2015

Measuring in-plane deflections and strains through visual sensing techniques for civil infrastructure applications

Youyi Feng

MEASURING IN-PLANE DEFLECTIONS AND STRAINS THROUGH VISUAL SENSING TECHNIQUES FOR CIVIL INFRASTRUCTURE APPLICATIONS

Youyi Feng

Thesis submitted to the College of Engineering and Mineral Resources
at West Virginia University
in partial fulfillment of the requirements
for the degree of

Master of Science
In
Civil Engineering

Fei Dai, Ph.D., Chair
Roger Chen, Ph.D.
Radhey S. Sharma, Ph. D.
Yoojung Yoon, Ph.D.

Department of Civil and Environmental Engineering

Morgantown, West Virginia
2015

Keywords: Visual sensing, infrastructure safety, strains, DIC, SIFT, SURF

Abstract

MEASURING IN-PLANE DEFLECTIONS AND STRAINS THROUGH VISUAL SENSING TECHNIQUES FOR CIVIL INFRASTRUCTURE APPLICATIONS

Youyi Feng

Maintaining the integrity and safety of civil infrastructures such as bridges, dams, tunnels and high-rise buildings is an essential task for civil engineers. Collapse or damage of these civil infrastructures may lead to a tremendous amount of injuries and casualties. To alleviate this situation, a real-time surveillance method enabled by visual sensing techniques is proposed in this thesis. The advances of applying visual sensing techniques, for instance, are allowing practical deployment for large extended systems in a more cost-effective way. Also, the image or video data can be easily used for long-term condition assessments.

The proposed method entails applying visual sensing techniques to measure in-plane deflections and strains of structural members for civil infrastructure applications. In specific, it employs visual sensors (digital/industrial cameras) to capture and record a series of continuous image frames of the targets. Then automated feature detection and matching algorithms are applied to detect and match object features in the consecutive image frames. Based on the location information of the detected features, the in-plane object displacement can be accurately calculated through keeping tracking those features in the continuous image frames. Next, an optimized interpolation procedure is conducted to obtain dense displacement field for the object. And the strains can be consequently recovered from the displacement field through computing its derivatives.

In this research, firstly, the work of evaluating the optimum feature detection and matching algorithm is reported, which is the key task to achieve accurate surveillance. A series of experiments were conducted to compare the three algorithms: DIC (Digital Image Correlation), SIFT (Scale Invariant Feature Transform), and SURF (Speeded-Up Robust Features). The experimental result indicated that the DIC algorithm reveals superiority among the three algorithms and holds the most potential for measuring in-plane deflections and strains of civil infrastructures. To further validate our method, we employed high-speed industrial camera (Manta G223B) to capture a series of continuous image frames of deformed real-world scenarios. The DIC algorithm was adopted for the feature detection and matching process. As the output, the displacement and strains were calculated and then compared with the ground truth in order to evaluate the accuracy performance of the method. Colored strain maps were generated by using different colors to reflect different strain levels in an intuitive way. The experimental result indicated that our method can achieve highly accurate measuring performance of computing in-plane displacements and strains for civil infrastructure applications. The proposed method has several advantages when compared to pre-existing methods (such as sensor networks). It can generate accurate full-field deflections and strains of the target. Besides, the cost-effective equipment and much more convenient set-up procedures will enable engineers to operate periodically and apply for different scales of civil infrastructure applications.

Dedication

This thesis is dedicated to my beloved parents, sister and girlfriend. Thank you for offering me the unconditional love and support.

Acknowledgements

I wish to thank my adviser, Dr. Fei Dai for his guidance and inspiring ideas throughout the research. I would like to extend my special thanks to my committee member, Dr. Roger H. L. Chen, also Mr. Zhanxiao Ma and Dr. Mark L. Skidmore for the assistance and valuable suggestions in conducting the LVDT experiments. I would also like to express my appreciation to my committee members, Dr. Radhey Sharma and Dr. Yoojung Yoon for reviewing this work.

I am thankful to my office mates Mr. Haidar Aldaach and Dr. Murat Dinc for their help in the laboratory. I am thankful to my friends for all their help during my stay at WVU. I would also like to thank my roommates Wei Qi and Jiaxin Li for making my stay in Morgantown a very pleasant and memorable one.

I also wish to express my gratitude to my family and girlfriend, their support and love gave me the motivation to do my best in life.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

With the continuous development of human civilization, more and more civil infrastructures have come into being in our today's life. However, along with the emergence of those infrastructure applications such as bridges, overpasses, tunnels, dams, and high-rise building, the security of the infrastructures has become a crucial issue for civil engineers. Recent research work has shown that quite a few catastrophes in civil engineering field are associated with the failure of those civil infrastructures (e.g., collapse of bridges, dams and tunnels) (Chang et al. 2003). As an inevitable consequence, the collapse will result in enormous loss, injuries and casualties.

