REFERENCES

Hild, François, and Stéphane Roux. “Measuring Stress Intensity Factors with a Camera: Integrated Digital Image Correlation (I-DIC).” *Comptes Rendus Mécanique* 334, no. 1 (January 2006): 8–12. doi:10.1016/j.crme.2005.11.002.

Waterfall, Paul, Nick McCormick, and Alan Owens. “Optical Imaging for Low-Cost Structural Measurements.” *Proceedings of the ICE - Bridge Engineering* 167, no. 1 (March 1, 2014): 33–42. doi:10.1680/bren.11.00055.

Tung, S.H., Kuo J.C. and Shih, M.S. “Strain distribution analysis using digital image correlation techniques.” The Eighteenth KKCNN Symposium on Civil Engineering-NTU29 (December 19,2005)

Hild, F., & Roux, S. (2006). Digital Image Correlation: from Displacement Measurement to Identification of Elastic Properties – a Review. *Strain*, *42*(2), 69–80. https://doi.org/10.1111/j.1475-1305.2006.00258.x

Løkberg, O. J. (1987). Electronic Speckle Pattern Interferometry. In O. D. D. Soares (Ed.), *Optical Metrology* (pp. 542–572). Springer Netherlands. Retrieved from http://link.springer.com/chapter/10.1007/978-94-009-3609-6_36

Single Camera Calibration App - MATLAB & Simulink. (n.d.). Retrieved July 24, 2016, from http://www.mathworks.com/help/vision/ug/single-camera-calibrator-app.html

J. Lord and N. McCormick, “Digital image correlation for structural measurements,” *Proc. ICE - Civ. Eng.*, vol. 165, no. 4, pp. 185–190, Nov. 2012.

H. Haddadi and S. Belhabib, “Use of rigid-body motion for the investigation and estimation of the measurement errors related to digital image correlation technique,” *Opt. Lasers Eng.*, vol. 46, no. 2, pp. 185–196, Feb. 2008.

C. Murray, W. A. Take, N. A. Hoult, and A. Hoag, “Field monitoring of a bridge using digital image correlation,” *Proc. ICE - Bridge Eng.*, vol. 168, no. 1, pp. 3–12, Mar. 2015.

M. J. . Feron, “Deformation analysis using image processing.”, *Mechanics of Electronic Textiles* (May, 2008)

P. Bergeron, “Parallel Lasers for Remote Measurements of Morphological Traits,” *J. Wildl. Manag.*, vol. 71, no. 1, pp. 289–292, Feb. 2007.

B. Pan and K. Li, “A fast digital image correlation method for deformation measurement,” *Opt. Lasers Eng.*, vol. 49, no. 7, pp. 841–847, Jul. 2011.

“Imetrum: non-contact precision measurement.” [Online]. Available: http://www.imetrum.com/. [Accessed: 23-Mar-2017].

Ali, Aziah, Rajasvaran Logeswaran, and Michel RM Bister. "Optimal camera calibration for 3 D object tracking." *WSEAS Transactions on Computers* 3, no. 1 (2004): 262-266

A. Diluxshan and H.M.Y.C. Mallikarachchi, “A Low Cost Optics Based Displacement Measuring Technique,” presented at the *Society of Structural Engineers Annual Sessions 2016* (SSEAS 2016), Sep. 2016.

A. Diluxshan and H.M.Y.C. Mallikarachchi, “Image Based Measuring Technique for In-plane Loading”, presented at the *Internation Conference on Substainable Built Environment 2016 (ICSBE* 2016), Sep. 2016.

BIBILIOGRAPHY

Hild, François, and Stéphane Roux. "Measuring Stress Intensity Factors with a Camera: Integrated Digital Image Correlation (I-DIC)." *Comptes Rendus Mécanique* 334, no. 1 (January 2006): 8–12. doi:10.1016/j.crme.2005.11.002.

Lord, Jerry, and Nick McCormick. "Digital Image Correlation for Structural Measurements." *Proceedings of the ICE - Civil Engineering* 165, no. 4 (November 1, 2012): 185–90. doi:10.1680/cien.11.00040.

Waterfall, Paul, Nick McCormick, and Alan Owens. "Optical Imaging for Low-Cost Structural Measurements." *Proceedings of the ICE - Bridge Engineering* 167, no. 1 (March 1, 2014): 33–42. doi:10.1680/bren.11.00055.

Feron, M.J.M. "Deformation analysis using image processing." *Mechanics of Electronic Textiles* (May, 2008)

Tung, S.H., Kuo J.C. and Shih, M.S. "Strain distribution analysis using digital image correlation techniques." The Eighteenth KKCNN Symposium on Civil Engineering-NTU29 (December 19,2005)

Hild, F., & Roux, S. (2006). Digital Image Correlation: from Displacement Measurement to Identification of Elastic Properties – a Review. *Strain*, *42*(2), 69–80. https://doi.org/10.1111/j.1475-1305.2006.00258.x

Løkberg, O. J. (1987). Electronic Speckle Pattern Interferometry. In O. D. D. Soares (Ed.), *Optical Metrology* (pp. 542–572). Springer Netherlands. Retrieved from http://link.springer.com/chapter/10.1007/978-94-009-3609-6_36

Single Camera Calibration App - MATLAB & Simulink. (n.d.). Retrieved July 24, 2016, from http://www.mathworks.com/help/vision/ug/single-camera-calibrator-app.html

Helfrick, Mark N., Christopher Niezrecki, Peter Avitabile, and Timothy Schmidt. "3D Digital Image Correlation Methods for Full-Field Vibration Measurement." *Mechanical Systems and Signal Processing* 25, no. 3 (April 2011): 917–27. doi:10.1016/j.ymssp.2010.08.013.

Pan, Bing, and Kai Li. "A Fast Digital Image Correlation Method for Deformation Measurement." *Optics and Lasers in Engineering* 49, no. 7 (July 2011): 841–47. doi:10.1016/j.optlaseng.2011.02.023.

Tong, W. "An Evaluation of Digital Image Correlation Criteria for Strain Mapping Applications." *Strain* 41, no. 4 (November 1, 2005): 167–75. doi:10.1111/j.1475-1305.2005.00227.x.

Zhang, Zhi-Feng, Yi-Lan Kang, Huai-Wen Wang, Qing-Hua Qin, Yu Qiu, and Xiao-Qi Li. "A Novel Coarse-Fine Search Scheme for Digital Image Correlation Method." *Measurement* 39, no. 8 (October 2006): 710–18. doi:10.1016/j.measurement.2006.03.008.

Kamaya, Masayuki, and Masahiro Kawakubo. "A Procedure for Determining the True Stress–strain Curve over a Large Range of Strains Using Digital Image Correlation and Finite Element Analysis." *Mechanics of Materials* 43, no. 5 (May 2011): 243–53. doi:10.1016/j.mechmat.2011.02.007.

Périé, Jean Noël, Hugo Leclerc, Stéphane Roux, and François Hild. "Digital Image Correlation and Biaxial Test on Composite Material for Anisotropic Damage Law Identification." *International Journal of Solids and Structures* 46, no. 11–12 (June 1, 2009): 2388–96. doi:10.1016/j.ijsolstr.2009.01.025.

Moerman, Kevin M., Cathy A. Holt, Sam L. Evans, and Ciaran K. Simms. "Digital Image Correlation and Finite Element Modelling as a Method to Determine Mechanical Properties of Human Soft Tissue in Vivo." *Journal of Biomechanics* 42, no. 8 (May 29, 2009): 1150–53. doi:10.1016/j.jbiomech.2009.02.016.

Pan, Bing, Anand Asundi, Huimin Xie, and Jianxin Gao. "Digital Image Correlation Using Iterative Least Squares and Pointwise Least Squares for Displacement Field and Strain Field Measurements." *Optics and Lasers in Engineering* 47, no. 7–8 (July 2009): 865–74. doi:10.1016/j.optlaseng.2008.10.014.

Murray, Chris, W. Andy Take, Neil A. Hoult, and Adam Hoag. "Field Monitoring of a Bridge Using Digital Image Correlation." *Proceedings of the ICE - Bridge Engineering* 168, no. 1 (March 1, 2015): 3–12. doi:10.1680/bren.13.00024.

Pan, Bing, Dafang Wu, and Yong Xia. "High-Temperature Deformation Field Measurement by Combining Transient Aerodynamic Heating Simulation System and Reliability-Guided Digital Image Correlation." *Optics and Lasers in Engineering* 48, no. 9 (September 2010): 841–48. doi:10.1016/j.optlaseng.2010.04.007.

Abanto-Bueno, Jorge, and John Lambros. "Investigation of Crack Growth in Functionally Graded Materials Using Digital Image Correlation." *Engineering Fracture Mechanics* 69, no. 14–16 (September 2002): 1695–1711. doi:10.1016/S0013-7944(02)00058-9.

V. Richter-Trummer, P.M.G.P. Moreira, S.D. Pastrama, M.A.P. Vaz, and P.M.S.T. de Castro. "Methodology for in Situ Stress Intensity Factor Determination on Cracked Structures by Digital Image Correlation." *International Journal of Structural Integrity* 1, no. 4 (April 1, 2010): 344–57. doi:10.1108/17579861011099178.

