties:
1) Combines real and virtual
2) Interactive in real time
3) Registered in 3-D

Marker-based AR

- Basic workflow of an AR application using *fiducial marker* tracking:
Markerless AR

- Markerless augmented reality systems rely on **natural features** instead of fiducial marks.
Features

• Also known as interesting points, salient points or keypoints.

• Points that you can easily point out their correspondences in multiple images using only local information.
Desired Properties for Features

- **Distinctive**: a single feature can be correctly matched with high probability.
- **Invariant**: invariant to scale, rotation, affine, illumination and noise for robust matching across a substantial range of affine distortion, viewpoint change and so on. That is, it is **repeatable**.
Components

• Feature detection locates where they are.
• Feature description describes what they are.
• Feature matching decides whether two are the same one.
Detectors & Descriptors

• Feature detector:
  – Harris, Hessian, MSER, **SIFT, SURF**, FAST, etc.

• Feature descriptor:
  – **SIFT, SURF**, DAISY, BRIEF, FREAK, etc.
Another Markerless AR

- Complete registration with GPS, inertial sensors, and magnetic sensors.
Outline

- Introduction
- System Overview
- Camera Calibration
- Marker Tracking
- Pose Estimation of Markers
- Conclusion
System Flow

Camera Calibration → Marker Tracking → Pose Estimation → Rendering

Camera Coordinates $(X_c, Y_c, Z_c)$

Marker Coordinates $(X_m, Y_m, Z_m)$

Camera Screen Coordinates $(x_c, y_c)$
Outline

- Introduction
- System Overview
- Camera Calibration
- Marker Tracking
- Pose Estimation of Markers
- Conclusion
We use a simple cardboard frame with a ruled grid of lines for the camera calibration. Relationships between the camera screen coordinates and the camera coordinates can be known.
The relationships among the camera screen coordinates \((x_c, y_c)\), the camera coordinates \((X_c, Y_c, Z_c)\) and the marker coordinates \((X_m, Y_m, Z_m)\) can be represented as below:

\[
\begin{bmatrix}
hx_c \\
hy_c \\
h \\
h \\
1
\end{bmatrix} = P \begin{bmatrix}
x_c \\
y_c \\
1
\end{bmatrix} = P \cdot T_{cm} \begin{bmatrix}
X_m \\
Y_m \\
Z_m \\
1
\end{bmatrix} = C \begin{bmatrix}
X_m \\
Y_m \\
Z_m \\
1
\end{bmatrix} = \begin{bmatrix}
C_{11} & C_{12} & C_{13} & C_{14} & X_m \\
C_{21} & C_{22} & C_{23} & C_{24} & Y_m \\
C_{31} & C_{32} & C_{33} & 1 & Z_m \\
0 & 0 & 0 & 1 & 0
\end{bmatrix}
\]

\[
P = \begin{bmatrix}
s_x f & 0 & x_0 & 0 \\
0 & s_y f & y_0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix},\quad T_{cm} = \begin{bmatrix}
R_{11} & R_{12} & R_{13} & T_x \\
R_{21} & R_{22} & R_{23} & T_y \\
R_{31} & R_{32} & R_{33} & T_z \\
0 & 0 & 0 & 1
\end{bmatrix}
\]
The relationships between the camera screen coordinates \((x_c, y_c)\) and the camera coordinates \((X_c, Y_c, Z_c)\) can be represented as below:

\[
\begin{bmatrix}
  h x_c \\
  h y_c \\
  h \\
  1
\end{bmatrix} = \mathbf{P} \begin{bmatrix}
  X_c \\
  Y_c \\
  Z_c \\
  1
\end{bmatrix} = \begin{bmatrix}
  s_x f & 0 & x_0 & 0 \\
  0 & s_y f & y_0 & 0 \\
  0 & 0 & 1 & 0 \\
  0 & 0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
  X_c \\
  Y_c \\
  Z_c \\
  1
\end{bmatrix}
\]

