Mesh Lens Distortion Correction (Mesh LDC)

Geometric Transform Engine

Arbitrary Image Warping with 2D Mesh LUT

8-bit and/or 12-bit YUV I/O Images

Second output image with 8/12-bit conversion

Error Concealment and Report

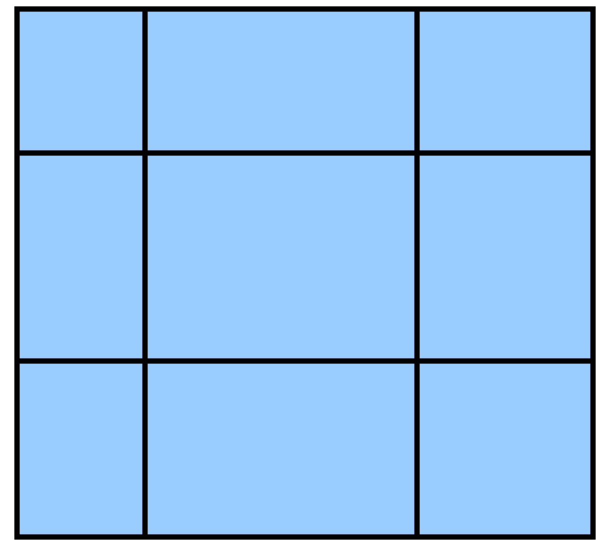
Mesh LDC Features

- Image formats
  - 8-bit YUV422 (UYVY)
  - 8-bit or 12-bit YUV420, or Y-only, or UV-only
- Image size
  - Up to 8192 x 8192 (for both input and output)
- Image warping
  - 2D projective transform: 3x4 homogeneous transform (16-bit int)
  - Arbitrary geometric transform: 2D mesh LUT with down-sampling
- Pixel interpolation
  - Spatial resolution: 1/8 pixel
  - Bi-linear Y and bi-linear Cb/Cr: 1 cycle/pixel
  - Bi-cubic Y and bi-linear Cb/Cr: 2 cycles/pixel
  - Anti-aliasing filter before interpolation: Not available
- Output format conversion
  - 422 to 420 conversion
  - From 8- or 12-bit input pixel to 8- or 12-bit output pixels
    - Up to two output images for one input image
    - Simultaneous YUV-12b and YUV-8b output

Mesh LDC Features

- Block based image processing
  - Autonomous memory-to-memory operation

- Up to 3x3 spatial processing regions for an output image
  - Programmable block parameters for each region
  - Each output region may be skipped
  - Reduce input image memory bandwidth overhead

