Projective texture mapping

Overview

- Approximate calibration of intrinsic parameters
- Calibration of extrinsic parameters for each reference photograph
- Rendering

Approximate calibration of intrinsics

- Print out black and white grid
  - As large as possible, at least 40 cm x 40 cm
  - OK to use several letter-sized pages; measure the global consistency of the grid
- Aim camera perpendicularly to the grid
  - Entire field of view covered by grid
  - Frame margins as parallel as possible to the grid lines
  - Measure distance from grid to camera f_{cm} in cm
  - In image, measure width w_{cm} and height h_{cm} of patch of grid in cm (using known size of checkers)
- Approximate intrinsics
  - Set pixel width to 1; set pixel height to $(h_{cm}/h) / (w_{cm}/w)$, where w and h are the image dimensions in pixels
  - Assume square pixels, and that the COP projects in the center of the image
  - a=(1, 0, 0), b=(0, $-pix_h$, 0), c=($-w$/2, $-h$/2*pix_h, $-f_{cm}/(w_{cm}/w)$), C=(0, 0, 0)

Extrinsic calibration

- Establish correspondences
  - (u, v) in image (triangle vertex projections)
  - (x, y, z) in model (triangle vertices)
- Implement error function
  - ExCalError(PHC_0, t_x, t_y, t_z, r_x, r_y, r_z)
  - PHC_0 is approximate guess established manually
  - t's are translations, r's are rotations; use your camera navigation functions
  - PHC = PositionAndAim(PHC_0, t_x, t_y, t_z, r_x, r_y, r_z)
  - For each correspondence
    - (u^*, v^*) = Project(PHC, x, y, z)
    - error += (u*-u) (u*-u) + (v*-v) (v*-v)
  - return error /= correspondencesN
- Search using downhill simplex in 6 dimensions, from initial guess (0, 0, 0, 0, 0, 0), using ExCalError()
- When search converges, scene rendered from found *PHC* matches image

Rendering

- For each desired view D
- For each triangle T
- Project vertices of T
- For each pixel p inside T
- For each reference image R
- If point P of triangle T seen at p is visible in R and R provides better sample than current sample
- Set p to p_R, where p_R is the color where p projects in R (use bilinear interpolation)

Mapping from desired to reference image

(C_2, a_2, b_2, c_2) – reference view
(u_2, v_2) – reference pixel coordinates
(C_1, a_1, b_1, c_1) – desired view
(u_1, v_1) – desired view pixel coordinates
(u_2, v_2, w_2) – unknowns
w_1 – computed using triangle $V_0V_1V_2$

$$\dot{C}_1 + (\bar{c}_1 + u_1\bar{a}_1 + v_1\bar{b}_1)w_1 = \dot{C}_2 + (\bar{c}_2 + u_2\bar{a}_2 + v_2\bar{b}_2)w_2$$

$$\begin{bmatrix}\bar{a}_2 & \bar{b}_2 & \bar{c}_2\end{bmatrix}\begin{bmatrix}w_2u_2\\ w_2v_2\\ w_2\end{bmatrix} = \dot{C}_1 - \dot{C}_2 + \begin{bmatrix}\bar{a}_1 & \bar{b}_1 & \bar{c}_1\end{bmatrix}\begin{bmatrix}w_1u_1\\ w_1v_1\\ w_1\end{bmatrix}$$

$$\begin{bmatrix}w_2u_2\\ w_2v_2\\ w_2\end{bmatrix} = \begin{bmatrix}\bar{a}_2 & \bar{b}_2 & \bar{c}_2\end{bmatrix}^{-1}(\dot{C}_1 - \dot{C}_2) + \begin{bmatrix}\bar{a}_2 & \bar{b}_2 & \bar{c}_2\end{bmatrix}^{-1}\begin{bmatrix}\bar{a}_1 & \bar{b}_1 & \bar{c}_1\end{bmatrix}\begin{bmatrix}w_1u_1\\ w_1v_1\\ w_1\end{bmatrix}$$

$$\begin{bmatrix}w_2u_2\\ w_2v_2\\ w_2\end{bmatrix} = \begin{bmatrix}q_{00}\\ q_{10}\\ q_{20}\end{bmatrix} + \begin{bmatrix}q_{01} & q_{02} & q_{03}\\ q_{11} & q_{12} & q_{13}\\ q_{21} & q_{22} & q_{23}\end{bmatrix}\begin{bmatrix}w_1u_1\\ w_1v_1\\ w_1\end{bmatrix}$$

$$u_2 = \frac{q_{00}\frac{1}{w_1} + q_{01}u_1 + q_{02}v_1 + q_{03}}{q_{20}\frac{1}{w_1} + q_{21}u_1 + q_{22}v_1 + q_{23}}, v_2 = \frac{q_{10}\frac{1}{w_1} + q_{11}u_1 + q_{12}v_1 + q_{13}}{q_{20}\frac{1}{w_1} + q_{21}u_1 + q_{22}v_1 + q_{23}}$$

$$w_1 = \frac{1}{k_0 + k_1u_1 + k_2v_1}, u_2 = \frac{h_{00} + h_{01}u_1 + h_{02}v_1}{h_{20} + h_{21}u_1 + h_{22}v_1}, v_2 = \frac{h_{10} + h_{11}u_1 + h_{12}v_1}{h_{20} + h_{21}u_1 + h_{22}v_1}$$