\[
x_c = s_x f \frac{X_c}{Z_c} + x_0
\]
\[
y_c = s_y f \frac{Y_c}{Z_c} + y_0
\]
The relationships between the camera coordinates \((X_c, Y_c, Z_c)\) and the marker coordinates \((X_m, Y_m, Z_m)\) can be represented as below:

\[
\begin{bmatrix}
X_c \\
Y_c \\
Z_c
\end{bmatrix} = \mathbf{T}_{cm} \begin{bmatrix}
X_m \\
Y_m \\
Z_m \\
1
\end{bmatrix} = \begin{bmatrix}
R_{11} & R_{12} & R_{13} & T_x \\
R_{21} & R_{22} & R_{23} & T_y \\
R_{31} & R_{32} & R_{33} & T_z \\
0 & 0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
X_m \\
Y_m \\
Z_m \\
1
\end{bmatrix} + \begin{bmatrix}
T_x \\
T_y \\
T_z
\end{bmatrix}
\]
So the relationships between the camera screen coordinates \((x_c, y_c)\) and the marker coordinates \((X_m, Y_m, Z_m)\) can be represented as below:

\[
\begin{bmatrix}
hx_c \\
hy_c \\
h \\
1
\end{bmatrix}
= P
\begin{bmatrix}
x_c \\
y_c \\
Z_c \\
1
\end{bmatrix}
= P \cdot T_{cm}
= C
\begin{bmatrix}
X_m \\
Y_m \\
Z_m \\
1
\end{bmatrix}
\]

\[
P =
\begin{bmatrix}
s_x f & 0 & x_0 & 0 \\
0 & s_y f & y_0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix},
T_{cm} =
\begin{bmatrix}
R_{11} & R_{12} & R_{13} & T_x \\
R_{21} & R_{22} & R_{23} & T_y \\
R_{31} & R_{32} & R_{33} & T_z \\
0 & 0 & 0 & 1
\end{bmatrix}
\]
A scalar variable \( k \) is added into \( P \) because matrix \( C \) has 11 independent variables but matrices \( P \) and \( T_{cm} \) have 4 and 6 respectively.

Since many pairs of \((x_c, y_c)\) and \((X_m, Y_m, Z_m)\) have been obtained by the procedure mentioned above, matrix \( C \) can be estimated.

\[
\begin{bmatrix}
  h x_c \\
  h y_c \\
  h \\
  1
\end{bmatrix}
= P 
\begin{bmatrix}
  X_c \\
  Y_c \\
  Z_c \\
  1
\end{bmatrix}
= P \cdot T_{cm}
= \begin{bmatrix}
  X_m \\
  Y_m \\
  Z_m \\
  1
\end{bmatrix}
= C
\begin{bmatrix}
  X_m \\
  Y_m \\
  Z_m \\
  1
\end{bmatrix}
\]

\[
P = \begin{bmatrix}
  s_x f & k & x_0 & 0 \\
  0 & s_y f & y_0 & 0 \\
  0 & 0 & 1 & 0 \\
  0 & 0 & 0 & 1
\end{bmatrix}, \quad T_{cm} = \begin{bmatrix}
  R_{11} & R_{12} & R_{13} & T_x \\
  R_{21} & R_{22} & R_{23} & T_y \\
  R_{31} & R_{32} & R_{33} & T_z \\
  0 & 0 & 0 & 1
\end{bmatrix}
\]

11

5

6
Outline

• Introduction
• System Overview
• Camera Calibration
• Marker Tracking
• Pose Estimation of Markers
• Conclusion
The tracking pipeline consists of four basic steps:\textsuperscript{[2]}:

1) Thresholding
2) Fiducial Marker detection
3) Rectangle fitting
4) Pattern checking

Thresholding

• The first step in the image processing pipeline is to convert the input image into a binary image to reliably detect the black and white portions of the fiducial markers.

• A heuristic that has proven effective is to use the median of all marker pixels extracted in the last frame as a threshold for current image.

• Vignetting compensation incorporates the radial brightness into the per-pixel threshold value.