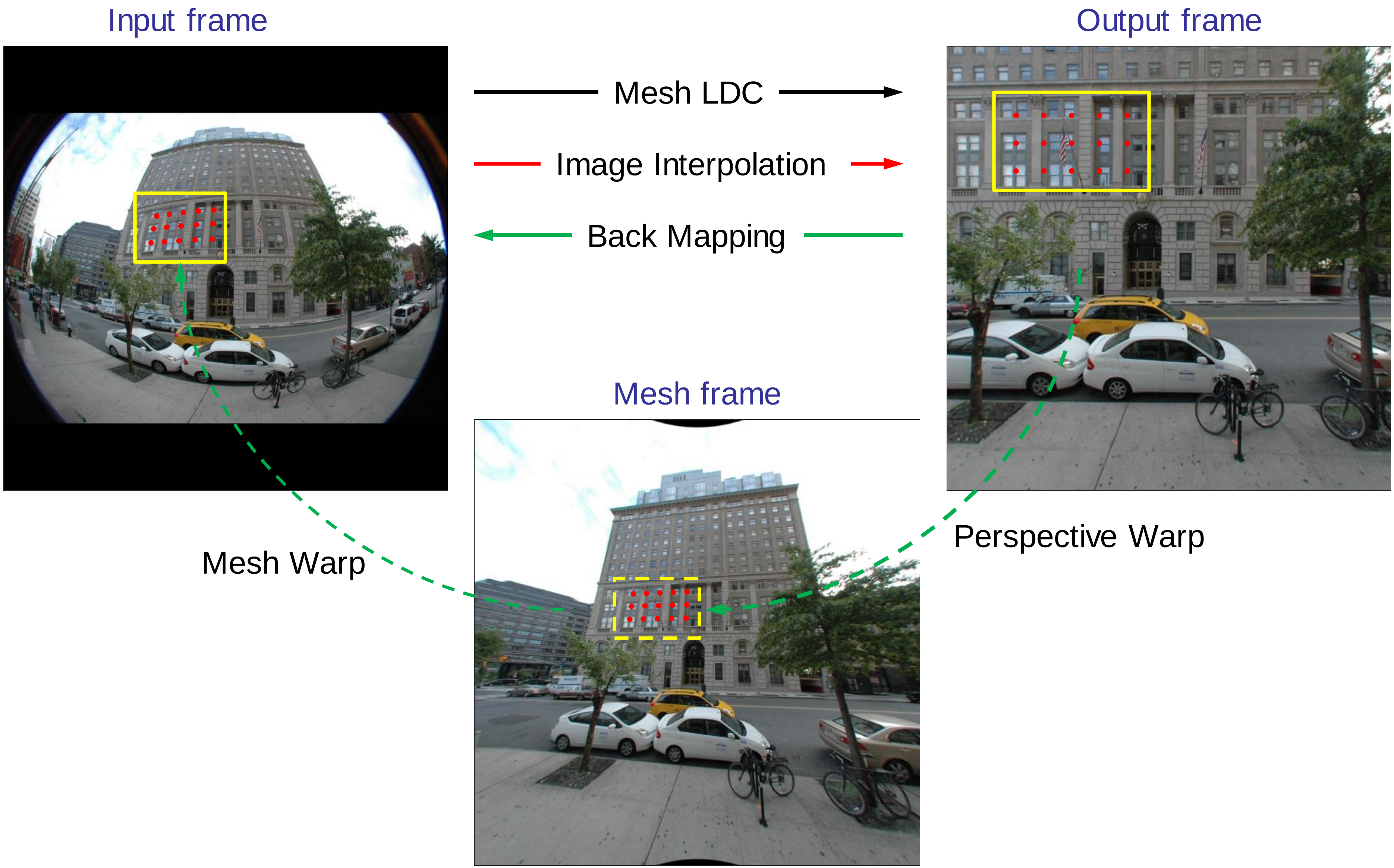


- Error concealment and error report
  - Out of image frame boundary or mesh frame boundary
  - Missing mesh entries in block interpolation
  - Missing pixel values in block interpolation
  - Projective transform overflow/underflow
  - ECC support on Mesh Data internal storage