J. Zhang, M. Li, C.Y. Xiong, J. Fang, and S. Yi. "Thermal Deformation Analysis of BGA Package by Digital Image Correlation Technique." *Microelectronics International* 22, no. 1 (April 1, 2005): 34–42. doi:10.1108/13565360510575530.

Berryman, James G., and Stephen C. Blair. "Use of Digital Image Analysis to Estimate Fluid Permeability of Porous Materials: Application of Two-point Correlation Functions." *Journal of Applied Physics* 60, no. 6 (September 15, 1986): 1930–38. doi:10.1063/1.337245.

A. Diluxshan and H.M.Y.C. Mallikarachchi, "A Low Cost Optics Based Displacement Measuring Technique," presented at the *Society of Structural Engineers Annual Sessions 2016* (SSEAS 2016), Sep. 2016.

A. Diluxshan and H.M.Y.C. Mallikarachchi, "Image Based Measuring Technique for In-plane Loading", presented at the *Internation Conference on Substainable Built Environment 2016 (ICSBE* 2016), Sep. 2016.

H. Haddadi and S. Belhabib, "Use of rigid-body motion for the investigation and estimation of the measurement errors related to digital image correlation technique," *Opt. Lasers Eng.*, vol. 46, no. 2, pp. 185–196, Feb. 2008.

C. Murray, W. A. Take, N. A. Hoult, and A. Hoag, "Field monitoring of a bridge using digital image correlation," *Proc. ICE - Bridge Eng.*, vol. 168, no. 1, pp. 3–12, Mar. 2015.

P. Bergeron, "Parallel Lasers for Remote Measurements of Morphological Traits," *J. Wildl. Manag.*, vol. 71, no. 1, pp. 289–292, Feb. 2007.

Ali, Aziah, Rajasvaran Logeswaran, and Michel RM Bister. "Optimal camera calibration for 3 D object tracking." *WSEAS Transactions on Computers* 3, no. 1 (2004): 262-266

APPENDICES

Appendix A: Equipment Used in the Experiment

A.1 Camera:

Type:		Sony W800 compact camera
Sensor:		
	Sensor type	0.31 in type Super HAD CCD
	No. of pixels	20.1MP
Lens:		
	F-No.	F3.2 (W) – 6.4 (T)
	Focal length	f=4.6-23mm
	Focus range	1.97 in – infinity (W), 1.97 ft – infinity (T)
Camera:		
	Exposure compensation	+/- 2.0 EV (1/3 EV steps)
	ISO sensitivity	ISO 100-3200
	Zoom	5x (2.6x used)
	External flash mode	Auto (Off during experiment)
	Shutter speed	Auto (1 – 1/1500)

A.2 Apparatus:

Hounsfield Tensometer:

Elongation per revolution	0.0085mm

Vernier Caliper:

Accuracy	0.02mm

Specimen:

Dog bone specimen with following dimensions,

Thickness	3mm
Width	38mm
Gauge length	50.8mm

Parallel length	63.5mm

A.3 Universal Testing Machine

Model	WAW-1000E
Max. Load(kN)	1000
Load measuring range	2~100%F.S. (0.2/0.4~100%F.S optional)
Load accuracy (%)	±1
Deformation measuring range	2~100%F.S. (0.2/0.4~100%F.S optional)
Deformation accuracy (%)	±1
Displacement position(mm)	0.001
Test loading speed(mm/min)	0.5-50 (0.01-50 if configured with EDC220 controller & Moog servo valve)
Max. Crosshead moving speed (mm/min)	200
Stress control range	1~60(N/mm2)S-1
Strain control range	0.00025/s~0.0025/s
Tensile space1(mm)	750
Compression space(mm)	620
Piston stroke(mm)	250
Column Distance(mm)	570
Column Diameter(mm)	90
Working table size(mm)	650x800
Flat jaw (mm)	0-40
Round jaw(mm)	Φ20-Φ60
Jaw length(mm)	110
Jaw width(mm)	110
Platen size(mm)	Φ148x40

Bending span(mm)	50-500
Roller diameter (mm)	Φ50
Roller length (mm)	160
Bending depth (mm)	180
Net weight (kg)	3500
Max. height(mm)	2750
Dimension of load frame(mm)	900X650X2500
Size of power pack(mm)	550x550x1410
Oil tank volume(L)	110
Oil pressure (MPa)	26
Footprint (L x W)	1600x1600
Gross weight (kg)	3700
Shipping dimension (mm)	2700x1160x1100 1540x980x1725
Power supply	3PH, 380VAC, 50H, 5Kw

A.4 Extensometer

The strain gauge resistance:	350 ohms
Bridge voltage value: acuities	6V (dc, ac all can)
Output sensitivity:	about 2mV/V
Extensometer gauge length:	YYU series20~200 mm, YYJ series 5~25 mm
Maximum deformation:	YYU series25 mm, YYJ Series 4mm
Output terminal connector:	four core or five core plugs, etc

A.5 Camera EOS 700D

IMAGE SENSOR

Type	22.3 x 14.9mm CMOS

Effective Pixels	Approx. 18.0 megapixels
Total Pixels	Approx. 18.5 megapixels
Aspect Ratio	3:2
Low-Pass Filter	Built-in/Fixed
Sensor Cleaning	EOS integrated cleaning system
Colour Filter Type	Primary Colour

LENS

Lens Mount	EF/EF-S
Focal Length	Equivalent to 1.6x the focal length of the lens

FOCUSING

Type	TTL-CT-SIR with a CMOS sensor
AF System/ Points	9 cross-type AF points (f/2.8 at centre)
AF Working Range	EV -0.5 -18 (at 23°C & ISO100)
AF Modes	AI Focus One Shot AI Servo
AF Point Selection	Automatic selection, Manual selection
Selected AF Point Display	Superimposed in viewfinder and indicated on LCD monitor
Predictive AF	Yes, up to 10m[1]
AF Lock	Locked when shutter button is pressed half way in One Shot AF mode.
AF Assist Beam	Intermittent firing of built-in flash or emitted by optional dedicated Speedlite
Manual Focus	Selected on lens

SHUTTER

Type	Electronically-controlled focal-plane shutter
Speed	30-1/4000 sec (1/2 or 1/3 stop increments), Bulb

(Total shutter speed range. Available range varies by shooting mode)

VIEWFINDER

Type	Pentamirror
Coverage (Vertical/Horizontal)	Approx. 95%
Magnification	Approx. 0.85x (4)
Eyepoint	Approx. 19mm (from eyepiece lens centre)
Dioptre Correction	-3 to +1 m-1 (dioptre)
Focusing Screen	Fixed
Mirror	Quick-return half mirror (Transmission: reflection ratio of 40:60, no mirror cut-off with EF600mm f/4 or shorter)
Depth of Field Preview	Yes, with Depth of Field preview button.
Eyepiece Shutter	On strap

OTHER FEATURES

Custom Functions

Appendix B: MATLAB Algorithm

B.1 Calibration Algorithm

```
%Camera Calibration
%--------------------

% Define calibration images to process
imageFileNames = {'F:\20160702_205334.jpg',...
'F:\20160702_205356.jpg',...
'F:\20160702_205404.jpg',...
'F:\20160702_205410.jpg',...
'F:\20160702_205417.jpg',...
'F:\20160702_205423.jpg',...
'F:\20160702_205427.jpg',...
'F:\20160702_205433.jpg',...
'F:\20160702_205437.jpg',...
'F:\20160702_205442.jpg',...
'F:\20160702_205446.jpg',...
'F:\20160702_205449.jpg',...
'F:\20160702_205504.jpg',...
'F:\20160702_205510.jpg',...
'F:\20160702_205520.jpg',...
'F:\20160702_205547.jpg',...
'F:\20160702_205623.jpg',...
'F:\20160702_205654.jpg',...
'F:\20160702_205738.jpg',...
'F:\20160702_205743.jpg',...
};

% Detect checkerboards in images
[imagePoints, boardSize, imagesUsed] =
detectCheckerboardPoints(imageFileNames);
imageFileNames = imageFileNames(imagesUsed);

% Generate world coordinates of the corners of the squares
squareSize = 25;  %calibration 1 square size in units of 'mm'
worldPoints = generateCheckerboardPoints(boardSize, squareSize);

% Calibrate the camera
[cameraParams, imagesUsed, estimationErrors] =
estimateCameraParameters(imagePoints, worldPoints, ...
'EstimateSkew', true, 'EstimateTangentialDistortion', false, ...
'NumRadialDistortionCoefficients', 2, 'WorldUnits', 'mm');

% View reprojection errors
h1=figure; showReprojectionErrors(cameraParams, 'BarGraph');

% Visualize pattern locations
h2=figure; showExtrinsics(cameraParams, 'PatternCentric');

% Display parameter estimation errors
displayErrors(estimationErrors, cameraParams);

% For example, we can use the calibration data to remove effects of
lens distortion.
originalImage = imread(imageFileNames{1});
%undistortedImage = undistortImage(originalImage, cameraParams);
```

```
[undistortedImage,newOrigin]=undistortImage(originalImage,
cameraParams);
imshow(undistortedImage);

% See additional examples of how to use the calibration data.  At
the prompt type:
% showdemo('MeasuringPlanarObjectsExample')
% showdemo('SparseReconstructionExample')
```

