Research Log

JeffGWood@mavs.uta.edu

May 23, 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 [ImageBasedRendering] [1]. **Question for Kamangar:** regarding [ImageBasedRendering] [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 [Chen93] [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 [Chen93] [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 [Chen93] [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 [Chen93] [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)

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.