Lens Distortion Correction Examples

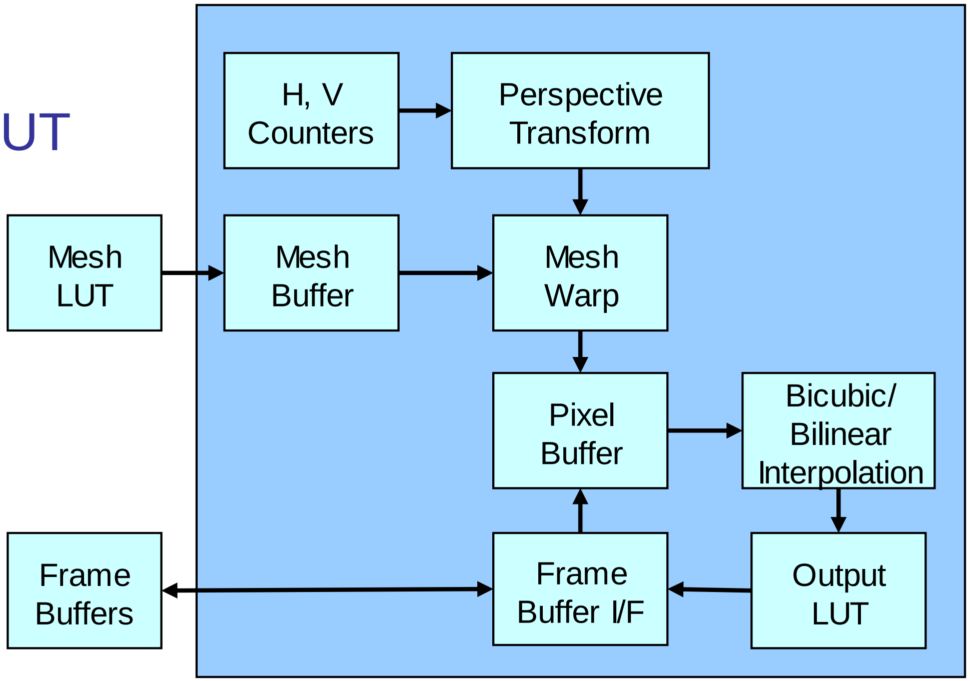
- Original Nikon D70 picture from Wikipedia under CC BY-SA 3.0 license
  - The original photo was taken by participant/team ***The Squirrels***
  - The original photo is modified and transformed for illustrations on this page and the other 7 pages in this LDC section
- Perspective views created by LDC
  - 1280x1280 LDC input image (resized and border adjusted from the original photo)
  - 180-degree equisolid fisheye projection model approximates lens distortion fairly well
  - 90-degree (H and V) and 120-degree rectilinear views

Mesh LDC Operations



Mesh LDC: Functional Overview

- H/V counters: undistorted coordinates
  - Processing in small 2D pixel blocks to optimize memory throughput
- Perspective transform on undistorted coordinates
- Mesh warp: undistorted coordinates → distorted coordinates
  - Flexible mesh LUT with down-sampling at 1, 2, 4, …, 128 (relative to mesh frame size)
  - Down-sampled mesh LUT will by bi-linearly interpolated by LDC H/W
- Bi-cubic or bi-linear pixel interpolation
- Output LUT
  - Second output image through LUT
- External frame buffers
  - YUV422: UYVY (8-bit)
  - YUV420: NV12 (8-bit or 12-bit)
  - Y-only
  - 420UV-only



LDC Back Mapping: Definitions

- 2D projective transform before distortion correction

$$\begin{bmatrix} h_{aff} \\ v_{aff} \\ z \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix} \bullet \begin{bmatrix} h_u \\ v_u \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} h_p \\ v_p \end{bmatrix} = \begin{bmatrix} h_{aff} \\ v_{aff} \end{bmatrix} \bullet \frac{1}{z}$$

(h_u, v_u): U13
(a, b, d, e): S16Q12
(c, f): S16Q3
(g, h): S16Q23
(h_{aff}, v_{aff}): U16Q3 (1/8 pixel resolution)
z: U16Q14

(h_p, v_p): U16Q3 (1/8 pixel resolution)

- Mesh based distortion correction
  - $(\Delta h, \Delta v)$ is bilinear interpolated from mesh LUT given (h_p, v_p)
  - Mesh LUT has optional 2^m 2D down-sampling (m = 0,1,…,7)
    - Mesh LUT size: ceil(mesh_frame_width / 2^m) + 1 by ceil(mesh_frame_height / 2^m) + 1

$$\begin{bmatrix} h_d \\ v_d \end{bmatrix} = \begin{bmatrix} h_p \\ v_p \end{bmatrix} + \begin{bmatrix} \Delta h \\ \Delta v \end{bmatrix}$$