![Vignetting. Left: original camera image. Middle: constant thresholding. Right: thresholding with vignetting compensation.](image_url)
Fiducial Marker Detection

- As a first step all scan-lines are searched left to right for edges. A sequence of white followed by black pixels is considered a candidate for a marker’s border.
- The software then follows this edge until either a loop back to the start pixel is closed or until the border of the image is reached.
- All pixels that have been visited are marked as processed in order to prevent following edges more than once.

Left: Source image; Middle: Threshold image; Right: Three closed polygons as candidates for rectangle fitting.
Rectangle Fitting

• A first corner point $c_0$ is selected as the contour point that lies at the maximum distance to an arbitrary starting point $x$ of the contour.
• The center of the rectangle is estimated as the center of mass of all contour points. A line is formed from the first corner point and the center.
• Further corner points $c_1, c_2$ are found on each side of the line by searching for those points that have the largest distance to this line. These three corners $c_0, c_1, c_2$ are used to construct more lines and recursively search for additional corners.
• An iterative process searches for corners until the whole contour has been searched or more than four corner points are detected.

Example for fitting a rectangle to a polygon
Pattern Checking

- First a marker’s interior region is resampled into a normalized arrays of pixels.
- For perspectively correct unprojection, the homography matrix is computed from the markers’ corner points, which are assumed to form a rectangle.
- ARToolKit uses simple L2-norm template matching.
- There are several types of markers, and some are designed for efficiency.

Marker types. Left: Template markers; Middle: BCHmarkers; Right: DataMatrix markers
Outline

- Introduction
- System Overview
- Camera Calibration
- Marker Tracking
- Pose Estimation of Markers
- Conclusion
Introduction to PnP Problem

- The aim of the **Perspective-n-Point (PnP)** problem is to determine the **position and orientation** of a camera given its:
  1) intrinsic parameters
  2) a set of n correspondences between **3D points** and their **2D projections**.

\[
\begin{bmatrix}
  h x_c \\
  h y_c \\
  h 
\end{bmatrix} =
\begin{bmatrix}
  s_x f & 0 & x_0 & 0 \\
  0 & s_y f & y_0 & 0 \\
  0 & 0 & 1 & 0 \\
  0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
  R_{11} & R_{12} & R_{13} & T_x \\
  R_{21} & R_{22} & R_{23} & T_y \\
  R_{31} & R_{32} & R_{33} & T_z \\
  0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
  X_m \\
  Y_m \\
  Z_m \\
  1
\end{bmatrix}
\]
Papers for Solving PnP Problem

- "Linear Pose Estimation from Points or Lines," TPAMI 2003.
- "Robust Pose Estimation from a Planar Target," TPAMI 2006.
- "Global Optimization through Searching Rotation Space and Optimal Estimation of the Essential Matrix," 2007 ICCV.
- "A Robust O(n) Solution to the Perspective-n-Point Problem," TPAMI 2012.
Position Estimation of Markers (1/5)

- All variables in the transformation matrix are determined by substituting screen coordinates and marker coordinates of detected marker's four vertices for \((x_c, y_c)\) and \((X_m, Y_m)\) respectively\(^3\).

\[
\begin{bmatrix}
  hx_c \\
hy_c \\
h
\end{bmatrix} =
\begin{bmatrix}
  N_{11} & N_{12} & N_{13} \\
  N_{21} & N_{22} & N_{23} \\
  N_{31} & N_{32} & 1
\end{bmatrix}
\begin{bmatrix}
  X_m \\
  Y_m \\
  1
\end{bmatrix}
\]

- When two parallel sides of a square marker are projected on the image, the equations of those line segments in the camera screen coordinates are the following

\[
\begin{align*}
a_1 x_c + b_1 y_c + c_1 &= 0 \\
a_2 x_c + b_2 y_c + c_2 &= 0
\end{align*}
\]

Position Estimation of Markers (2/5)

