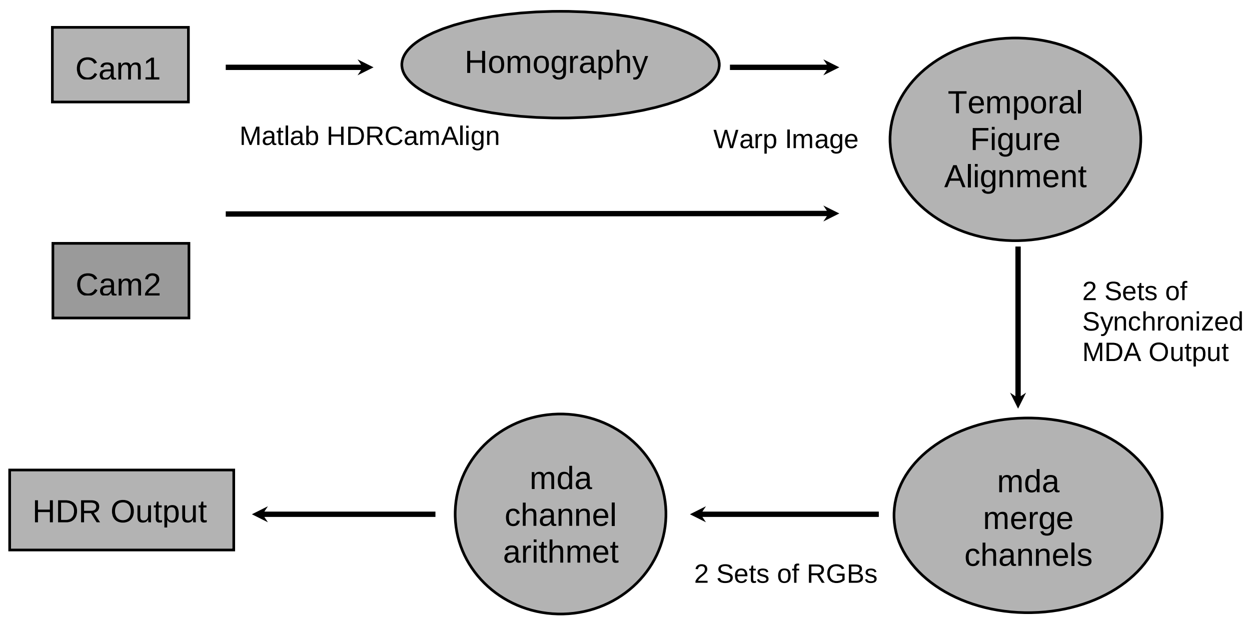
HDRCamera Project

1) Pipeline Diagram:



2) Details on the implementation of Simple Homography:

Files: (1) mda-homography.C (2) SimpleHomography.hh (3) SimpleHomography.C

Basic mda-homography Usage:

```
mda-homography [Options] metadata.xml < input.mda > output.mda
```

mda-homography reads both multi-dimensional array stream from input mda file and homography matrix from metadata, then passes the stream to SimpleHomography to do warping. The SimpleHomography is resolution-independent and the homography matrix is normalized, so before using it in rational resampling, mapping image from original dimension to [-1.. 1] and then back to output dimension is necessary. Here's how it's computed:

$$\begin{bmatrix} X_o / 2 & 0 & X_o / 2 \\ 0 & Y_o / 2 & Y_o / 2 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} H_N \end{bmatrix} * \begin{bmatrix} 2 / X_i & 0 & -1 \\ 0 & 2 / Y_i & -1 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

Note: (X_o, Y_o): output dimensions
(X_i, Y_i): input dimensions
H_N: normalized homography matrix

Once we have the matrix computed, then it's ready to do resampling. However, resampling is only done horizontally, so we do it first on X and then transpose the whole images and then do it horizontally again (Y of the original image) and transpose it back. Simple Homography will resample one frame after another, start from the first frame to the last, from bottom to top within each frame.

Let's say the transformation matrix we get from the calculation above is: $\boxed{\begin{matrix} a & b & c \\ d & e & f \\ g & h & i \end{matrix}}$

Basic rational resampling formula is: a'x + b' / c'x + d'

The actual computation:

1. When doing resampling horizontally:
   - a' = a
   - b' = by + c
   - c' = g
   - d' = hy + i

2. When doing resampling vertically:
   - a' = e
   - b' = dx + f
   - c' = h
   - d' = gx + i

3) <u>Basic idea on how Matlab computing homography matrix:</u>

Quite a few scripts we're currently using in HDRCamAlign can be obtained from Peter's Functions for Computer Vision by Peter Kovesi. Open hdrcamalign.m to see the details. First, it reads both images of the checkerboard and then finds a list of corner points. Then we map the points to normalized coordinates [-1 .. 1]. Once we have two sets of matching points, ransacfithomography will "robustly fit a homography to a set of putatively matched image points." Normalized matrix will be written to metadata.xml automatically, so you can just copy file over to where you execute the mda-homography.

Note. You will notice that one of the image appears to be flipped. It's because of the beam splitter we're using. However, you don't need to worry about it here, hdrcamalign will flip one of the image before find the corner points.

4) <u>Instructions on how to use mda-homography</u>:

First, get one photo or video from each camera. The original file should be in MTS. Extract one frame from each of them, rename them to cam1.jpg and cam2.jpg. (The checkerboard has to be included in the image) You can either run mts2mda on MTS and mda-toimage to get single frame, or you can try "ffmpeg -i input.MTS -s hd1080 -f image2 cam-%03d.png” to convert frame to images (You can terminate the process once the frame you need is extracted).

Run Matlab, put both images in the same folder with HDRCamAlign scripts, enter “hdrcamalign” in Matlab. (You might need to manually set the path in Matlab before you’re able to run it) Homography matrix will be automatically computed once it’s done and written to metadata.xml (See the directory panel on the left to find where the file is)

Locate the metadata, move it to where you store cam1.mda. Run “mda-homography metadata.xml < cam1.mda > cam1_out.mda”. cam1_out.mda is transformed, the output we need.

Note: If you see the error: “ *** glibc detected *** corrupted double-linked list ” while running mda-homography, it’s likely that the homography matrix you’re using is somehow incorrect. Verify and try again.