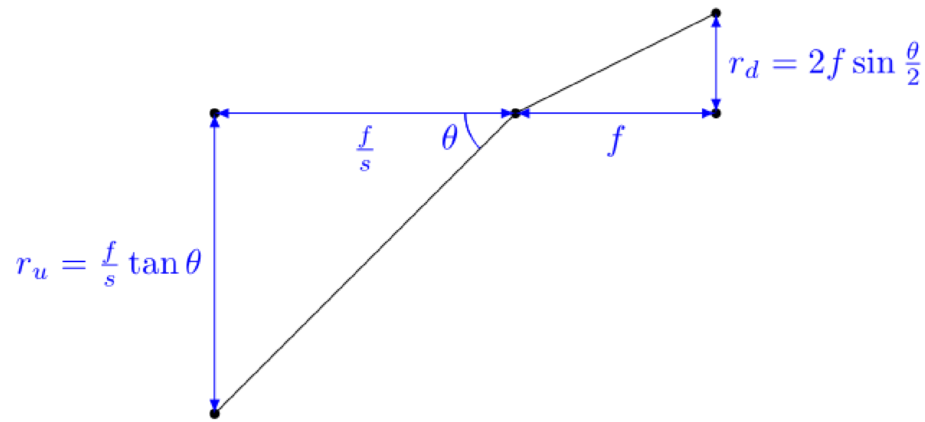
(h_d, v_d): U16Q3 (1/8 pixel resolution)
(h_p, v_p): U16Q3 (1/8 pixel resolution)
$(\Delta h, \Delta v)$: S16Q3 (1/8 pixel resolution)

LDC Back Mapping: Mathematical Lens Model

- Equisolid projection $r_d = 2f \bullet \sin\frac{\theta}{2}$
- Pinhole perspective projection $r_u = f/s \bullet \tan\theta$

$$\frac{r_d}{r_u} = s \bullet \frac{\cos\theta}{\cos\frac{\theta}{2}}$$

- Parameters
  - Image size: 1280x1280
  - Image center: (640, 640)
  - f (focal length): 1280/4/sin(45º)
  - s (mesh coverage): 4
  - m (down-sampling): 4



- Mesh LUT

$$r_u = \sqrt{(h_p - h_c)^2 + (v_p - v_c)^2}$$
$$\theta = \arctan(r_u \bullet s / f)$$
$$f_c = \frac{r_d}{r_u} = s \cdot \frac{\cos\theta}{\cos\frac{\theta}{2}}$$

$$\Rightarrow \begin{bmatrix} h_d \\ v_d \end{bmatrix} = \begin{bmatrix} h_c \\ v_c \end{bmatrix} + \begin{bmatrix} h_p - h_c \\ v_p - h_c \end{bmatrix} \bullet f_c$$

LDC Back Mapping: From Lens Model to Mesh LUT

- Distortion correction is specified by offsets stored in a mesh table

$$\begin{bmatrix} \Delta h \\ \Delta v \end{bmatrix} = \begin{bmatrix} h_d \\ v_d \end{bmatrix} - \begin{bmatrix} h_p \\ v_p \end{bmatrix}$$

$(\Delta h, \Delta v)$: S16Q3 (1/8 pixel resolution)
(h_d, v_d): U16Q3 (1/8 pixel resolution)
(h_p, v_p): U16Q3 (1/8 pixel resolution)

- Mesh LUT of WxH frame with down-sampling rate 2^m (m=0,1,…,7)
  - Interleaved h and v offsets in raster scan order
  - Mesh size: ceil(W/2^m)+1 by ceil(H/2^m)+1
  - Mesh coverage scale: s

```
W = 1280;                hc = W/2;
H = 1280;                vc = H/2;
f = W/4/sin(pi/4);  s = 2;  m = 4;
[h_p, v_p] = meshgrid(0:W, 0:H);
r_u = sqrt((h_p-hc).^2 + (v_p-vc).^2);
theta = atan(r_u * s / f);
f_c = s * cos(theta) ./ cos(theta/2);
h_d = hc + (h_p-hc) .* f_c;
v_d = vc + (v_p-vc) .* f_c;
h_delta = round((h_d - h_p) * 8);
v_delta = round((v_d - v_p) * 8);
mh = h_delta(1:2^m:end, 1:2^m:end)';
mv = v_delta(1:2^m:end, 1:2^m:end)';
dlmwrite('mesh.txt', [mh(:), mv(:)], 'delimiter', ' ');
```