In order to ameliorate this situation and to improve the security, a real-time surveillance method enabled by visual sensing is proposed in this thesis. It proposes to utilize high-speed industrial cameras to measure the deflections and strains for civil infrastructure applications so that their integrity and safety can be monitored in a cost-effective way while they are undergoing excitements during operations. In this method, key features on the target object's surface are continuously detected and matched to quantitatively measure the deflection values of the target, which then can be further processed into strains (Young and Budynas 2002).

The objective of this research is to measure the in-plane deflections and strains for civil infrastructure applications. However, different algorithms might be applied for the feature detection and matching procedure in the visual sensing-based method; and the performance of the algorithms has not been compared for measuring in-plane deflections and strains of civil infrastructures in terms of accuracy and efficiency. To address this

problem, firstly, we need identify the optimal feature detection and matching algorithm for the visual sensing-based method. In specific, we evaluate three selected feature detection and matching algorithms DIC (Digital Image Correlation), SIFT (Scale Invariant Feature Transform), and SURF (Speeded-Up Robust Features). In this research, measuring accuracy and running efficiency of the algorithms are compared in detail. Also, the influences on the measuring accuracy of the three algorithms when utilizing images with different resolutions and using different camera shooting distances have been evaluated and analyzed.

A series of experiments were conducted to compare the three algorithms. The experimental result indicated that the DIC algorithm reveals superiority among the three algorithms and holds the most potential for measuring in-plane deflections and strains of civil infrastructures. To further test the method, we employed high-speed industrial camera (Manta G223B) to capture a series of continuous image frames a concrete sample under deforming in WVU structural lab. Then, the image correlation-based algorithm was adopted for the feature detection and matching procedure. As the output, the displacement and strains were calculated and then compared with the ground truth in order to evaluate the accuracy performance of the method. Colored strain maps were generated by using different colors to reflect different strain levels in an intuitive way. The experimental result indicated that our method can achieve highly accurate measuring performance of computing in-plane displacements and strains for civil infrastructure applications. The proposed method has several advantages when compared to pre-existing methods (such as sensor networks). It can generate accurate full-field deflections

and strains of the target. Besides, the cost-effective equipment and much more convenient set-up procedures will enable engineers to operate periodically and apply for different scales of civil infrastructure applications.

The outline of this thesis can be described as follows: Chapter 2 presents the detailed literature review of current existing methods for measuring deflections and strains in civil engineering related field. Chapter 3 explains our motivation and objectives to conduct the research work in detail. Also, a flow chat is generated in order to show the whole procedures of our research work. Chapter 4 focuses on the methodologies applied in our research to implement the designed comparison and validation experiments. The experimental results are presented in Chapter 5, and then Chapter 6 further provides some in-depth discussion and analysis regarding the experimental results presented in Chapter 5. Finally, conclusion of the research and future work is presented in Chapter 7.

CHAPTER 2: BACKGROUND

Civil infrastructure applications should meet the safety, serviceability and durability requirements under certain circumstance (Karbhari and Zhao 2000). Once any of the requirements is not strictly satisfied, the integral security of the infrastructure applications will definitely be threatened. The failures will undoubtedly induce tremendous loss, delay, injuries, and causalities as the consequence.

The table below shows some collapse accidents that are related with civil infrastructure applications in recent years around the world.

Table 1. Collapse accidents of civil infrastructure applications in recent years

Time	Location	Injuries/Fatalities	Accident depiction
7/10/06	Boston, USA	1 injured, 1 deaths	Boston Fort-Point tunnel collapse
9/30/06	Quebec, Canada	6 injured, 5 deaths	Lawal city overpass collapse
3/27/09	Jakarta, Indonesia	130 injured, 96 deaths	Collapse of Situ-Gintung dam
12/3/12	Yamanashi, Japan	2 injured, 9 deaths	High-way tunnel collapse
1/21/14	Lai Chua, Vietnam	37 injured, 7 deaths	Collapse of a bridge across the river
5/03/14	Guangdong, China	16 injured, 11 deaths	Bridge collapse under construction

To alleviate this issue, we propose to properly monitor the safety of civil infrastructure applications by measuring their real time dynamic deflections and deformations. It aims at ensuring whether the deformation is within limits in terms of stability (Brownjohn 2007). Hence, while strains or stress may be measured, the emphasis is on measuring deflections. In our research, the deflection is defined as the spatial displacement of structural members, which can be computed by recovering the target's spatial coordinates as time goes by. After obtaining the deflection information of civil infrastructures, strains of the target can be recovered and corresponding preventive decisions can be made in response to corresponding safety situations, for instance, sounding an alarm when the strains of civil infrastructure exceed the pre-specified threshold. This action will potentially help to reduce those unexpected accidents, and also gain more evacuation time for people to escape from the terrible disasters (Rainieri et al. 2011).