B.2 Output of the above (Appendix B.1) Algorithm:

Standard Errors of Estimated Camera Parameters

Intrinsics

Focal length (pixels): [2414.6298 +/- 8.5917 2435.9077 +/- 8.5264]
Principal point (pixels):[1294.2716 +/- 2.0554 909.5780 +/- 2.6184]
Skew: [-6.1185 +/- 0.5285]
Radial distortion: [0.1609 +/- 0.0079 -0.7222 +/- 0.0607]

Extrinsics

Rotation vectors:

[	0.5757 +/- 0.0022	0.1639 +/- 0.0026	3.0168 +/- 0.0006	]
[	0.0254 +/- 0.0041	-0.0163 +/- 0.0057	3.1235 +/- 0.0003	]
[	0.2953 +/- 0.0026	-0.2413 +/- 0.0030	-3.1148 +/- 0.0004	]
[	-0.3264 +/- 0.0030	-0.1310 +/- 0.0035	3.0844 +/- 0.0004	]
[	0.0189 +/- 0.0035	-0.3056 +/- 0.0040	-3.0788 +/- 0.0005	]
[	-0.0051 +/- 0.0036	0.0349 +/- 0.0050	-3.1282 +/- 0.0003	]
[	-0.1588 +/- 0.0027	-0.3494 +/- 0.0030	-3.0924 +/- 0.0005	]
[	-0.1368 +/- 0.0025	-0.1282 +/- 0.0030	2.9796 +/- 0.0003	]
[	0.3424 +/- 0.0023	-0.2966 +/- 0.0025	-3.0627 +/- 0.0004	]
[	0.0354 +/- 0.0026	-0.0192 +/- 0.0037	-3.1404 +/- 0.0003	]
[	0.0018 +/- 0.0067	0.0966 +/- 0.0093	3.1110 +/- 0.0005	]
[	0.0953 +/- 0.0035	0.1436 +/- 0.0039	3.0006 +/- 0.0004	]
[	-0.0876 +/- 0.0023	-0.3696 +/- 0.0026	3.0722 +/- 0.0004	]
[	0.0146 +/- 0.0024	-0.5755 +/- 0.0027	-3.0821 +/- 0.0006	]
[	-0.5878 +/- 0.0023	0.0969 +/- 0.0026	3.0785 +/- 0.0006	]
[	0.2899 +/- 0.0035	0.0142 +/- 0.0041	3.0834 +/- 0.0005	]
[	0.1053 +/- 0.0027	-0.2824 +/- 0.0032	3.0814 +/- 0.0005	]
[	0.0910 +/- 0.0023	0.2455 +/- 0.0026	3.0053 +/- 0.0004	]
[	0.5013 +/- 0.0019	0.4959 +/- 0.0022	3.0515 +/- 0.0006	]
[	-0.4972 +/- 0.0028	0.1027 +/- 0.0031	3.0984 +/- 0.0005	]

Translation vectors (mm):

[	38.4922 +/- 0.4421	66.3069 +/- 0.5466	506.1958 +/- 1.8718	]
[	75.3411 +/- 0.4455	56.9876 +/- 0.5623	522.0011 +/- 1.8702	]
[	119.9770 +/- 0.4152	44.0626 +/- 0.5397	500.8516 +/- 1.7495	]
[	93.8129 +/- 0.4505	101.1754 +/- 0.5727	539.3369 +/- 1.8309	]
[	114.6758 +/- 0.4777	41.6797 +/- 0.6051	556.8783 +/- 2.0105	]
[	88.7685 +/- 0.4210	90.9182 +/- 0.5316	494.2845 +/- 1.7489	]

[	53.9191 +/- 0.4205	40.9917 +/- 0.5278	484.8456 +/- 1.7683	]
[	145.0586 +/- 0.3816	66.6888 +/- 0.4878	460.7869 +/- 1.6038	]
[	93.4762 +/- 0.3761	39.9689 +/- 0.4875	451.8492 +/- 1.5621	]
[	80.4392 +/- 0.3482	68.5073 +/- 0.4423	410.0927 +/- 1.4529	]
[	84.1385 +/- 0.5727	69.7984 +/- 0.7252	667.0181 +/- 2.4494	]
[	36.4667 +/- 0.4912	0.3751 +/- 0.6220	573.1633 +/- 2.0598	]
[	89.2566 +/- 0.4146	106.1776 +/- 0.5230	495.4692 +/- 1.6913	]
[	100.0227 +/- 0.4139	50.6485 +/- 0.5254	483.7036 +/- 1.7705	]
[	79.8624 +/- 0.3872	64.2443 +/- 0.4888	460.6280 +/- 1.4878	]
[	64.8680 +/- 0.4756	65.0443 +/- 0.5904	547.8949 +/- 1.9632	]
[	93.5285 +/- 0.4042	80.1241 +/- 0.5084	479.1034 +/- 1.6615	]
[	76.9006 +/- 0.3785	-11.5440 +/- 0.4778	440.2027 +/- 1.5679	]
[	74.0567 +/- 0.3537	42.4776 +/- 0.4334	401.2955 +/- 1.5124	]
[	77.2469 +/- 0.4089	123.1866 +/- 0.5167	484.6958 +/- 1.6176	]

