Research Log

JeffGWood@mavs.uta.edu

August 5, 2016

March 30, 2016	Established research log after 3 hours of learning new LaTeX
April 2, 2016	Added some additional comments to the **Process**
April 3, 2016	Have been reading [Shum2007] [1]. **Question for Kamangar:** regarding [Shum2007] [1] about difference between: • **Camera Plane** : Cooridinates u,v • **Focal Plane** : Cooridinates s,t
April 11, 2016	Reviewing blog articles located at: • `https://erget.wordpress.com/2014/02/01/calibrating-a-stereo-camera-with-opencv/` • `https://erget.wordpress.com/2014/02/28/calibrating-a-stereo-pair-with-python/` • `https://erget.wordpress.com/2014/03/13/building-an-interactive-gui-with-opencv/` • `https://erget.wordpress.com/2014/04/27/producing-3d-point-clouds-with-a-stereo-camera-in-opencv/` for process to get webcam up and running. Previous issues related to fine-tuning *block matching* parameters. Need to review sources at list at bottom of `http://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html` to understand.
April 19, 2016	Made adjustments to python for image acquisition scripts (from blogs mentioned on April 11, 2016.) **NOTE:** Consider creating rig with glue to keep stereo camera placement / direction constant.
April 19, 2016	**UPDATE:** Error with `calibrate_cameras` python code causing linux machine to crash. If can't be resolved switch over to MacBook. **NOTE:** Package should be setup by calling `$ python setup.py install`.
April 19, 2016	**UPDATE:** Crash due to recursive shell call and was fixed. OpenCV not detecting all chessboard corners. Will try a new board.
April 20, 2016	Did small amount of work on **Change of Reference** section in the paper. Added a section to the intro containing a map of commonly used symbols and notation.

April 29, 2016

Read following sections of [Chen1993] [2]:

- Abstract
- Introduction
- Visibility Morphing

SUMMARY: Explicit Geometry is ignored (i.e. surface mesh and 3d-points). Geometry is kept in 2-d. Whereas Image Morphing interpolates between *pixel intensity values in fixed locations* the method in this article interpolates between *pixel locations with (relatively) fixed intensity values*. **Question:** Sections read mention that pixel positions are stored in 3d (3-tuple) data structure. I'm not sure I understand this correctly, since

1. This would effectively make this structure a point cloud (but no mention of it in the paper).
2. There is no mention of special "depth-based" hardware or cameras (Far as I know this is upposed to be a regular image).

April 30, 2016

Checked understanding of *epipolar constraint* through reading of [Hartley2004] [3] and its derivation of

$$\begin{aligned} {}'\mathbf{x}^T \cdot \mathbf{E} \cdot \mathbf{x} &= {}'\mathbf{x}^T \cdot [\mathbf{t}]_\times \cdot \mathbf{R} \cdot \mathbf{x} \\ &= {}'\mathbf{x}^T \cdot {}'l \end{aligned}$$

and creation of MatLab code verifying this.

I may have been mistaken about relation of **Fundamental Matrix** and **Essential Matrix**.

My current understanding is the *Fundamental Matrix* describes point/epipolar line correspondance for images under **scale invariant** conditions (i.e. point correspondance and Fundamental matrix does not change when one image (or both images) are scaled (uniformly or omni-directionally).

Essential Matrix describes point/epipolar line correspondance for images under **normalized** conditions (i.e. unit-length is set equal to focal-length, and projection center is set at $(0, 0, 1)$.

May 2, 2016

Additional wording to Stereo-vision section. I am unsure of best order to present ideas related to *multi-view* geometry.

May 18, 2016

Reviewed [Chen1993] [2] Section 2. Consider reviewing follow relevant articles:

- Disparity [Gosh89]
- Optical Flow [Nage86]
- Look-up tables [Wolb89]
- 3d scenes [Pogg91]

Working on MatLab code to pick correspondig points in stereo-images, and calculate pixel offset vectors.

May 19, 2016

Read Section 2.3 of [Chen1993] [2]. View interpolation is limited by:

- **Penumbra**: pixels visible in one source image *but not both*
- **Umbra**, pixels visible in neither source image, and *invisible* in destination image.
- **Holes**, pixels visible in neither source image, but *visible* in destination image.

Calculatred formula for *pre-displaced* quad-pixel calculation using a bi-linear interpolation as:

$$\mathbf{P}(u, v) = \mathbf{P}(0, 0)\cdot(1-u)\cdot(1-v)+\mathbf{P}(1, 0)\cdot u\cdot(1-v)+\mathbf{P}(0, 1)\cdot(1-u)\cdot v+\mathbf{P}(1, 1)\cdot u\cdot v$$

May 20, 2016 Derived formula for uv calculation using *geometry matrix*, *blending matrix* and *basis vectors* of $\mathbf{u} = [u\ 1]^T$ and $\mathbf{v} = [v\ 1]^T$

$$x_{uv} = \begin{bmatrix} u & 1 \end{bmatrix} \begin{bmatrix} -1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x_{00} & x_{01} \\ x_{10} & x_{11} \end{bmatrix} \begin{bmatrix} -1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} v \\ 1 \end{bmatrix}$$
$$y_{uv} = \begin{bmatrix} u & 1 \end{bmatrix} \begin{bmatrix} -1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} y_{00} & y_{01} \\ y_{10} & y_{11} \end{bmatrix} \begin{bmatrix} -1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} v \\ 1 \end{bmatrix}$$

Question for Kamangar: Is there a way given x and y to solve for u and v?

May 22, 2016 Added more to thesis document.

