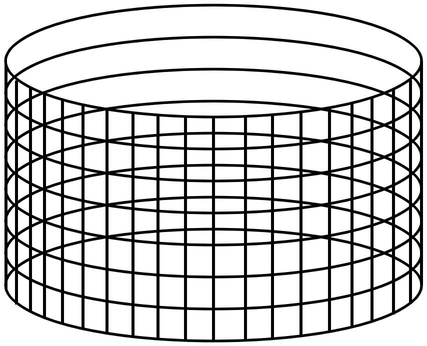
Panoramas

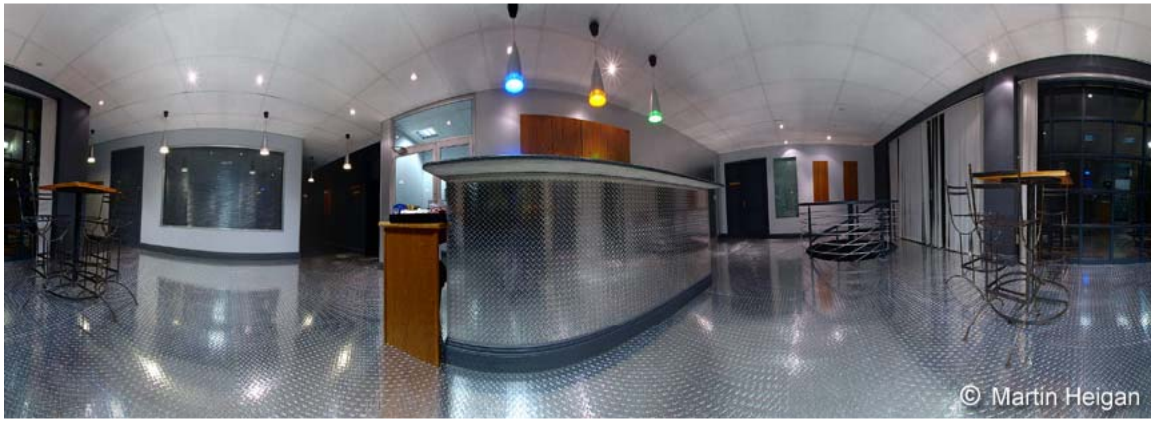
Capturing and rendering real world scenes using panoramas

- Panorama
  - An image acquired with a camera with a large field of view
- Versatile method
  - Any scene, anywhere (near or far, indoor or outdoor, shiny or diffuse surfaces, simple or complex geometry)
  - Inexpensive acquisition and rendering hardware
- Limitation
  - Only allows viewing the scene from a single viewpoint, i.e. the center of the panorama

Panoramic camera models

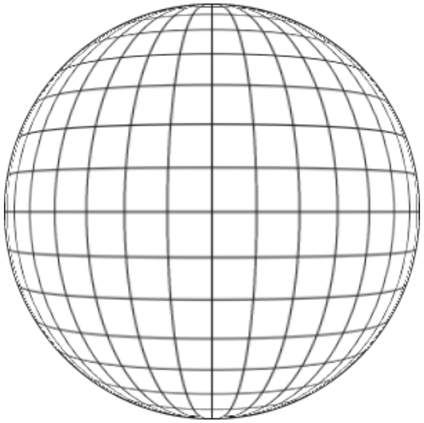
- Cylindrical
  - Same distance between pixels in columns
  - Same angle between pixels in row
  - A one-column PPC panned equal angles
  - No sampling of north and south pole