B.3 Finding Out the Centre of the Circles

```
%Finding out the centre of circles with calibration
%---------------------------------------------------

clc;
clearvars; % Get rid of variables from prior run of this m-file.
imtool close all;  % Close all imtool figures.
fprintf('Running the programme...\n'); % Message sent to command window.
workspace; % Make sure the workspace panel with all the variables is showing.
imtool close all;  % Close all imtool figures.
format long g;
format compact;
captionFontSize = 14;


% Define images to process
imageFileNames = {'F:\20160702_205334.jpg',...
'F:\20160702_205356.jpg',...
'F:\20160702_205404.jpg',...
'F:\20160702_205410.jpg',...
'F:\20160702_205417.jpg',...
'F:\20160702_205423.jpg',...
'F:\20160702_205427.jpg',...
'F:\20160702_205433.jpg',...
'F:\20160702_205437.jpg',...
'F:\20160702_205442.jpg',...
'F:\20160702_205446.jpg',...
'F:\20160702_205449.jpg',...
'F:\20160702_205504.jpg',...
'F:\20160702_205510.jpg',...
'F:\20160702_205520.jpg',...
'F:\20160702_205547.jpg',...
'F:\20160702_205623.jpg',...
'F:\20160702_205654.jpg',...
'F:\20160702_205738.jpg',...
'F:\20160702_205743.jpg',...
};

% Detect checkerboards in images
[imagePoints, boardSize, imagesUsed] = detectCheckerboardPoints(imageFileNames);
imageFileNames = imageFileNames(imagesUsed);

% Generate world coordinates of the corners of the squares
squareSize = 25;  % in units of 'mm'
worldPoints = generateCheckerboardPoints(boardSize, squareSize);

% Calibrate the camera
[cameraParams, imagesUsed, estimationErrors] = estimateCameraParameters(imagePoints, worldPoints, ...
'EstimateSkew', true, 'EstimateTangentialDistortion', false, ...
'NumRadialDistortionCoefficients', 2, 'WorldUnits', 'mm');

% View reprojection errors
%%h1=figure; showReprojectionErrors(cameraParams, 'BarGraph');
```

```
% Visualize pattern locations
h2=figure; showExtrinsics(cameraParams, 'PatternCentric');

% Display parameter estimation errors
displayErrors(estimationErrors, cameraParams);

% For example, we can use the calibration data to remove effects of lens distortion.
%originalImage = imread(imageFileNames{17});
%undistortedImage = undistortImage(originalImage, cameraParams);
imOrig=imread('F:\20160702_205701.jpg');
%imshow(undistortedImage);

% Display one of the calibration images
magnification = 25;

subplot(3,3,1);
imshow(imOrig, 'InitialMagnification', magnification);
title('Input Image');

[im,newOrigin]=undistortImage(imOrig, cameraParams);
subplot(3,3,2);
imshow(im);
title('Undistorted Image');

im=rgb2gray(im);
subplot(3,3,3);
imshow(im, 'InitialMagnification', magnification);
title('GreyScale Image');

%%binaryImage = (im>50 & im<170);
%%imCoin = imfill(binaryImage, 'holes');
%%labeledImage = bwlabel(binaryImage, 8);

%%subplot(3,3,4);
%%imshow(imCoin, 'InitialMagnification', magnification);
%%title('Segmented Coins');

%Histogram
%[pixelCount, grayLevels] = imhist(im);
%subplot(3, 3, 4);
%bar(pixelCount);
%title('Histogram of original image');
%xlim([0 grayLevels(end)]);
%grid on;

%thresholdValue = 100;
binaryImage =  im<50; % Bright objects will be chosen if you use >.

binaryImage = imfill(binaryImage, 'holes');
%hold on;
%maxYValue = ylim;
%line([thresholdValue, thresholdValue], maxYValue, 'Color', 'r');
% Place a text label on the bar chart showing the threshold.
%annotationText = sprintf('Thresholded at %d gray levels', thresholdValue);
% For text(), the x and y need to be of the data class "double" so let's cast both to double.
%text(double(thresholdValue + 5), double(0.5 * maxYValue(2)), annotationText, 'FontSize', 10, 'Color', [0 .5 0]);
```

```
%text(double(thresholdValue - 70), double(0.94 * maxYValue(2)), 'Background', 'FontSize', 10, 'Color', [0 0 .5]);
%text(double(thresholdValue + 50), double(0.94 * maxYValue(2)), 'Foreground', 'FontSize', 10, 'Color', [0 0 .5]);

% Display the binary image.
subplot(3, 3, 4);
imshow(binaryImage);
title('Binary Image, obtained by thresholding');

%----------------------------------------------------------------------------------------------------------------------
labeledImage = bwlabel(binaryImage, 8);     % Label each blob so we can make measurements of it
%labeledImage is an integer-valued image where all pixels in the blobs have values of 1, or 2, or 3, or ... etc.
%subplot(3, 3, 5);
%imshow(labeledImage, []);  % Show the gray scale image.
%title('Labeled Image, from bwlabel()');

% Let's assign each blob a different color to visually show the user the distinct blobs.
coloredLabels = label2rgb (labeledImage, 'hsv', 'k', 'shuffle'); % pseudo random color labels
% coloredLabels is an RGB image.  We could have applied a colormap instead (but only with R2014b and later)
subplot(3, 3, 5);
imshow(coloredLabels);
axis image; % Make sure image is not artificially stretched because of screen's aspect ratio.
caption = sprintf('Pseudo colored labels');
title(caption);

% Get all the blob properties.  Can only pass in originalImage in version R2008a and later.
blobMeasurements = regionprops(labeledImage, im, 'all');
numberOfBlobs = size(blobMeasurements, 1);

%----------------------------------------------------------------------------------------------------------------------


% bwboundaries() returns a cell array, where each cell contains the row/column coordinates for an object in the image.
% Plot the borders of all the coins on the original grayscale image using the coordinates returned by bwboundaries.
subplot(3, 3, 6);
imshow(im);
title('Outlines, from bwboundaries()', 'FontSize', captionFontSize);
axis image; % Make sure image is not artificially stretched because of screen's aspect ratio.
hold on;
boundaries = bwboundaries(binaryImage);
numberOfBoundaries = size(boundaries, 1);
for k = 1 : numberOfBoundaries
thisBoundary = boundaries{k};
plot(thisBoundary(:,2), thisBoundary(:,1), 'g', 'LineWidth', 2);
end
hold off;
```

```matlab
textFontSize = 14;  % Used to control size of "blob number" labels put atop the image.
labelShiftX = -7;   % Used to align the labels in the centers of the coins.
blobECD = zeros(1, numberOfBlobs);
% Print header line in the command window.
fprintf(1,'Blob #      Mean Intensity  Area   Perimeter    Centroid Diameter\n');
% Loop over all blobs printing their measurements to the command window.
i=0;
for k = 1 : numberOfBlobs           % Loop through all blobs.
% Find the mean of each blob.  (R2008a has a better way where you can pass the original image
% directly into regionprops.  The way below works for all versions including earlier versions.)
thisBlobsPixels = blobMeasurements(k).PixelIdxList;  % Get list of pixels in current blob.
meanGL = mean(im(thisBlobsPixels)); % Find mean intensity (in original image!)
meanGL2008a = blobMeasurements(k).MeanIntensity; % Mean again, but only for version >= R2008a

blobArea = blobMeasurements(k).Area;        % Get area.
blobPerimeter = blobMeasurements(k).Perimeter;      % Get perimeter.
blobCentroid = blobMeasurements(k).Centroid;        % Get centroid one at a time
blobECD(k) = sqrt(4 * blobArea / pi);                   % Compute ECD - Equivalent Circular Diameter.
if(blobArea<30000 & blobArea>18000 & blobPerimeter<30000)
i=i+1;
fprintf(1,'#%2d %17.1f %11.1f %8.1f %8.1f %8.1f % 8.1f\n', k, meanGL, blobArea, blobPerimeter, blobCentroid, blobECD(k));
% Put the "blob number" labels on the "boundaries" grayscale image.
text(blobCentroid(1) + labelShiftX, blobCentroid(2), num2str(k), 'FontSize', textFontSize, 'FontWeight', 'Bold');
blobMeasurements(i)=blobMeasurements(k);
end
end

allBlobCentroids = [blobMeasurements.Centroid];
centroidsX = allBlobCentroids(1:2:end-1);
centroidsY = allBlobCentroids(2:2:end);

x=[centroidsX;centroidsY]; %combining centroid matrix as Mx2 matrix
combinedCentroids=transpose(x);

%Distance in between blobs-------------------------------------
for j=1:i-1
fprintf('Distance in between blobs ');
deltax=(centroidsX(j)-centroidsX(j+1));
deltay=(centroidsY(j)-centroidsY(j+1));
distance=hypot(deltax,deltay);
fprintf('%d & %d is %f pixels\n',j,j+1,distance);
end

%--------------------------------------------------------------

% Detect the checkerboard.
%corimage=imread(imageFileNames{17});
```

```
[170] [imagePoints, boardSize] =
      detectCheckerboardPoints(imageFileNames{18});

[171] % Compute rotation and translation of the camera.
[172] [R, t] = extrinsics(imagePoints, worldPoints, cameraParams);

[173] % Get the world coordinates of the corners
[174] worldPoints2 = pointsToWorld(cameraParams, R, t,combinedCentroids);
```