Worked on singular-value of previous blending equation. where:

$$\begin{bmatrix} x_{uv} & 0 \\ 0 & y_{uv} \end{bmatrix} = \begin{bmatrix} \mathbf{u} & \mathbf{0} \\ \mathbf{0} & \mathbf{u} \end{bmatrix}^T \begin{bmatrix} \mathbf{M} & \mathbf{0} \\ \mathbf{0} & \mathbf{M} \end{bmatrix}^T \begin{bmatrix} \mathbf{X} & \mathbf{0} \\ \mathbf{0} & \mathbf{Y} \end{bmatrix} \begin{bmatrix} \mathbf{M} & \mathbf{0} \\ \mathbf{0} & \mathbf{M} \end{bmatrix} \begin{bmatrix} \mathbf{v} & \mathbf{0} \\ \mathbf{0} & \mathbf{v} \end{bmatrix}$$

where

$$\mathbf{u} = \begin{bmatrix} u \\ 1 \end{bmatrix}, \mathbf{v} = \begin{bmatrix} v \\ 1 \end{bmatrix}, \mathbf{X} = \begin{bmatrix} x_{00} & x_{01} \\ x_{10} & x_{11} \end{bmatrix}, \mathbf{Y} = \begin{bmatrix} y_{00} & y_{01} \\ y_{10} & y_{11} \end{bmatrix}, \text{ and } \mathbf{M} = \begin{bmatrix} -1 & 1 \\ 1 & 0 \end{bmatrix}$$

May 23, 2016 Read [Chen1993] [2] section 2.4 on *Block Compression*.

SUMMARY: Blocks are established established by *threshold* where each block contains pixels that are *offset by no more than the threshold*, allowing all pixels to be offset at once.

Question for Kamangar: Doesn't this assume that all pixels in the block have a uniform offset?

Working on MatLab program to perform pixel offsets of corresponding points (i.e. assign corresponding points to pixels in MatLab by non automatic methods)

May 24, 2016 Read following sections from [Chen1993] [2]:

- Implementations (3)
  - Preprocessing (3.1)
  - Interactive Interpolation (3.2)
  - Examples (3.3)
- Applications (4)
  - Virtual Reality (4.1)
  - Motion Blur (4.2)

Question for Kamangar: With regards to Section 3.1 and Section 1, why is a graph structure needed? Why is it a lattice?

Question for Kamangar: With regards to Section 4.1, I don't understand the concepts of *temporal anti-aliasing* and *super-sampling*?

Made additional changes / added material to thesis document.

May 25, 2016	Was using figures from `http://www.robots.ox.ac.uk/~vgg/hzbook/hzbook2/HZfigures.html` as test images, which may not be best source as there white borders, appear to be up-sampled, and do not contain (extrinsic) calibration info. Consider using images located at `http://vision.middlebury.edu/stereo/data/scenes2014/` that contain meta-info including (intrinsic) calibration info.
May 29, 2016	Finished [Chen1993] [2]. Not sure if remaining article is of consequence. Finished MatLab program for *animating / hand-drawing* (See wording in [Chen1993] [2]) offset vectors. Program performs offsets in 2-dimensional space. Conisder adding automatic *feature correspondance* and *z-buffer* information from depth map images avaiable on MiddleBury database.
May 30, 2016	Point-correspondances do not follow even pattern as indicated in [Chen1993] [2]: *Bi-linear coordinates* and *quad partitionions*; May be better to use *Barycentric coordinates triangle partitions.* Read on MatLab `tform`, `maketform`, and `Delaunay` triangles for purpose of image partitions.
June 1, 2016	Read and finished [Park2003] [4]. **SUMMARY:** Multiple sections including *point correspondance* and *interpolation*. **Point correspondance**: Breaks images into rectangular partitions. Gets maximum horizontal and vertical pixel gradients using *Sobel operator* in each partition. The maximum gradient in each partition is thresholded to disregard homogeneous and textured regions. **Interpolation**: The images are partitioned with *Delaunay triangulation* using the point correspondances as triangle vertices. **Question for Kamangar:** Article published seems to be vastly different depending on source (See `Park2003` folder). ScienceDirect version has more math and detail (maybe too much since it details what a *Sobel filter* is). Why would critical information, including algorithm steps and details, be ommitted?
June 2, 2016	Reviewing PDF at `https://staff.fnwi.uva.nl/l.dorst/hz/chap11_13.pdf` for information on *tri-focal tensor*. Don't understand *practical* calculation of *fundamental matrix* from *Singular Value Decomposition* and *Linear Least Squares* (i.e. don't understand LLS calculation from SVD).
June 3, 2016	Working on implementing *triangle patch transform* in MatLap (using previously mentioned `delaunay`, `tform`, and `maketform` functions) needed for [Chen1993] [2] and [Park2003] [4].
June 4, 2016	Continuting work on getting triangular patches transformed in MatLab. Will use `affine2d` and `imwarp` instead of `maketform` and `imtransform`. Spent several hours on a false start trying to implement line drawing on pixel data, in order to implment polygon seperation. Finally found MatLab's `roipoly` function which does what I need.

June 5, 2016 Almost done with MatLab triangle interpolation program. Hoping to have something to show Kamangar in the next few days.

Was reading up on image-segmentation as a way to improve feature detection through masking. Came accross references to **spectral clustering** which I still don't understand after data mining class. Was reading tutorial at `http://classes.engr.oregonstate.edu/eecs/spring2012/cs534/notes/Spectral.pdf` for starters.