- Given the intrinsic matrix $\mathbf{P}$ that is obtained by the camera calibration, equations of the planes that include these two sides respectively can be represented in the camera coordinates.

$$\begin{bmatrix} h x_c \\ h y_c \\ h \\ 1 \end{bmatrix} = \mathbf{P} \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = \begin{bmatrix} P_{11} & P_{12} & P_{13} & 0 \\ 0 & P_{22} & P_{23} & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix}$$

$$a_1 x_c + b_1 y_c + c_1 = 0$$
$$a_2 x_c + b_2 y_c + c_2 = 0$$

$$a_1 P_{11} X_c + (a_1 P_{12} + b_1 P_{22}) Y_c + (a_1 P_{13} + b_1 P_{23} + c_1) Z_c = 0$$
$$a_2 P_{11} X_c + (a_2 P_{12} + b_2 P_{22}) Y_c + (a_2 P_{13} + b_2 P_{23} + c_2) Z_c = 0$$
• Given that normal vectors of these planes are \( \mathbf{n}_1 \) and \( \mathbf{n}_2 \) respectively, the direction vector of parallel two sides of the square is given by the outer product \( \mathbf{u}_1 = \mathbf{n}_1 \times \mathbf{n}_2 \).

• Given that two unit direction vectors that are obtained from two sets of two parallel sides of the square is \( \mathbf{u}_1 \) and \( \mathbf{u}_2 \), these vectors should be perpendicular.
• However, image processing errors mean that the vectors won't be exactly perpendicular. To compensate for this two perpendicular unit, direction vectors are defined by $\mathbf{r}_1$ and $\mathbf{r}_2$ in the plane that includes $\mathbf{u}_1$ and $\mathbf{u}_2$.

![Diagram showing vectors $\mathbf{r}_1$, $\mathbf{r}_2$, $\mathbf{u}_1$, and $\mathbf{u}_2$.]

\[
\begin{bmatrix}
X_c \\
Y_c \\
Z_c \\
1
\end{bmatrix} = T_{cm} \begin{bmatrix}
X_m \\
Y_m \\
Z_m \\
1
\end{bmatrix} = \begin{bmatrix}
R_{11} & R_{12} & R_{13} & T_x \\
R_{21} & R_{22} & R_{23} & T_y \\
R_{31} & R_{32} & R_{33} & T_z \\
0 & 0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
X_m \\
Y_m \\
Z_m \\
1
\end{bmatrix}
\]

• Given that the unit direction vector which is perpendicular to both $\mathbf{r}_1$ and $\mathbf{r}_2$ is $\mathbf{r}_3$, the rotation component $R_{3\times3}$ in the transformation matrix $T_{cm}$ from marker coordinates to camera coordinates is $[\mathbf{r}_1 \ \mathbf{r}_2 \ \mathbf{r}_3]$. 
• The four vertices coordinates of the marker in the marker coordinate frame and those coordinates in the camera screen coordinate frame, eight equations including translation component $T_x\ T_y\ T_z$ are generated.

\[
\begin{bmatrix}
hx_c \\
hy_c \\
h \\
1
\end{bmatrix}
= 
\begin{bmatrix}
s_x f & 0 & x_0 & 0 \\
0 & s_y f & y_0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
R_{11} & R_{12} & R_{13} & T_x \\
R_{21} & R_{22} & R_{23} & T_y \\
R_{31} & R_{32} & R_{33} & T_z \\
0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
X_m \\
Y_m \\
Z_m \\
1
\end{bmatrix}
\]

• The vertex coordinates of the markers in the marker coordinate frame can be transformed to coordinates in the camera screen coordinate frame by using the transformation matrix obtained.
Outline

• Introduction
• System Overview
• Camera Calibration
• Marker Tracking
• Pose Estimation of Markers
• Conclusion
Conclusion

• We propose a method for tracking fiducial markers and a calibration method for camera based on the marker tracking.
  – The most important thing is to get the intrinsic and extrinsic matrix.

• The rendering part can be implemented with the transformation matrix obtained.
  – Topics of GPU group.