B.4 Timber Deflection Measurement

```
%Analyse the digital camera (fixed on tripod) images experiment 2

clc;
clearvars; % Get rid of variables from prior run of this m-file.
imtool close all;  % Close all imtool figures.
fprintf('Running the programme...\n'); % Message sent to command window.
workspace; % Make sure the workspace panel with all the variables is showing.
imtool close all;  % Close all imtool figures.
format long g;
format compact;
captionFontSize = 14;
%-------------------------------------------------------


% Define images to processa
%Total Images = 35
% Define images to process
imageFileNames = {'F:\DSC03297.JPG',...
'F:\DSC03299.JPG',...
'F:\DSC03300.JPG',...
'F:\DSC03302.JPG',...
'F:\DSC03305.JPG',...
'F:\DSC03306.JPG',...
'F:\DSC03307.JPG',...
'F:\DSC03308.JPG',...
'F:\DSC03311.JPG',...
'F:\DSC03312.JPG',...
'F:\DSC03313.JPG',...
'F:\DSC03316.JPG',...
'F:\DSC03317.JPG',...
'F:\DSC03318.JPG',...
'F:\DSC03320.JPG',...
'F:\DSC03322.JPG',...
'F:\DSC03325.JPG',...
'F:\DSC03326.JPG',...
'F:\DSC03327.JPG',...
'F:\DSC03328.JPG',...
'F:\DSC03335.JPG',...
};

%Image 26 is compared for scaling


% Detect checkerboards in images
[imagePoints, boardSize, imagesUsed] = detectCheckerboardPoints(imageFileNames);
imageFileNames = imageFileNames(imagesUsed);

% Generate world coordinates of the corners of the squares
squareSize = 2.17943e+01;  % in units of 'mm'
worldPoints = generateCheckerboardPoints(boardSize, squareSize);

% Calibrate the camera
[cameraParams, imagesUsed, estimationErrors] = estimateCameraParameters(imagePoints, worldPoints, ...
```

```
'EstimateSkew', true, 'EstimateTangentialDistortion', true, ...
'NumRadialDistortionCoefficients', 2, 'WorldUnits', 'mm');

% View reprojection errors
h1=figure; showReprojectionErrors(cameraParams, 'BarGraph');

% Visualize pattern locations
%h2=figure; showExtrinsics(cameraParams, 'CameraCentric');

% Display parameter estimation errors
%displayErrors(estimationErrors, cameraParams);

% For example, you can use the calibration data to remove effects of lens distortion.
%originalImage = imread(imageFileNames{17});
%undistortedImage = undistortImage(originalImage, cameraParams);
imOrig=imread('F:\DSC03371.JPG');
%imshow(undistortedImage);

% Display one of the calibration images
magnification = 25;

subplot(3,3,1);
imshow(imOrig, 'InitialMagnification', magnification);
title('Input Image');

[im,newOrigin]=undistortImage(imOrig, cameraParams);
subplot(3,3,2);
imshow(im);
title('Undistorted Image');

im=rgb2gray(im); %earlier it was not 'imOrig' but 'im'
subplot(3,3,3);
imshow(im, 'InitialMagnification', magnification);
title('GreyScale Image');

%%binaryImage = (im>50 & im<170);
%%imCoin = imfill(binaryImage, 'holes');
%%labeledImage = bwlabel(binaryImage, 8);

%%subplot(3,3,4);
%%imshow(imCoin, 'InitialMagnification', magnification);
%%title('Segmented Coins');

%Histogram
%[pixelCount, grayLevels] = imhist(im);
%subplot(3, 3, 4);
%bar(pixelCount);
%title('Histogram of original image');
%xlim([0 grayLevels(end)]);
%grid on;

%thresholdValue = 100;
binaryImage =  im>130 & im<200; % Bright objects will be chosen if you use >.

binaryImage = imfill(binaryImage, 'holes');
%hold on;
%maxYValue = ylim;
%line([thresholdValue, thresholdValue], maxYValue, 'Color', 'r');
% Place a text label on the bar chart showing the threshold.
```

```
%annotationText = sprintf('Thresholded at %d gray levels', thresholdValue);
% For text(), the x and y need to be of the data class "double" so let's cast both to double.
%text(double(thresholdValue + 5), double(0.5 * maxYValue(2)), annotationText, 'FontSize', 10, 'Color', [0 .5 0]);
%text(double(thresholdValue - 70), double(0.94 * maxYValue(2)), 'Background', 'FontSize', 10, 'Color', [0 0 .5]);
%text(double(thresholdValue + 50), double(0.94 * maxYValue(2)), 'Foreground', 'FontSize', 10, 'Color', [0 0 .5]);

% Display the binary image.
subplot(3, 3, 4);
imshow(binaryImage);
title('Binary Image, obtained by thresholding');

%-------------------------------------------------------------------------------------------------------------------
labeledImage = bwlabel(binaryImage, 8);     % Label each blob so we can make measurements of it
%labeledImage is an integer-valued image where all pixels in the blobs have values of 1, or 2, or 3, or ... etc.
%subplot(3, 3, 5);
%imshow(labeledImage, []);  % Show the gray scale image.
%title('Labeled Image, from bwlabel()');

% Let's assign each blob a different color to visually show the user the distinct blobs.
coloredLabels = label2rgb (labeledImage, 'hsv', 'k', 'shuffle'); % pseudo random color labels
% coloredLabels is an RGB image.  We could have applied a colormap instead (but only with R2014b and later)
subplot(3, 3, 5);
imshow(coloredLabels);
axis image; % Make sure image is not artificially stretched because of screen's aspect ratio.
caption = sprintf('Pseudo colored labels');
title(caption);

% Get all the blob properties.  Can only pass in originalImage in version R2008a and later.
blobMeasurements = regionprops(labeledImage, im, 'all');
numberOfBlobs = size(blobMeasurements, 1);

%-------------------------------------------------------------------------------------------------------------------


% bwboundaries() returns a cell array, where each cell contains the row/column coordinates for an object in the image.
% Plot the borders of all the coins on the original grayscale image using the coordinates returned by bwboundaries.
subplot(3, 3, 6);
figure;imshow(im);
title('Outlines, from bwboundaries()', 'FontSize', captionFontSize);
axis image; % Make sure image is not artificially stretched because of screen's aspect ratio.
hold on;
boundaries = bwboundaries(binaryImage);
numberOfBoundaries = size(boundaries, 1);
for k = 1 : numberOfBoundaries
```

```
thisBoundary = boundaries{k};
plot(thisBoundary(:,2), thisBoundary(:,1), 'g', 'LineWidth', 2);
end
hold off;

textFontSize = 20;  % Used to control size of "blob number" labels put atop the image.
labelShiftX = -7;   % Used to align the labels in the centers of the coins.
blobECD = zeros(1, numberOfBlobs);
% Print header line in the command window.
fprintf(1,'Blob #      Mean Intensity  Area   Perimeter    Centroid Diameter\n');
% Loop over all blobs printing their measurements to the command window.
i=0;
for k = 1 : numberOfBlobs           % Loop through all blobs.
% Find the mean of each blob.  (R2008a has a better way where you can pass the original image
% directly into regionprops.  The way below works for all versions including earlier versions.)
thisBlobsPixels = blobMeasurements(k).PixelIdxList;  % Get list of pixels in current blob.
meanGL = mean(im(thisBlobsPixels)); % Find mean intensity (in original image!)
meanGL2008a = blobMeasurements(k).MeanIntensity; % Mean again, but only for version >= R2008a

blobArea = blobMeasurements(k).Area;        % Get area.
blobPerimeter = blobMeasurements(k).Perimeter;      % Get perimeter.
blobCentroid = blobMeasurements(k).Centroid;% Get centroid one at a time
blobECD(k) = sqrt(4 * blobArea / pi);                   % Compute ECD - Equivalent Circular Diameter.
if(blobArea<3500 & blobArea>2000 & blobPerimeter>100 & blobPerimeter<240 & meanGL>100 &meanGL<210)
i=i+1;
fprintf(1,'#%2d %17.1f %11.1f %8.1f %8.1f %8.1f % 8.1f\n', k, meanGL, blobArea, blobPerimeter, blobCentroid, blobECD(k));
% Put the "blob number" labels on the "boundaries" grayscale image.
text(blobCentroid(1) + labelShiftX, blobCentroid(2), num2str(k), 'FontSize', textFontSize, 'FontWeight', 'Bold');
%allBlobCentroids = [blobMeasurements.Centroid];
%centroidsX = allBlobCentroids(1:2:end-1);
%centroidsY = allBlobCentroids(2:2:end);
%plot(centroidsX(k), centroidsY(k), 'r+', 'MarkerSize', 10, 'LineWidth', 2);
blobMeasurements(i)=blobMeasurements(k);
end
end

allBlobCentroids = [blobMeasurements.Centroid];
centroidsX = allBlobCentroids(1:2:end-1);
centroidsY = allBlobCentroids(2:2:end);

x=[centroidsX;centroidsY]; %combining centroid matrix as Mx2 matrix
combinedCentroids=transpose(x);


% Detect the checkerboard.
%corimage=imread(imageFileNames{17});
```

```
[imagePoints, boardSize] = detectCheckerboardPoints(imageFileNames{21});

% Compute rotation and translation of the camera.
[R, t] = extrinsics(imagePoints, worldPoints, cameraParams);

% Get the world coordinates of the corners
worldPoints1 = pointsToWorld(cameraParams, R, t,combinedCentroids);

worldPoints1(1:i,:)

% blobAnalysis = vision.BlobAnalysis('AreaOutputPort', true,...
%     'CentroidOutputPort', false,...
%     'BoundingBoxOutputPort', true,...
%     'MinimumBlobArea', 2000,'MaximumBlobArea',3000, 'ExcludeBorderBlobs', true);
% [areas, boxes] = step(blobAnalysis, binaryImage);
%
% % Sort connected components in descending order by area
% [~, idx] = sort(areas, 'Descend');
%
% % Get the two largest components.
% boxes = double(boxes(idx(1:2), :));
%
% % Adjust for coordinate system shift caused by undistortImage
% boxes(:, 1:2) = bsxfun(@plus, boxes(:, 1:2), newOrigin);
%
% % Reduce the size of the image for display.
% magnification=25,
% scale = magnification / 100;
% imDetectedCoins = imresize(im, scale);
%
% % Insert labels for the coins.
% imDetectedCoins = insertObjectAnnotation(imDetectedCoins, 'rectangle', ...
%     scale * boxes, 'blob');
% figure; imshow(imDetectedCoins);
% title('Detected blobss');
%
% % Detect the checkerboard.
% [imagePoints, boardSize] = detectCheckerboardPoints(imageFileNames{21});
%
% % Compute rotation and translation of the camera.
% [R, t] = extrinsics(imagePoints, worldPoints, cameraParams);
%
% % Get the top-left and the top-right corners.
% box1 = double(boxes(1, :));
% imagePoints1 = [box1(1:2); ...
%                 box1(1) + box1(3), box1(2)];
%
% % Get the world coordinates of the corners
% worldPoints1 = pointsToWorld(cameraParams, R, t, imagePoints1);
%
% % Compute the diameter of the coin in millimeters.
% d = worldPoints1(2, :) - worldPoints1(1, :);
% diameterInMillimeters = hypot(d(1), d(2));
% fprintf('Measured diameter of one penny = %0.2f mm\n', diameterInMillimeters);
%
% % Get the top-left and the top-right corners.
```

```
% box2 = double(boxes(2, :));
% imagePoints2 = [box2(1:2); ...
%                 box2(1) + box2(3), box2(2)];
%
% % Apply the inverse transformation from image to world
% worldPoints2 = pointsToWorld(cameraParams, R, t, imagePoints2);
%
% % Compute the diameter of the coin in millimeters.
% d = worldPoints2(2, :) - worldPoints2(1, :);
% diameterInMillimeters = hypot(d(1), d(2));
% fprintf('Measured diameter of the other penny = %0.2f mm\n', diameterInMillimeters);
%
% % Compute the center of the first coin in the image.
% center1_image = box1(1:2) + box1(3:4)/2;
%
% % Convert to world coordinates.
% center1_world  = pointsToWorld(cameraParams, R, t, center1_image);
%
% % Remember to add the 0 z-coordinate.
% center1_world = [center1_world 0];
%
% % Compute the distance to the camera.
% distanceToCamera = norm(center1_world + t);
% fprintf('Distance from the camera to the first penny = %0.2f mm\n', ...
%     distanceToCamera);
```