June 8, 2016 Finalized most recent changes to MatLab program. It performs interpolation (between *source* and *destination* images of triangular patches defined by Delaunay triangularization of point correspondances from stereo images (See `Wood_Kamangar/StatusReports/StatusReport_00/Images`). Delaunay triangularization is performed on the source image only then extended to the corresponding points in the destination image so the arrangement of Delaunay triangles remains the same between images.

Summary of results is as follows:

- Triangles confined to one disparity region (See statue head in `image_source.png`, `image_destination.png`, and `truedisp.row3.col3.pgm`) show few artifacts and minimal blurring.
- Triangles crossing disparity regions or containing pixels occluded in the source or destination images (see camcorder tripod and lamp stand) have visibly more artifacts.

Started reading first page (*Abstract* and *Introduction* sections) of [Sharstein2002] [5].

June 9, 2016 Continuing to read [Scharstein2002] [5].

SUMMARY: Disparity can be defined by two ideas:

- *Human Vision* : Difference in location of features in the left and right eye.
- *Computer Vision* : Inverse depth. Can be treated as a 3-dimensional projective transformation (collineation or homographyv)of 3-d space (X,Y,Z).

Define fllowing terms:

- **Disparity Map**: $d(x, y)$
- **Disparity Space**: (x, y, d)
- **Correspondance**: Pixel (x, y) in reference image r and corresponding pixel (x', y') in matching image m given by $x' = x + sd(x, y)$ and$y' = y$ (assuming horizontal displacement *only*), where $s = \pm 1$ is chose do d is always positive.
- **Disparity Space Image**: Any function or image defined over continous or dispartiy space.

June 11, 2016 Continuing to read [Scharstein2002] [5]:

SUMMARY: Algorithms can be ordered in 4 common subsets:

1. Matching cost computation;
2. Cost (support) aggregation;
3. Disparity computation / optimization;
4. Disparity refinement;

Two main types of agorithms:

- **Local**: Including *Squared Intensity Differences* and *Absolute intensity differences*.
- **Global** Includeing *Energy minimizatio*.

Continuing to read up on *Spectral Clustering* and *Laplacian embedding* for uses in image segmentation.

June 14, 2016 Working on implementing [Park2003] [4] in MatLab.

Also working on implementing Spectral Clustering (for images) in MatLab. Started working on `fnDistance.m` to calculate pixel distances (*Distance Matrix*) for vectorized (row major and column major) images, needed for segmentation through spectral clustering.

June 16, 2016 Added some additional text regarding the *epipolar constraint* to the thesis document.

June 17, 2016 Finished implmenting and testing `fnDistance.m` for distance matrix. Next finished working on and testing `fnSimilarity.m` implementing a *Similarity Matrix* for spectral clustering.

June 18, 2016 Wrote small amount additional text on *epipolar contstraint*, and verified understanding through MatLab functions.

June 20, 2016 Holding off on reading any more of [Scharstein2002] [5](*Have completed up to end of page 5*): May be too advanced for me and of little use; Compares methods, but does not go into enough detail about how to implement them. Instead reading [Scharstein1999] [6] which may be more my level.

Started reading in *Correspondance problem* section of [Scharstein1999] [6]. **SUMMARY:** Matching can be done via *Fearure based correspondacne* and *Area based correspondance*.

Feature based correpondance finds locally unique or identifiable pixels (i.e. Corners or edge gradients), matchingbetween images occurrs between these reduced set of points. Advantages are only a few points are necessary. Disadvantages are that disparity calculations are confined to these points, so interpoint disparity have to be calculated through interpolation and may not be accurate.

Area based correspondance occurrs over *regions in the image* instead of points used in feature correspondance. Advantages are a denser (and therefore more accurate) disparity map, but require assumptions about local disparity.

June 21, 2016 Continued reading [Scharstein1999] [6].

SUMMARY: 3 general methods are being differentiated:

- **Image Synthesis based on Stereo**: Uses stereo mathods for image creation.
- **Image Interpolation**: Similar to *Image Synthesis based on Stereo*, except mages generated must be on baseline, and baseline must be parallel to image planes.
- **Information from Many Images**: Includes image stitching and panoramic mosaicing.

Other sections involve summaries of various papers and methods published under each of the 3 categories.

Got further clarification on steps for coorespondance matching for *feature-based correspondance.*

1. **Preprocessing**: Color correction between stereo images for conconsitancy, and image warping through rectification so features occur at (approximatley) same horizontal distance reducing search area to the scanline.
2. **Cost Calculation**: Per-pixel cost calculation done as either a *square difference* or *absolute difference.*
3. **Aggregation**: The summing of the cost calculations over the window in question.
4. **Comparison / Calculation**: Window on feature trying to be matched is kept fixed.Window in corresponding image is moved along the scanline for a comparison of potential window aggregates. Correspondance with minimum aggregate (in difference of costs) is selected as the corresponding point in the image being scanned.
5. **Sup-pixel Calculation**: Not yet read. Could be smoothing.

Read up to section 2.2.5 *Disparity Selection* (PDF page 49, Numbered page 35). Stopped to read up on using Dynamic Programming to increase consistancy of stereo points and disparity, including following sourceses:

- `http://www.robots.ox.ac.uk/~az/lectures/opt/lect2.pdf`
- `http://www.cs.umd.edu/~djacobs/CMSC426/PS7.pdf`

June 22, 2016 Continued reading [Sharstein1999] [6]. I'm still unclear about the process (and use of) *Sub-Pixel Disparity Computation* mentioned in section 2.2.6.