LDC Back Mapping: From Lens Specification to Mesh LUT

- Read the distortion specification table (angle and height)

```
1 function lut = read_spec(spec_file, pitch_in_mm)
2 lut0 = dlmread(spec_file);
3 theta = lut0(:,1)/180*pi;
4 lut = [theta, lut0(:,2)/pitch_in_mm];
```

Angle (deg)	Height (mm)
0	0
0.89	0.01405228
1.78	0.02810717
2.67	0.04216725

- Back map

```
1 function [h_d, v_d] = xyz2distorted(x, y, z, hc, vc, spec_file, pitch_in_mm)
2 xt = x-hc; yt = y-vc; zt = z*ones(size(xt));
3 [phi, r] = cart2pol(xt, yt);
4 theta = atan2(r, zt);
5 lut = read_spec(spec_file, pitch_in_mm);
6 r_d = interp1(lut(:,1), lut(:,2), theta);
7 [h_d, v_d] = pol2cart(phi, r_d);
8 h_d = h_d + hc; v_d = v_d + vc;
```

If necessary, insert your change of camera view point as a transform on 3D points (xt, yt, zt) here between line 2 and 3.

- Generate the mesh LUT in GNU Octave or Matlab

```
1 function [] = gen_lut(spec_file, pitch_in_mm, f_in_mm, W, H, hc, vc, s, m)
2 f = f_in_mm/pitch_in_mm;
3 [h_p, v_p] = meshgrid(0:W, 0:H);
4 [h_d, v_d] = xyz2distorted(h_p, v_p, f/s, hc, vc, spec_file, pitch_in_mm);
5 h_delta = round((h_d - h_p)*8);
6 v_delta = round((v_d - v_p)*8);
7 mh=h_delta(1:2^m:end, 1:2^m:end)';
8 mv=v_delta(1:2^m:end,1:2^m:end)';
9 dlmwrite('mesh.txt', [mh(:), mv(:)], 'delimiter', ' ');
```

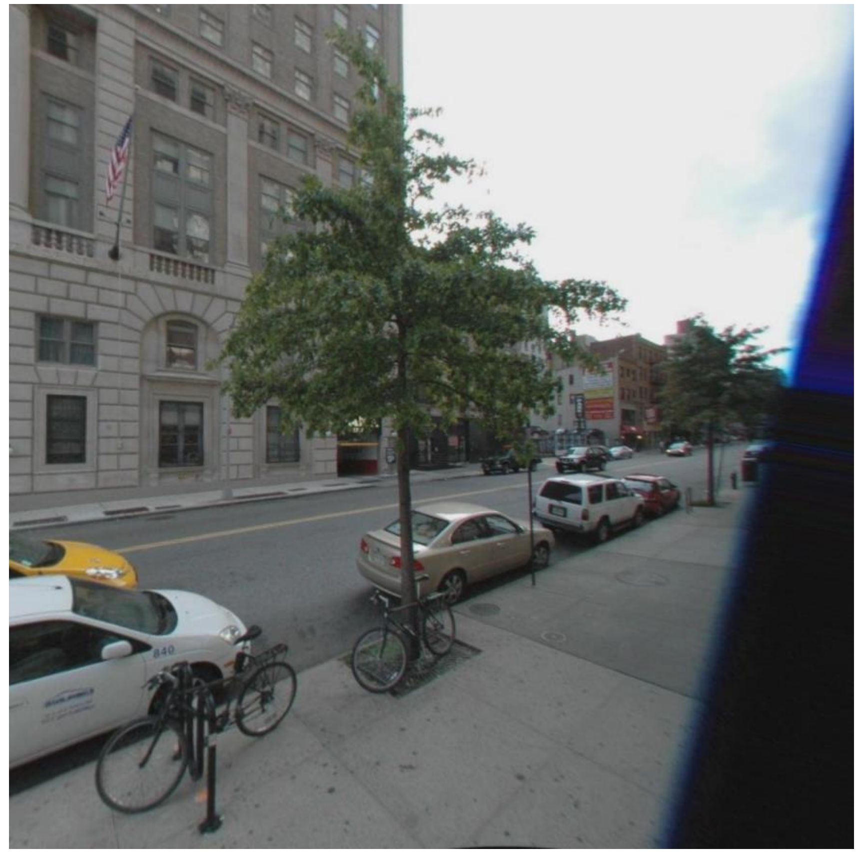
If necessary, clip (h_d, v_d) into your image boundary between line 4 and 5