B.5:Template Matching Algorithm

```
%   % Find maximum response
I = im2double(imread('lena.jpg'));
I=imrotate(I,30); %angle of rotation is 30 degrees

%   % Template of Eye Lena
T=I(124:200,124:200,:);

%   % Calculate SSD and NCC between Template and Image
[I_SSD,I_NCC]=template_matching(T,I);

%   % Find maximum correspondence in I_SDD image
[x,y]=find(I_SSD==max(I_SSD(:)));

%   % Show result
figure,
subplot(2,2,1), imshow(I); hold on; plot(y,x,'r*'); title('Result');
subplot(2,2,2), imshow(T); title('The eye template');
subplot(2,2,3), imshow(I_SSD); title('SSD Matching');
subplot(2,2,4), imshow(I_NCC); title('Normalized-CC');
```

B.6 Pattern Recognition Algorithm

```
clc;clear;
clearvars; % Get rid of variables from prior run of this m-file.
imtool close all;  % Close all imtool figures.
fprintf('Running the programme...\n'); % Message sent to command window.
workspace; % Make sure the workspace panel with all the variables is showing.
format long g;
format compact;
captionFontSize = 14;
%-------------------------------------------------------

%Camera Calibration

% Define images to process % overall mean error 2.24 pixel
imageFileNames = {'F:\IMG_3510.JPG',...
'F:\IMG_3511.JPG',...
'F:\IMG_3512.JPG',...
'F:\IMG_3530.JPG',...
'F:\IMG_3531.JPG',...
'F:\IMG_3532.JPG',...
'F:\IMG_3533.JPG',...
'F:\IMG_3536.JPG',...
'F:\IMG_3539.JPG',...
'F:\IMG_3540.JPG',...
'F:\IMG_3541.JPG',...
'F:\IMG_3542.JPG',...
'F:\IMG_3547.JPG',...
'F:\IMG_3552.JPG',...
'F:\IMG_3556.JPG',...
'F:\IMG_3559.JPG',...
'F:\IMG_3560.JPG',...
'F:\IMG_3561.JPG',...
'F:\IMG_3564.JPG',...
'F:\IMG_3565.JPG',...
'F:\IMG_3566.JPG',...
'F:\IMG_3567.JPG',...
'F:\IMG_3569.JPG',...
'F:\IMG_3570.JPG',...
'F:\IMG_3571.JPG',...
'F:\IMG_3572.JPG',...
'F:\IMG_3574.JPG',...
'F:\IMG_3575.JPG',...
'F:\IMG_3578.JPG',...
'F:\IMG_3579.JPG',...
'F:\IMG_3580.JPG',...
'F:\IMG_3581.JPG',...
'F:\IMG_3582.JPG',...
'F:\IMG_3583.JPG',...
'F:\IMG_3588.JPG',...
};

% Detect checkerboards in images
[imagePoints, boardSize, imagesUsed] = detectCheckerboardPoints(imageFileNames);
imageFileNames = imageFileNames(imagesUsed);
```

```
% Generate world coordinates of the corners of the squares
squareSize = 13.64;  % in units of 'mm'
worldPoints = generateCheckerboardPoints(boardSize, squareSize);

% Calibrate the camera
[cameraParams, imagesUsed, estimationErrors] =
estimateCameraParameters(imagePoints, worldPoints, ...
'EstimateSkew', false, 'EstimateTangentialDistortion', false, ...
'NumRadialDistortionCoefficients', 2, 'WorldUnits', 'mm');
% Display parameter estimation errors
%displayErrors(estimationErrors, cameraParams);

%Read the image and process them (only 2 for the moment)
clc;
Im1=rgb2gray(imread('IMG_3513.jpg'));
Im2=rgb2gray(imread('IMG_3525.jpg'));

[Im1,newOrigin]=undistortImage(Im1, cameraParams);
[Im2,newOrigin]=undistortImage(Im2, cameraParams);

% Detect the checkerboard.
%corresponding plane image : (imageFileNames{8});
[imagePoints, boardSize] =
detectCheckerboardPoints(imageFileNames{5});

% Compute rotation and translation of the camera.
[R, t] = extrinsics(imagePoints, worldPoints, cameraParams);

%Im2=rgb2gray(Im2);
%Im1=rgb2gray(Im1);
Im1=Im1(1049:1792,2805:2996,:);
x1=2805;
y1=1049;
x2=2996;
y2=1792;
P=[x1 y1;x1 y2]; %Vertical line in the image
P= pointsToWorld(cameraParams, R, t,P); % World coordinate of P
theta=atan((P(2,2)-P(1,2))/(P(2,1)-P(1,1)));
% Angle theta (in radians) to the horizontal of the vertical line in
world coordinate

%Im2=imrotate(Im2,1); %Angle is anticlockwise
a=size(Im1);
i=1;x=1;y=1;j=1;
s=60; %Size of the template sxs
%--k=1;

for (x=1:s:(a(2)-s-1))  %%%s/8
j=1;
for (y=1:s:(a(1)-s-1))   %%%s/8
boxImage=Im1(y:y+s-1,x:x+s-1,:); %x=y coordiante, y=x coordinate
sceneImage=Im2;
boxPoints = detectSURFFeatures(boxImage);
dimbP=size(boxPoints);

if (dimbP(1,1)>2) % at least 3 unique points
% for (m=1:2s:(b(1)-4s))
%    n=1
%   for (n=1:2s:(b(2)-4s))
%      sceneImage=Im2(m:m+4s,n:n+4s,:)
scenePoints = detectSURFFeatures(sceneImage);
```

```
%figure;
%imshow(boxImage);
%title('200 Strongest Feature Points from Box Image');
%hold on;
%plot(selectStrongest(boxPoints, 100));

%figure;
%imshow(sceneImage);
%title('5000 Strongest Feature Points from Scene Image');
%hold on;
%plot(selectStrongest(scenePoints, 5000));

%Extract Feature Descriptors
[boxFeatures, boxPoints] = extractFeatures(boxImage, boxPoints);
[sceneFeatures, scenePoints] = extractFeatures(sceneImage, scenePoints);

%Find Putative Point Matches
boxPairs = matchFeatures(boxFeatures, sceneFeatures);
dimbPs=size(boxPairs);

if(dimbPs(1,1)>2)

matchedBoxPoints = boxPoints(boxPairs(:, 1), :);
matchedScenePoints = scenePoints(boxPairs(:, 2), :);

%figure;
%showMatchedFeatures(boxImage, sceneImage, matchedBoxPoints, ...
%matchedScenePoints, 'montage');
%title('Putatively Matched Points (Including Outliers)');

[tform, inlierBoxPoints, inlierScenePoints] = ...
estimateGeometricTransform(matchedBoxPoints, matchedScenePoints, 'affine'); %affine is the original

%figure;
%showMatchedFeatures(boxImage, sceneImage, inlierBoxPoints, ...
%inlierScenePoints, 'montage');
%title('Matched Points (Inliers Only)');

boxPolygon = [1, 1;...                           % top-left
size(boxImage, 2), 1;...                         % top-right
size(boxImage, 2), size(boxImage, 1);...         % bottom-right
1, size(boxImage, 1);...                         % bottom-left
1, 1];                   % top-left again to close the polygon

newBoxPolygon = transformPointsForward(tform, boxPolygon);

% Get the world coordinates of the corners
newBoxPolygon = pointsToWorld(cameraParams, R, t,newBoxPolygon);

%figure;
%imshow(sceneImage);
%hold on;
%line(newBoxPolygon(:, 1), newBoxPolygon(:, 2), 'Color', 'r');
%title('Detected Box');

%Finding centroid of the detected area
centroidsX(i,j)=(newBoxPolygon(1,1)+newBoxPolygon(3,1)+newBoxPolygon(2,1)+newBoxPolygon(4,1))/4;
```