I moved onto Chapter 3 (View Synthesis) and have been reading on *three-view rectification.* Read all of Section 3.1 (*Geometry*) (up to but not including PDF page 60, Numbered page 47).

SUMMARY: A new image I_3 is synthesisized from images I_1 and I_2, by estabishing reference frame containing camera centers $\mathbf{C_3}$, $\mathbf{C_1}$, and $\mathbf{C_2}$ respectively. The unit-length is established as the difference between camera centers $\mathbf{C_1}$ and $\mathbf{C_2}$. The positions are set along the x-axis such that $\mathbf{C_1} = [0,0,0]^\top$ and $\mathbf{C_2} = [1,0,0]^\top$. The xy-plane is oriented such that it contains $\mathbf{C_3} = [a,b,0]^\top$ (for some constants a and b).Images I_1 and I_2 are *horizontally rectified* (such that pixel-features occur at the same vertical position), through an *affine warp* to images I_1' and I_2' which occur in the xy-plane at $z = 1$. The synthetic image I_3 is produced from the horizontally rectified image I_3' which also occurs in the $z = 1$ plane.

Question for Kamangar: How can the homography matrix $\mathbf{H}_i = [\mathbf{R}_i|\mathbf{S}_i|\mathbf{O}_i - \mathbf{C}_i]$ be calculated if the vectors $\mathbf{R}_i$, $\mathbf{S}_i$, and $\mathbf{O}_i$ are unknown. How can they be determined from available information?

June 24, 2016	Added additional material to thesis document for *Epipolar constraint* section.
June 25, 2016	Added additional text to thesis document in *Epipolar constraint* and *Fundamental matrix* sections. Reading up on on *homographies* and *rectification* for [Scharstein1999] [6] and for derivation of *Fundamental matrix* for thesis document.
June 26, 2016	Started reading Chapter 2 of [Hartley2004] [3] for information regarding *Homographices*. Worked on graphics regarding *Epipolar constraint* for inclusion in thesis document.

June 27, 2016

Continued reading Chapter 2 of [Hartley2004] [3] containing information on *Homographies* for purpose(s) of deriving *Fundamental matrix* formula as well as understanding *Horizontal rectification* used for matching features along scanlines of images.

SUMMARY: Transformations of points in the image plane can be grouped into the following categories:

- **Isometries** (Denoted by $\mathbf{H}_E$): Transfomrations in $\mathbb{P}_2$ including *translation* and *rotation* (including composites of the two) that peserve *Euclidean*-distance. Tranformations are of the form

$$\begin{bmatrix} \epsilon\cos(\theta) & -\sin(\theta) & t_x \\ \epsilon\sin(\theta) & \cos(\theta) & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

  where $\epsilon = \pm 1$. Angles are preserved if $\epsilon = 1$, else if $\epsilon = -1$ angles are reversed (reflection accross an axis).
- **Similarity** (Denoted by $\mathbf{H}_S$): Transformations include *translation*, *rotation*, and *scaling*. Matrices are of the form

$$\begin{bmatrix} s\cos(\theta) & -s\sin(\theta) & t_x \\ s\sin(\theta) & s\cos(\theta) & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

  where s is the scaling factor. While *distances* are not preserved, the *ratio of distances* and *angles* are preserved.
- **Affine** (Denoted by $\mathbf{H}_A$): Transformations include all linear transformations of *translation*, *rotation*, *scaling*, and *shearing*. Matrices are of the form

$$\begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

- **Projective** (Denoted by $\mathbf{H}_P$): Transformations in $\mathbb{P}_2$ that are linear transformations in $\mathbb{R}_3$. Matrices are of the form

$$\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$$

June 29, 2016 Continuing to read [Hartley2004] [3] for *affine rectification*. Chapters of [Hartley2004] [3] include:

- **Chapter 2: Projective Geometry**:
  - **Section 2.1: Planar Geometry**:
  - **Section 2.2: The 2D projective plane**:

    Lines in $\mathbb{R}^2$ are detailed by $\mathbf{l} = [a, b, c]^\intercal$ and points as $\mathbf{x} = [x, y, 1]^\intercal$ such that $\mathbf{l}^\intercal \cdot \mathbf{x} = a \cdot x + b \cdot y + 1 = 0$. Coordinates $\mathbf{x} = [x, y, 0]^\intercal$ with a 0 instead of 1 in the last place represent a *point at infinity* since they are the only points where $a \cdot x + b \cdot y + c \cdot 0 = a \cdot x + b \cdot y + c' \cdot 0$ for the two *parallel* lines of $\mathbf{l} = [a, b, c]^\intercal$ and $\mathbf{l}' = [a, b, c']^\intercal$

    Cross product of points $\mathbf{x}$ and $\mathbf{x}'$ result in line $\mathbf{l}$ joining the two points (i.e. $\mathbf{x} \times \mathbf{x}' = \mathbf{l}$). Cross product of lines $\mathbf{l}$ and $\mathbf{l}'$ result in point $\mathbf{x}$ where intersection of two lines (i.e. $\mathbf{l} \times \mathbf{l}' = \mathbf{x}$).