Mesh LDC Sample Images

- Perspective front view
  - Input image: 2864x2864 (fisheye view)
  - Output image: 1280x1280 (90-degree H/V front view)
  - Mesh: perspective + lens distortion (m=4)

Mesh LDC Sample Images

- Perspective side views
  - Input image: 1280x1280 (fisheye view)
  - Output images: 1280x1280 (dual 90-degree views with 90-degree separation)
  - Mesh: perspective + lens distortion (m=4)



Mesh LDC Sample Images

- Cylindrical panorama
  - Input image: 1280x1280 (185-degree fisheye)
  - Output image: 1280x720 (H: 180-degree; V: 90-degree)
  - Mesh: panorama + perspective + lens distortion (m=4)

Mesh LDC Artifacts

- Perspective transform + mesh model for radial distortion
  - 90-degree perspective view (30-degree and 2-degree rotations)
  - Mesh (m=3): s=1 (left, 109-degree mesh coverage) or 4 (right, 160-degree mesh coverage)
- Both perspective transform and mesh change with s (for mesh coverage)
- Artifacts show up around edges for s=4 (for mesh) in the example below
  - Due to precision of perspective transform combined with mesh down-sampling
  - NOT an aliasing-only or mesh precision-only problem
  - May use mesh only to avoid artifacts (turn off perspective transform)
  - Mya use lower mesh down-sampling to avoid artifacts (but mesh LUT becomes large)

Notes on Mesh LDC

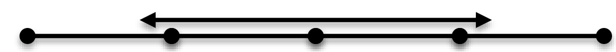
- Artifacts may show up if 2D transform and mesh LUT are used together
  - Projective transform (brute force) has limited precision
  - Use a dedicated mesh (mesh-only) for your view to avoid artifacts
- Use projective transform only for perspective changes
  - When no lens distortion exists (mesh is disabled)
  - May use mesh only as well, but not as flexible
- Aliasing may happen at high down-sampling rate
  - Perspective change may result in high down-sampling rate
  - Mesh LDC has no internal anti-aliasing filter
- One mesh cannot cover the entire 180-degree perspective view
  - Mesh is specified relative to the mesh frame size
  - A mesh + an input image → a distortion-corrected image
  - Fully corrected 180-degree perspective view goes to infinity
- Increase "m" to reduce mesh size
  - Small m (e.g., 0,1,2) results in large mesh LUTs for external storage
  - Small m causes high DDR traffic for loading mesh LUT into LDC frame by frame

Fisheye Lens Calibration for Mesh LDC

- Find the center of your lens accurately
  - Point the camera to the intersection of a horizontal line and a vertical line
  - Adjust camera so that both lines are as straight as possible
  - Capture the LDC input image from camera
  - Look at the image to verify that both lines are straight
  - Read out the image coordinate of the intersection which is the lens center
- Find the fisheye distortion function for your lens
  - Lens vendors typically provide a distortion function
  - Typically as a LUT from the angle of incoming ray (degree) to the image height (mm)
  - Reformat the LUT into 2 columns as a text file (degree and mm)
- Other information
  - Find the center and focal length in mm of your lens
  - Find the pixel pitch in mm of your image sensor
  - LUT down sampling rate m
  - Run the matlab script to generate mesh LUT
  - Capture one LDC input image
- Input all above information into tuning tool
- Check the tuning tool preview of the corrected image