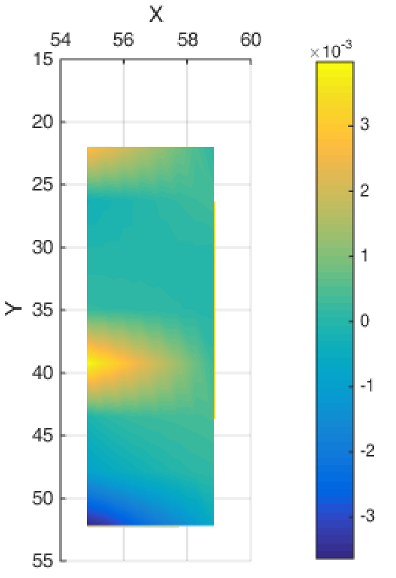
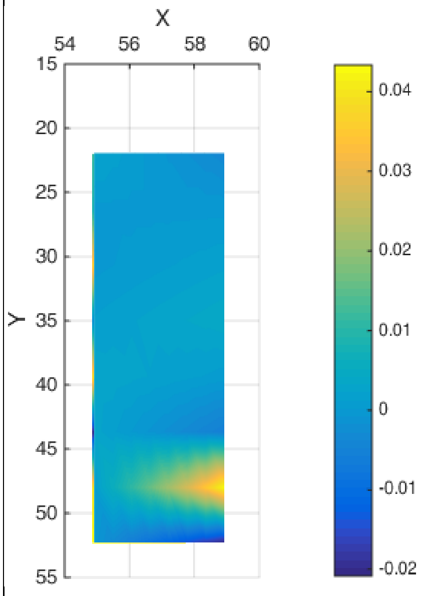
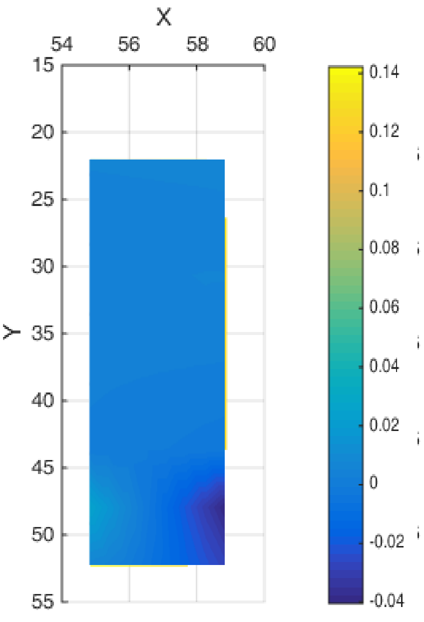
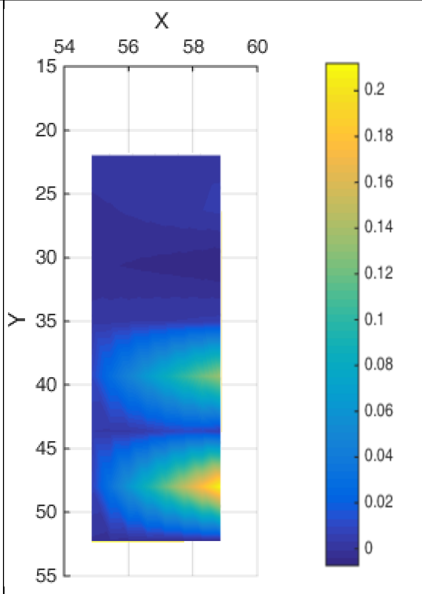
```
centroidsY(i,j)=(newBoxPolygon(1,2)+newBoxPolygon(3,2)+newBoxPolygon(2,2)+newBoxPolygon(4,2))/4;

initialCentroid=[x1+(x-1)+s/2 y1+(y-1)+s/2];
initialCentroid = pointsToWorld(cameraParams, R, t,initialCentroid);
centroidsX1(i,j)=initialCentroid(1,1);
centroidsY1(i,j)=initialCentroid(1,2);
% Calculate SSD and NCC between Template and Image
%--[sceneImage_SSD,sceneImage_NCC]=template_matching(boxImage,sceneImage);
% Find maximum correspondence in I_SDD image
%--[x1,y1]=find(sceneImage_SSD==max(sceneImage_SSD(:)));
%--else
%--end
%--centroidsX(i,j)=y1;
%--centroidsY(i,j)=x1;
%disp(x1);
%disp(y1);%figure;imshow(sceneImage);title('image');plot(x1,y1,'r*');
%--k=k+1;
j=j+1;
%y=y+s/2;
end
end
end

i=i+1;
%x=x+s/2;
end

fprintf('First loop completed...\n');
dimTemplate=size(centroidsX);
T1=dimTemplate(1,1);
T2=dimTemplate(1,2);
i=1;j=1;
bmw=centroidsX;
while(j<(T2+1))     %Rearramge the matix
fprintf('loop1 \n');
while(i<T1)
%fprintf('loop2 \n');
if(centroidsX(i,j)==0)
if(centroidsX(i+1,j)~=0)
fprintf('analysing %d %d \n',i,j);
centroidsX(i,j)=centroidsX(i+1,j);
centroidsX1(i,j)=centroidsX1(i+1,j);
centroidsY(i,j)=centroidsY(i+1,j);
centroidsY1(i,j)=centroidsY1(i+1,j);
centroidsX(i+1,j)=0;
centroidsX1(i+1,j)=0;
centroidsY(i+1,j)=0;
centroidsY1(i+1,j)=0;
i=0;
end
end
i=i+1;
end
j=j+1;
i=1;
fprintf('i,j = %d %d',i,j)
end
```

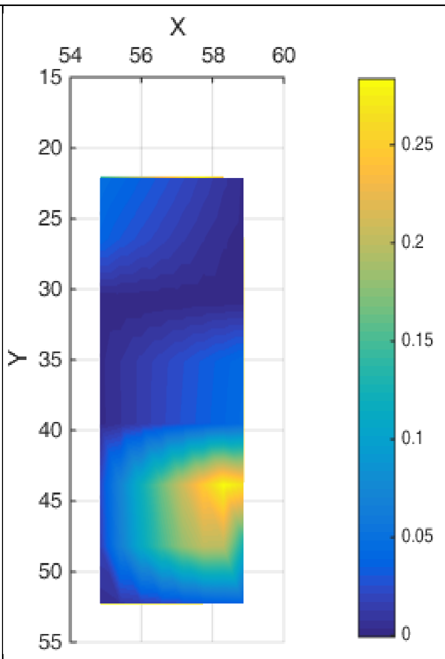
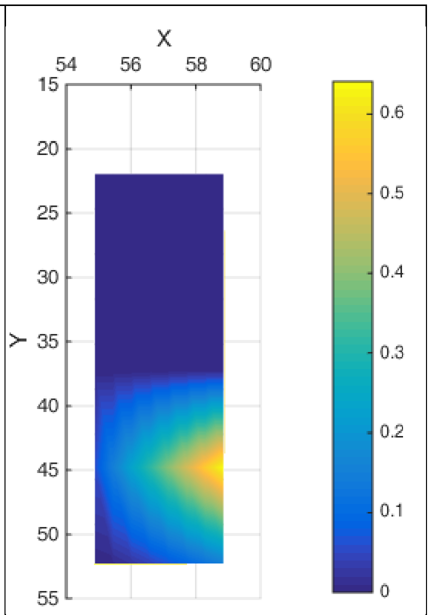
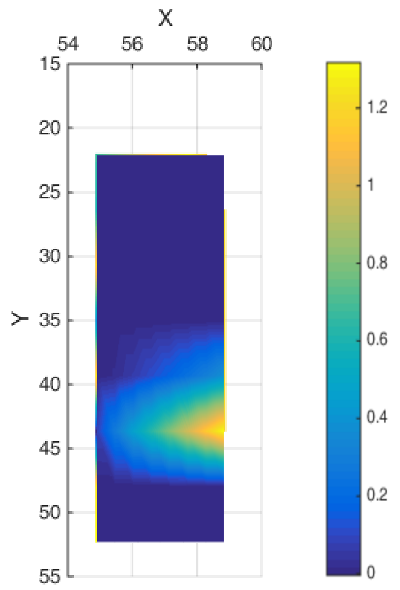
```
m=2;n=1;
z=0;X=0;Y=0;Z=0;
a=2;
%Finding out the deformation and centroid of connecting lines

for(m=2:1:T1)
b=1;
for(n=1:1:T2)
if(centroidsX(m,n)~=0)
strain(a,b) = (-1)*(sqrt(((centroidsY1(m,n)-centroidsY1(m-1,n))^2)+((centroidsX1(m,n)-centroidsX1(m-1,n))^2))-sqrt(((centroidsY(m,n)-centroidsY(m-1,n))^2)+((centroidsX(m,n)-centroidsX(m-1,n))^2)))/sqrt(((centroidsY1(m,n)-centroidsY1(m-1,n))^2)+((centroidsX1(m,n)-centroidsX1(m-1,n))^2));
centroidLineY(a,b)=((centroidsY1(m,n)+centroidsY1(m-1,n))/2);
centroidLineX(a,b)=((centroidsX1(m,n)+centroidsX1(m-1,n))/2);
%if(abs(centroidsLineY(a,b)-centroidsLineY(a-1,b))>2s)   % maximum strain 300%

if(strain(a,b)~=0)
Z(a-1,b)=strain(a,b);
fprintf('\n strain(%d,%d) = %3f, centroidsY(%d,%d)-centroidsY(%d-1,%d) = %3f, centroidsY1(m,n)-centroidsY1(m-1,n) = %3f',a,b,strain(a,b),m,n,m,n,(centroidsY(m,n)-centroidsY(m-1,n)),(centroidsY1(m,n)-centroidsY1(m-1,n)));
%if(m==2 & n==2)
%Y=[centroidLineY(m,n-1) centroidLineY(m+1,n-1) centroidLineY(m+1,n) centroidLineY(m+1,n+1) centroidLineY(m,n+1) centroidLineY(m-1,n+1) centroidLineY(m-1,n) centroidLineY(m-1,n-1)];
%X=[centroidLineX(m,n-1) centroidLineX(m+1,n-1) centroidLineX(m+1,n) centroidLineX(m+1,n+1) centroidLineX(m,n+1) centroidLineX(m-1,n+1) centroidLineX(m-1,n)  centroidLineX(m-1,n-1)];
%else

Y=[Y centroidLineY(a,b)];
X=[X centroidLineX(a,b)];
b=b+1;
end
end
end
a=a+1;
end
m=1;
B=(reshape(Z,[],1));  %convert Z matrix into row matrix
p=size(B);
for(k=1:1:p(1,1))
if(B(k,1)~=0)
A(m,1)=B(k,1);
m=m+1;
end
end

X=(X(2:end))';
Y=(Y(2:end))';

n=10; %number of subdivisions for plotting purposes
[xi,yi]=meshgrid(linspace(min(X),max(X),n),linspace(min(Y),max(Y),n)); %set limits for x and y
%[xi,yi]=meshgrid(min(X):0.1:max(X),min(Y):0.1:max(Y));%
```