    Circles and ovals can be reprsented by a *conic-matrix* of the form

$$\begin{aligned} 0 &= \mathbf{x}^\intercal \cdot \mathbf{C} \cdot \mathbf{x} \\ &= \begin{bmatrix} x & y & 1 \end{bmatrix} \cdot \begin{bmatrix} a & b/2 & d/2 \\ b/2 & c & e/2 \\ d/2 & e/2 & f \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \\ &= a \cdot x^2 + b \cdot xy + c \cdot y^2 + d \cdot x + e \cdot y + f \cdot 1 \end{aligned}$$

  - **Section 2.3: Projective transformations**:

    Point $\mathbf{x}$ on an image is mapped to point $\mathbf{x}'$ via a homography $\mathbf{H}$, such that $\mathbf{x}' = \mathbf{H} \cdot \mathbf{x}$. Because a point $\mathbf{x}$ lies on line $\mathbf{l}$ if $\mathbf{l}^\intercal \cdot \mathbf{x} = 0$, then because

$$\begin{aligned} 0 &= \mathbf{l}^\intercal \cdot \mathbf{x} \\ &= \mathbf{l}^\intercal \cdot \mathbf{H}^{-1} \cdot \mathbf{H} \cdot \mathbf{x} \\ &= \mathbf{l}^\intercal \cdot \mathbf{H}^{-1} \cdot \mathbf{x}' \end{aligned}$$

    the point $\mathbf{x}'$ lies on the line $\mathbf{l}'$ defined by $\mathbf{l}'^\intercal = \mathbf{l}^\intercal \cdot \mathbf{H}^{-1}$, or $\mathbf{l}' = \mathbf{H}^{-\intercal} \cdot \mathbf{l}$. Therefore a homograhy that gives a *point-mapping* of $\mathbf{x}' = \mathbf{H} \cdot x$ has a corresponding *line-mapping* of $\mathbf{l}' = \mathbf{H}^{-\intercal} \cdot \mathbf{l}$.

    Similarly, for a homography given by $\mathbf{x}' = \mathbf{H} \cdot \mathbf{x}$, the conic under the homography is given by

$$\begin{aligned} 0 &= \mathbf{x}^\intercal \cdot \mathbf{C} \cdot \mathbf{x} \\ &= (\mathbf{H}^{-1} \cdot \mathbf{x}')^\intercal \cdot \mathbf{C} \cdot (\mathbf{H}^{-1} \cdot \mathbf{x}') \\ &= \mathbf{x}'^\intercal \cdot \mathbf{H}^{-\intercal} \cdot \mathbf{C} \cdot \mathbf{H}^{-1} \cdot \mathbf{x}' \\ &= \mathbf{x}'^\intercal \cdot \mathbf{C}' \cdot \mathbf{x}' \end{aligned}$$

    where $\mathbf{C}' = \mathbf{H}^{-\intercal} \cdot \mathbf{C} \cdot \mathbf{H}^{-1}$.

  - **Section 2.4: A hierarchy of transformations**:

    See entry from June 27, 2016.

June 29, 2016 *...Continued*

- **Chapter 6: Camera Models**:
  - **Section 6.1: Finite cameras**:

Transformation from *world-coordinate* system $\mathbf{x}$ to *camera-coordinate* system ${}^{C}\mathbf{x}$ is given by ${}^{C}\mathbf{x} = \mathbf{R} \cdot (\mathbf{x} - \mathbf{c})$. The Camera in *world-space* occurs at $\mathbf{x} = \mathbf{c}$. *Camera-space* has the camera located at ${}^{C}\mathbf{x} = 0$ and includes an *image-plane* at $z = f$. All rays intersect the *image plane* at $z = f$ and converge on the origin ${}^{C}\mathbf{x} = 0$ which is known as the *camera center*. This results in points ${}^{C}\mathbf{x}$ in *camera space* being projected to points $\tilde{\mathbf{y}}$ in the *image plane* by means of the *projection matrix* $\mathbf{P}$ such that

$$\mathbf{P} \cdot {}^{C}\tilde{\mathbf{x}} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} {}^{C}x_1 \\ {}^{C}x_2 \\ {}^{C}x_3 \\ 1 \end{bmatrix} = \begin{bmatrix} f \cdot {}^{C}x_1 \\ f \cdot {}^{C}x_2 \\ {}^{C}x_3 \end{bmatrix}$$

$$= {}^{C}x_3 \cdot \begin{bmatrix} f \cdot {}^{C}x_1 / {}^{C}x_3 \\ f \cdot {}^{C}x_2 / {}^{C}x_3 \\ 1 \end{bmatrix} = {}^{C}x_3 \cdot \tilde{\mathbf{y}}$$

This results in points containing infinitley large values of x_3 being mapped to the same *principal point* of $\mathbf{y} = 0$ in the *image plane*. This assumes the *princpal point* is always located in the *image plane* at $\mathbf{y} = 0$. Projecting point $\tilde{\mathbf{x}}$ to the *image plane* with arbitrary *principal point* $\mathbf{p} = [p_x, p_y]$ requires modifying the *projection matrix* to include *camera-specific* parameters. The *camera calibration matrix* $\mathbf{K}$ is given as

$$\mathbf{P} \cdot {}^{C}\tilde{\mathbf{x}} = \begin{bmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} {}^{C}x_1 \\ {}^{C}x_2 \\ {}^{C}x_3 \\ 1 \end{bmatrix} = \begin{bmatrix} f \cdot {}^{C}x_1 + p_x \cdot {}^{C}x_3 \\ f \cdot {}^{C}x_2 + p_y \cdot {}^{C}x_3 \\ {}^{C}x_3 \end{bmatrix}$$

$$= {}^{C}x_3 \cdot \begin{bmatrix} f \cdot {}^{C}x_1 / {}^{C}x_3 + p_x \\ f \cdot {}^{C}x_2 / {}^{C}x_3 + p_y \\ 1 \end{bmatrix} = {}^{C}x_3 \cdot \tilde{\mathbf{y}}$$

June 30, 2016 **Question for Kamangar:** On pages 162 and 244, how is the ray back-projected from $\mathbf{x}$ by $\mathbf{P}$ (where $\mathbf{x} = \mathbf{P}\mathbf{X}$ and $\mathbf{P} = \mathbf{K}[\mathbf{R}|\mathbf{t}]$) given by the formula $\mathbf{X}(\lambda) = \mathbf{P}^{+}\mathbf{x} + \lambda\mathbf{C}$? How is the formula derived?