Programming Mesh LDC

- Internal pixel block buffer
  - LDC fetches up to 21KB luma and/or 15KB chroma pixels for each block automatically
  - Including input image block with PIXPAD and other overhead
- Internal mesh LUT block buffer
  - LDC fetches up to 5KB for each block automatically (10KB total with ping-pong buffering)
  - LUT size for each block + memory banking overhead <= 5KB
    - Safe condition (may not be tight) when affine/perspective transform is off
      - floor(OBW/2^m + 6) * floor(OBH/2^m + 3) <= 1280



- Processing block parameters
  - OBW: 8, 16, 24, 32, ……, 248
  - OBH: 2, 4, 6, 8, ……, 254
  - PIXPAD: 0, 1, 2, 3, ……, 15
- I/O image buffers
  - Input from DDR, or SL2
  - Output to DDR, or other processing blocks via SL2
- Limitations
  - Processing blocks per row: up to 1023 blocks
  - Processing pixel block size: up to 1023x1023 pixels

Programming Mesh LUT

- Input frame size
  - The input image available in memory
  - Anything going outside will be clipped back
- Output frame size
  - The output image to produce
- Mesh frame size
  - Decide the mesh LUT size
  - Anything going outside will be clipped back
- Mesh LUT formats
  - In 2-column text file: created by the Matlab code earlier and used as tuning tool input
  - In DCC xml file: in H/W format with proper row alignment
- Mesh LUT in DDR (i.e., H/W format)
  - One 32-bit word for each entry (little endian)
    - 16 LSBs: vertical offset
    - 16 MSBs: horizontal offset
  - Mesh LUT row alignment: 16-Byte alignment
  - Tuning tool (DCC) shall take care of the conversion automatically
    - From mesh LUT text file to DCC xml file

Mesh LUT Down-sampling

- Mesh down-sampling
  - The mesh LUT as a vector field is quite smooth for typical fisheye lens distortion
  - Down-sampling at 16x16 (m=4) typically has little impact on image quality
  - Lens calibration error may be larger than error caused by mesh down-sampling

- Verify mesh down-sampling with tuning tool
  - Generate full mesh LUT and down-sampled mesh LUTs (m=0,1,2,…)
  - Use tuning tool to generate output images on test scenes for above LUTs
  - Compare full mesh output image to down-sampled mesh output images
  - Pick the largest down-sampling factor with acceptable image quality

Programming Mesh LDC Projective Transform

- Using projective transform and mesh LUT together may be problematic
  - Small edge artifacts may show up
    - Caused by the limited precision of projective transform with integer implementation
- Coefficients (a~h) are programed in 16-bit registers
  - Some transforms have a~h beyond 16-bit range and therefore cannot be used in LDC
  - Use mesh only instead
  - Coefficients depend on the mesh LUT
- LDC clips “z” to 16-bit (U16Q14) internally
  - Some transforms may result in z beyond 16-bit range and therefore cannot be used
  - Use mesh only instead
  - This problem can be detected by calculation or looking at tuning tool output image

$$\begin{bmatrix} h_{aff} \\ v_{aff} \\ z \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix} \bullet \begin{bmatrix} h_u \\ v_u \\ 1 \end{bmatrix}$$

(h_u, v_u):	U13
(a, b, d, e):	S16Q12
(c, f):	S16Q3
(g, h):	S16Q23
(h_{aff}, v_{aff}):	U16Q3 (1/8 pixel resolution)
z:	U16Q14