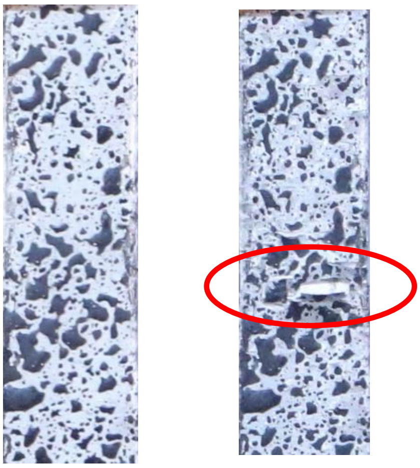
```
zi=griddata(X,Y,A,xi,yi,'natural'); %interpolate Z to generate zi for complete grid
%contour(xi,yi,zi);
%figure;
surf(xi,yi,zi); shading('interp');
%scatter(xi,yi,zi);
xlabel('X'); ylabel('Y'); zlabel('Z');
```

Appendix C: Strain Contours

Specimen 1:

Time (mins)	Strain Contour	Time (mins)	Strain Contour
1		2	
3		4	

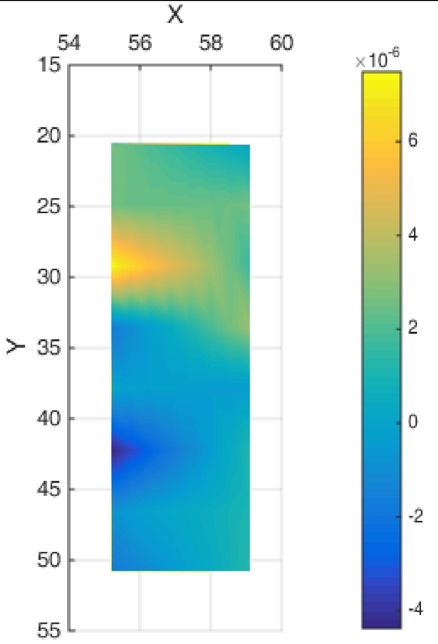
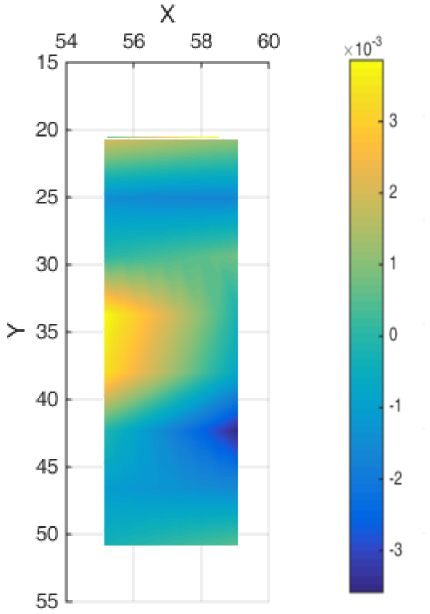
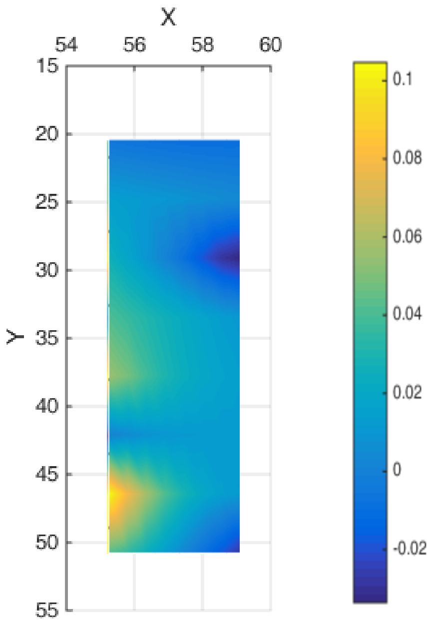
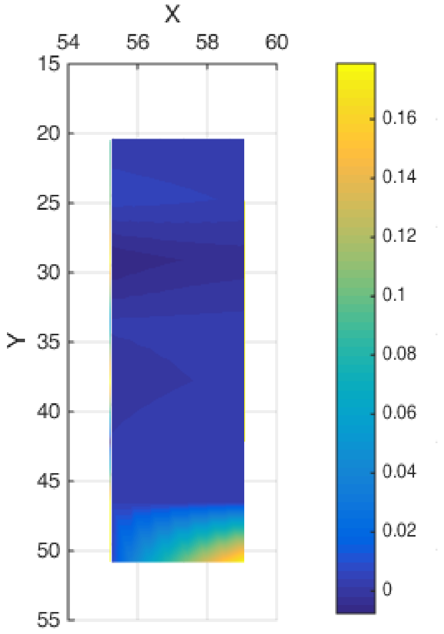
5	X 54 56 58 60 Y 15 20 25 30 35 40 45 50 55 0.25 0.2 0.15 0.1 0.05 0	6	X 54 56 58 60 Y 15 20 25 30 35 40 45 50 55 0.6 0.5 0.4 0.3 0.2 0.1 0
8	X 54 56 58 60 Y 15 20 25 30 35 40 45 50 55 1.2 1 0.8 0.6 0.4 0.2 0		

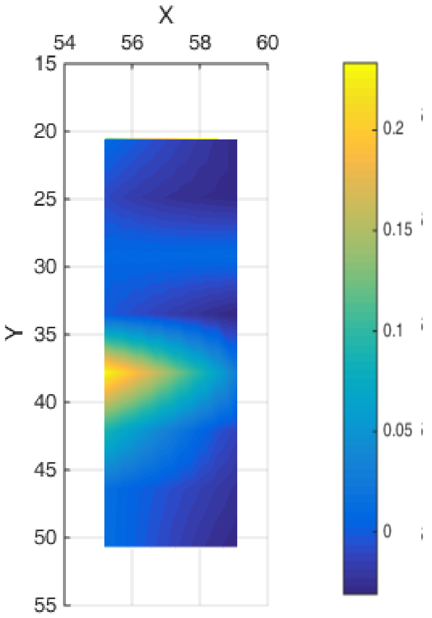
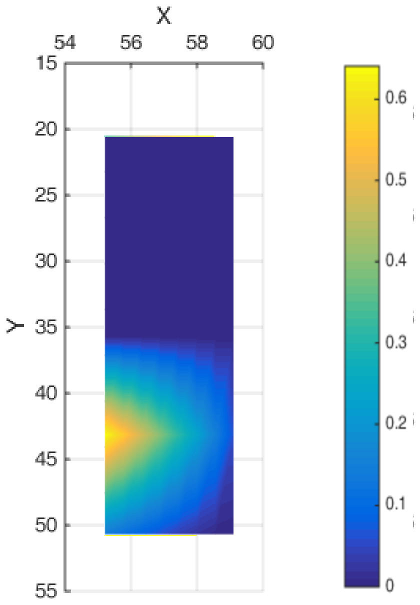
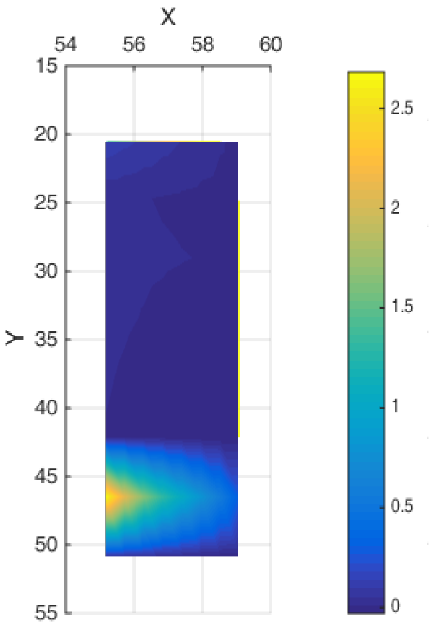


(a) (b)

Figure 0.1: ROI (a) before loading (b) during necking

Specimen 2:

Time (mins)	Strain Contour	Time (mins)	Strain Contour
1		2	
3		4	

5		8	
9			

(a) (b)

Figure 0.2: ROI (a) before loading (b) during necking