July 1, 2016 Added section called **Points and Lines in the Image Plane** in the **Background** section.

July 5, 2016 Continued adding text to **Background** section of Thesis Document, in *Epipolar Geometry* and *Intrinsic Calibration Matrix* sections.

July 11, 2016 Trying to consolidate knowledge (and explain in thesis document) behind the pinhole camera model. Specifically the concept of *focal-length* as it relates to *similarity of triangles*.

July 12, 2016 Started reading [Martin2008] [7].

July 13, 2016 Reading [Fusiello1999] [8]. Running throuh MatLab code at `http://www.diegm.uniud.it/fusiello/demo/rect/` to understand algorithm. [Fusiello1999] [8] gives more insight into *rectification* discussed on June 22, 2016:

SUMMARY: *Rectification of stereo images* warps each image so that points are (vertically) aligned with their conjugate epipolar lines, and so that the collection of epipolar lines (in each image) are parallel. This aids in the use of Dynamic Programming for searching of corresponding points along each *scan-line* of the rectified image.

Normally, when the *camera centers* do not lie in *focal planes*[1], the *epipolar lines* intersect at the *epipole*. When the *camera center* of image A is located in the *focal plane* of image B, the *epipolar lines* in image B will be parallel. Similary, when the *camera center* of image B is located in the *focal plane* of image A, the *epipolar lines* in image A will be parallel.

Rectification consists of transforming the cameras in each image such that the *camera centers* are co-planar

Question for Kamangar: My current understanding is this: *Rectification* of images is used to search along *scanlines* for *point correspondances*. In order to do *Rectification*, *point correspondances* are required. Doesn't this present a problem? It seems to be a *chicken and the egg* type problem.

July 15, 2016 Finished reading [Fusiello2000] [**?**]. Aside from details of algorithm and errors in experimental resuls, no more useful information gained since summarizing on July 13, 2016.

Since implementation is already done in `MatLab`, I'm porting methodology to `Python` using `OpenCV` and `OpenGL` in *final demonstration*.

Resumed reading of [Martin2008] [7].

July 17, 2016 Spent a couple of hours working on *demonstration* code in OpenGL and OpenCV.

July 18, 2016 Spending day working on thesis document. Sections worked on include:

- Intrinsic Calibration Matrix
- Fundamental Matrix

July 19, 2016 Continuing to add material to thesis document, including:

- Extrinsi Calibration Matrix
- Fundamental Matrix

Going back to reread first parts of Chapter 6 from [Hartley2004] [3], as I need clarification on some aspects of the *calibration matrix*. Namely, I *still* do not understand how $\mathbf{X}(\lambda) = \mathbf{P}^{+}\mathbf{x} + \lambda\mathbf{C}$ represents the equation of a ray passing through *optical center* $\mathbf{C}$ in *world space*, with *projection matrix* $\mathbf{P}$.

July 20, 2016 Added material on *fundamental matrix calculation from data* to thesis document. Reading additional material from [Hartley2004] [3] on *fundamenta matrix theoretical calculation*.

---

[1] May cause confusion depending on understanding of the terms *focal plane* and *retinal plane*. [Fusiello1999] [8] refers to *focal plane* as the plane containing the *optical center* and parallel to the *image plane*. The *image plane* is also referred to as the *retinal plane*. [Hartley2004] [**?**, Hartley2004]efers to *focal plane* as being synonymous with the *image plane*, but the *retinal plane* is the plane containing the *optical center* and parallel to the *image plane*. Here we are using the definition from [Fusiello1999] [8].

July 21, 2016 Continuing to read [Martin2008] [7]. See questions below.

Question for Kamangar: I don't understand the difference between *forward mapping* and *backward mapping*.

I'm a bit confused about most of the material being discussed in [Martin2008] [7]. Will read [Karathanasis1996] [9] for background on *disparity estimation using dynamic programming*.

UPDATE: My question on July 13, 2016 may have been worded wrong: The *dynamic programming* is used for estimating *disparity*, which in turn is used for *point correspondance*. The *dynamic programming* is not used DIRECTLY, in calcuating *point correspondance*.

Orignal question still holds though:

Question for Kamangar: I understand *ALL* of the following to be *TRUE*, which one needs to be *FALSE* (or my understanding revised):

- *Point correspondence* is needed to compute *rectifying homographies*.
- *Rectifying homography* is needed to compute *disparities*.
- *Disparity* is needed to compute *point correspondence*.

July 22, 2016 Started reading [Karathanasis1996] [9], no new information from first few sections.

July 25, 2016 Started woring on implmentation of *disparity estimation using dynamic programming* in `MatLab`. So far I have completed the *dynamic programming* aspect only. I need to work on:

- Seperation of image into seperate scanlines, where number is based on window size.
- Conversion of window values to values used in the dynamic programming.

The generic method (summary below) seems to be a little different than method described in [Karathanasis1996] [9].

SUMMARY: A left image L and right image R each contain many *scanlines*, each at the same vertical position. Though each image's scanline is 1-*dimensional*, each point in the scanline is a $k \times k$ square matrix of *normalized* pixel values (commonly referred to as a *Window*). The window centered at pixel i in L is denoted by vector $\mathbf{L}(i,k)$, and similarly the window centered at pixel j in R is denoted by vector $\mathbf{R}(j,k)$.