© Martin Heigan

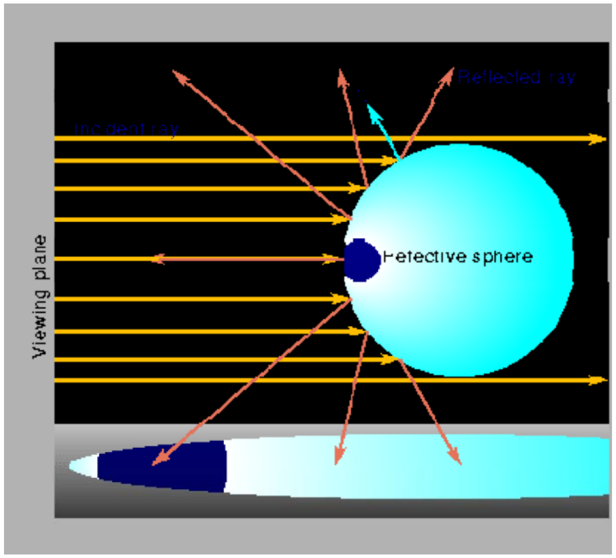
Panoramic camera models

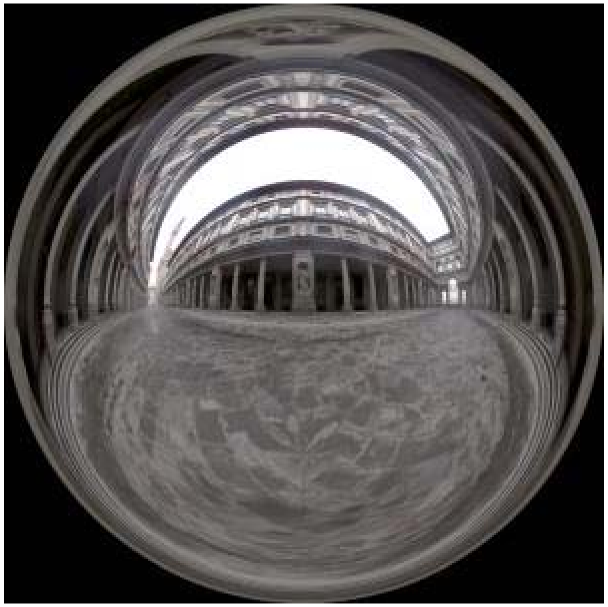
- Spherical 1
  - Equal angle between 2 consecutive pixels in same row or column
  - Like latitude / longitude system on Earth globe
  - Oversampling at poles



Panoramic camera models

- Spherical 2
  - Orthographic sampling of reflective sphere
  - Poor sampling at sphere periphery

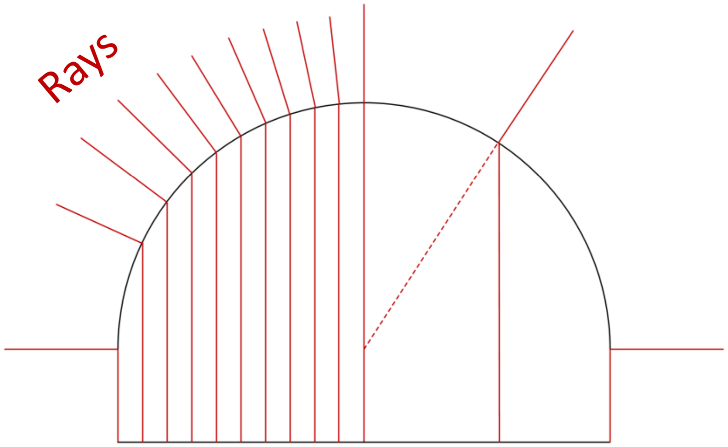




© Debevec

Panoramic camera models

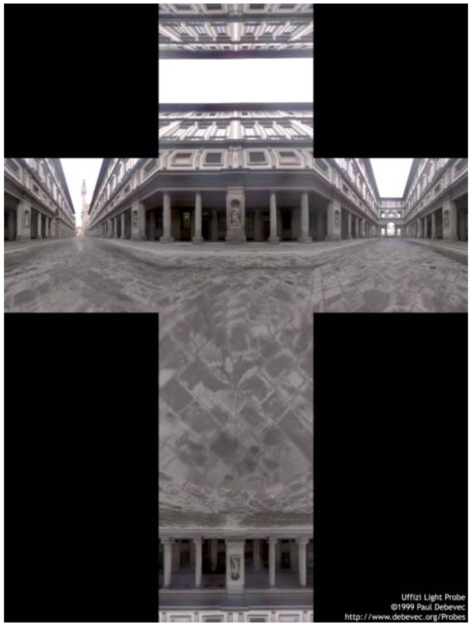
- Fisheye
  - Hemisphere projected onto image plane
  - Coarse sampling at periphery
  - Max 180º x 180º FOV



© Wikimedia

Panoramic camera models

- Cube map
  - 6 PPCs with 90^o x 90^o FOV
  - Good sampling rate control
  - However: 6 images



© Debevec

Acquisition

- Single shot with panoramic camera
  - Fisheye lens
  - Multiple PPCs
  - Catadioptric imaging systems (mirrors + lenses)
- Multiple shots
  - Conventional PPC handheld or on tripod
  - Single column camera on tripod

Panorama construction

- Input
  - a stack of overlapping same viewpoint PPC images
- Output
  - Cube map panorama
- Steps
  - Register input images
  - Construct cube map from registered images

Image registration

- Register pairs of consecutive images
  - Register image I_{i+1} to I_i
  - In other words modify PPC_{i+1} to align the 2 images
- There are 3 degrees of freedom
    - Pan, tilt, roll
- Initial guess of (0, 0, 0)
- Search for rotation angles that minimize registration error
  - Error is defined as average color difference at region of overlap

Cube map construction

- For every face F_i
- For every pixel p_i in F_i
- For each input image I_i
- Project p_i on I_i at pp_i using registered camera PPC_i
- If projection is valid, lookup I_i at pp_i to get color c_i
- Blend all colors c_i to get final color for pixel p_i

Rendering from panorama

- Input: cube map C, desired view ppc
- Output: desired image fb
- Algorithm
  - For every pixel p in fb
  - For every face F in C
  - Project p onto F using F's planar pinhole camera
    - Note: you need to project the 3-D point corresponding to the pixel center, which is computed by unprojection using ppc
  - If valid projection, set p to face color at projection location