A feature at i in L is closely matched to the feature at j in R if the *sum of square differences* $SSD(i,j,k) = ||\mathbf{L}(i,k) - \mathbf{R}(j,k)||_2$ is minimal (ideally 0). The dynamic programming approach to disparity estimation attempts to minimize the sum of $SSD(i,j,k)$ over all possible i and j, by including a constant *occlusion cost* (OC) for instances when a window centered at i in L does not have a matching feature at j in R, and simlarly a window centered at j in R does not have a matching feature at i in L. The *matching cost* ($MC(i,j,m)$) at for the windows centered at i in L and j in R is then assigned to be the *minimum* of:

- $MC(i-1, j-1, m) + SSD(i,j,k)$
- $MC(i-1, j, m) + OC$
- $MC(i, j-1, m) + OC$

to a $(m+1) \times (n+1)$ table (where m is the number of window values (*image width* less $(k-1)$) in L, and n is the number of window values in R. In addition to the above assignments, we let

- $MC(0,0,m) = 0$ for the initial cost.
- $MC(s \cdot OC, 0, m)$ (for all $s \leq m$) to denote first s windows in L are occluded from R.
- $MC(0, t \cdot OC, m)$ (for all $t \leq n$) to denote first t windows in R are occluded from L.

July 27, 2016 Continued reading [Karathanasis1996] [9]

Made additional changes to python Demo using OpenCV and OpenGL. Still a long way from finished.

July 28, 2016 Resumed work on *disparity estimaion using dynamic programming* in `MatLab`. Completed seperating images into seperate scanlines, as well as windows into dynamic programming values. Calculated disparities based on this technique and included output in relative `statusreport_week11` folder.

July 31, 2016 — Decided to test *spectral clustering* routines `fnDistance` and `fnSimilarity` from June 5, 2016. Routines work on small images (approximaltey 100 pixels in size), but are bombing out `matlab` on larger images since for an image containing n pixels, the *Laplacian matrix* would be $n \times n$ in size requiring large amounts of memory. Put functions and test scripts in `Wood_Kamangar/StatusReports/StatusReport_12/`

I am looking into other methods of *image segmentation* including *graph-cuts* (described as the "gold-standard").

August 1, 2016 — Started reading [Mark1997] [10].

SUMMARY: Paper describes expanded algorithm for *view interpolation* that building on [Chen1993] [2]. Pixels (including z-buffer and color information) in source images (referred to in article as *reference frames*) are transformed to the new new frame (referred to in article as *derived frames*) via *Euclidean*-transformations and *Affine*-transformations.

The paper addresses problems associated with *holes* being proudced in the derived frame, which result from a number of sources. They inlcude pixels *occluded* in the reference frame. Another source are surfaces that are highly incident to the image plane in the refence frames, but more closely parallel to the image plane in the derived images. The occurance of holes can be addressed through the use of a *mesh* for surface reprsenation (similar to that resulting from a *point cloud*). This results in holes of the latter type (surfaces of different angles to the image plane) being filled. Holes of the former type (from occluded pixel areas) occur along a siloutte of the backround/foreground surfaces. Normally the mesh results in a distorted surface connecting that foreground and background surface. The proposed algorithm (referred to in the article as *compositing*) addresses this issue by keeping the surfaces distinct and seperate and filling in the missing pixels with those containing the maximum (farthest) z-value.

August 2, 2016 Finished reading [Mark1997] [10]. Still unclear about some aspects including details calculations in section **4.3 Connectedness Calculation**.

SUMMARY: The compositing algorithm works by transforming the pixels in each *reference* frame to seperate *transformed-reference* frames. Each pixel buffer contain position, z-buffer, and color information. Because a multiple pixels from a single *reference* frame can be mapped to the same pixel buffer in its *transformed-reference* frame, pential new pixel values are compared with those already in the pixel buffer, with those pixels containing the closest (minimum) z-value remaining in the buffer.

Another aspect of the proposed algorithm is the treatment of *rubber surfaces* that occur along the siloutte lines between the foreground and backround segments of the generated mesh surface. This is handled by the notion of *connectedness* of surfaces. Pixels with mesh vertices part of a single object surface are considered to be *highly-connected*, whereas mest trianges covering disjoint and seperate surfaces have *low-connectiveness*.

An additional concept the authors make use of is *confidence value*. The article references the confidence value as the ratio of a pixel's *solid angle* in the reference frame to the *solid angle* in the derived frame. It is essentially a measure of how much a surface is parallel to the image plane. Surfaces, whose normal vector turns *towards* the image plane when transforming from the reference frame to derived frame, result in *pixel holes* and have low confidence values. Sufaces, whos normal vector turns away from the image plane are oversampled and have high confidence values.

When selecting pixels for the *derived* frame a number of scenarios arise: If both pixels have high connectedness, the one with closer Z-value is used in the derived frame. If the Z-values are the same (within a tolerance), the pixel with higher *confidence value* is used. If only one of the pixels has high connectedness, that pixel is selected. If nneither pixel has high connectedness, the pixel with the higher confidence value is used. When dealing mesh triangles of low connectedness, instead of interpolating between the *foreground* and *background* surface textures, the surface texture with the *farthest* Z-value is used to approximate the occluded areas.

Question for Kamangar: I don't understand the explantion given for section **4.3 Connectedness Calculation**.

August 3, 2016 Started reading [Saito2002] [11]. I am trying to understand concept of *cross-ratios* for the article material. I also plan to add section regarding *homographies* to thesis document. I put `MatLab` code in `Wood_Kamangar/StatusReports/StatusReport_12/`

August 4, 2016 Continued reading [Saito2002] [11]. Will need to read [Saito1999] [12] for background on *projective grid space*. Summary of [Saito2002] [11] follows.

SUMMARY: [Saito2002] [11] describes a system used in *virtualized television* and *free viewpoint television*, and specifically with regards to televizing soccer matches. It defines a *projective grid space* (PGS) betweens two images I_1 and I_2. Instead of a cooridinate system where the basis vectors are all *orthoganal*, the PGS defines two of the basis vectors as being along the principal axis of each of the images being interpolated.

There exist *fundamental matrices* from I_1 to I_2 (denoted in the article as $\mathbf{F}_{12}$) and from I_2 to I_1 (denoted as $\mathbf{F}_{21}$) which transforms points in one image to *epipolar lines* in a the other image. Corresponding points $\mathbf{P}_1$ in I_1 and $\mathbf{P}_2$ in I_2 can similarly be transformed to epipolar lines in a mid-view image I_i by the fundamental matrices $\mathbf{F}_{1i}$ and $\mathbf{F}_{2i}$. The intersection of the two generated epipolar lines is the position of the corresponding point in $\mathbf{P}_i$ in I_i.

The viewing angle and position of the interpolated image I_i is confined to the baseline between two images I_1 and I_2 using *linear interpolation*. When a third image I_3 (viewed from a higher angle) is adde, the viewing angle and position are confined to the triangle connecting the optical centers of the 3 image planes using *barycentric interpolations*. The interpolation methods are used for both *pixel position* and *pixel color*.

Scene components are divided into 3 major components including:

- Players and Ball
- Field ground and goal
- Background including stands

pixel operations dependant on the component type to which it belongs. The *players and ball* component is considered to be *dynamic* since it is changing between fames. The *players and ball* components are actually sillouttted areas created from extracting the *field ground and goal* components. The pixels in the intermediate views are determined from interpolating pixel values between boundaries (along the epipolar lines) of the sillouetted areas. The *field ground and goal* components are transformed via *homographies* with the planes corresponding to the *field ground* and sides of the *goal post* treated as seperate planes. **Although not explicitly stated, I'm guessing pictures of the field ground (without the goal post or players) may also be taken before hand to account for pixels that might otherwise be occluded (with the inclusion of the players and goal post)**. The remaining area containing the *background and stands* are transformed with *image mosaicing* and a *plane at infinity*.

August 5, 2016 **UPDATE:** Discussed the matter of memory issues related to *spectral clustering* detailed on July 31, 2016 with `levine@uta.edu`. Due to the memory issues related to rendering moderate size images (300x200 pixels) I decided it might be better to:

1. Downsample original image to manageble size on which *spectral clustering* can be performed.
2. Extract edge regions of down sampled areas.
3. Partition original size image into manageble sub areas.
4. Perform *spectral clustering* on sub areas corresponding to edge regions extracted from downsampled image.
5. Join sub areas from image partitioning by mapping possibly non-equal segment labels between sub areas.

References

[1] Sing Bing Kang Heung-Yeung Shum, Shing-Chow Chan. *Image Based Rendering.* Springer Publishing, 1 edition, 2007. Available online at: `http://link.springer.com/content/pdf/10.1007%2F978-0-387-32668-9.pdf` Pages cited are **Book Page** Numbers. Formula for **PDF Page** Number is (**PDF Page Number** = **Book Page Number** + 17).

[2] Shenchang Eric Chen and Lance Williams. View interpolation for image synthesis. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '93, pages 279–288, New York, NY, USA, 1993. ACM.

[3] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision.* Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[4] Joon Hong Park and HyunWook Park. Fast view interpolation of stereo images using image gradient and disparity triangulation. In *Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on*, volume 1, pages I–381–4 vol.1, Sept 2003.

[5] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *Int. J. Comput. Vision*, 47(1-3):7–42, April 2002.

[6] Daniel Scharstein. *View Synthesis Using Stereo Vision.* Springer-Verlag, Berlin, Heidelberg, 1999.

[7] N. Martin and S. Roy. Fast view interpolation from stereo: Simpler can be better. In *Fourth International Symposium on 3D Data Processing, Visualization and Transmission*, Proceedings of 3DPVT'08, 2008.

[8] Andrea Fusiello, Emanuele Trucco, Alessandro Verri, and Ro Verri. A compact algorithm for rectification of stereo pairs, 1999.

[9] J. Karathanasis, D. Kalivas, and J. Vlontzos. Disparity estimation using block matching and dynamic programming. In *Electronics, Circuits, and Systems, 1996. ICECS '96., Proceedings of the Third IEEE International Conference on*, volume 2, pages 728–731 vol.2, Oct 1996.

[10] William R. Mark, Leonard McMillan, and Gary Bishop. Post-rendering 3d warping. In *Proceedings of the 1997 Symposium on Interactive 3D Graphics*, I3D '97, pages 7–ff., New York, NY, USA, 1997. ACM.

[11] Hideo Saito Makoto, Makoto Kimura, Satoshi Yaguchi, and Naho Inamoto. View interpolation of multiple cameras based on projective geometry. In *In: International Workshop on Pattern Recognition and Understanding for Visual Information*, 2002.

[12] H. Saito and T. Kanade. Shape reconstruction in projective grid space from large number of images. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, volume 2, page 54 Vol. 2, 1999.