2.1 Current methods for measuring deflections of civil infrastructures

This section mainly focuses on the available methods that can be applied to measure deflections of civil infrastructures. These methods can be introduced as follows: based on the spatial location relationships between the measuring instrument and the

target, the methods can be divided into two main categories: contact and non-contact measuring methods.

2.1.1 Contact measuring methods

Contact measuring method, namely, the methods that require the measuring instruments to be put into the target or installed onto the surface of the target.

Wire/wireless sensor networks: Dargie and Poellabauer summarized the wireless sensor networks method for monitoring civil infrastructure applications (Dargie and Poellabauer 2010). It is the most typical method using for measuring deflections and strains. In terms of this method, usually professional operators will install the wire/wireless sensors onto the target that needs to be monitored, whereby they can collect the target's spatial positon changing information (Kim et al. 2007). This method can achieve sufficient accuracy performance for measuring deflections. However, although the development of wireless sensing technique may reduce the extra expense of the wire transmission to some extent (Lynch and Kenneth 2006), the convenience of switching operation between different measuring targets still needs improvement. Besides, when facing relative large-scale applications, the number of sensors needed for installation and uninstallation will be another issue which calls for extra efforts (Yuan et al. 2012). What is more, since the sensor networks method can only detect the deflections of those particular positions where the sensors were put onto, the full-field accurate deflections of the integral target still cannot be achieved (Chintalapudi et al. 2006).

Global Positioning System (GPS) based methods:

The GPS based method belongs to contact measuring method since the GPS receivers will be put onto the surface of the target to collect the data. Figure 1 shows the general schematic of the GPS deployment on a high-rise structure.

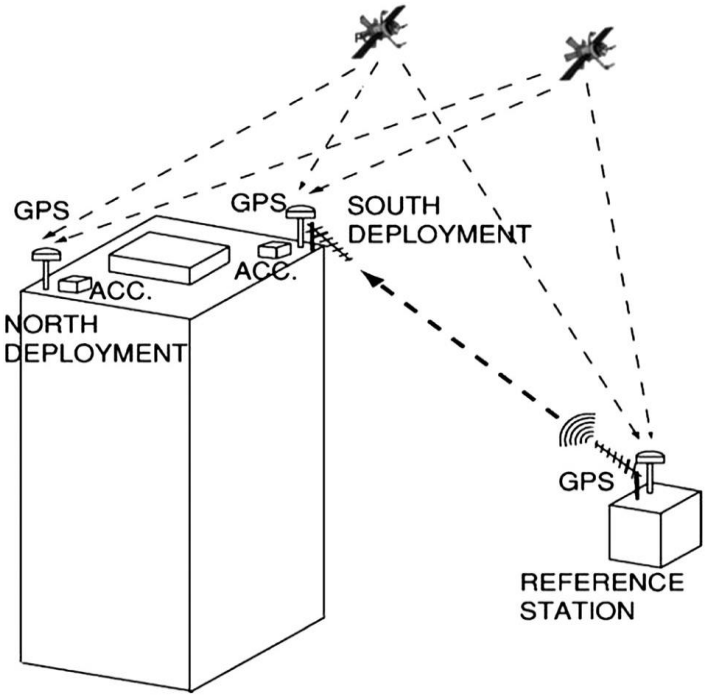


Figure 1. General schematic of the GPS deployment on a high-rise structure (Ting et al. 2013)

The GPS methods include static, fast-static, and RTK (real-time kinematic) modes. Some previous research in the GPS monitoring of civil engineering structures is about the static monitoring of settlements and deflection trends for banks or dams. The RTK method is also applied in structural health monitoring (SHM). In the RTK mode, the reference-station is considered as a fixed station point for checking, and the point's 3D coordinates can be determined by the static GPS method and by recording the difference

between its already known spatial position and calculated spatial positon from the satellite data (Ting et al. 2013).

PVDF (polyvinylindine fluoride) film sensor: For this method, according to the fact that the larger the PVDF film area, the more charge is produced after being squeezed, and also the piezoelectric constant along the stretch direction is the largest, the size of the film, including the film area and the length to width ratio, plays an important role in sensor design. Four different sizes have been investigated with different area and different length-to-width ratio (length is along the stretch direction of PVDF film). They were mounted to the same place of a cantilever beam with one end fixed and the other being free to be moved up or down to generate mechanical deformation. The dimensions of the sensor patch are shown in Figure 2 (Gu et al. 2005).

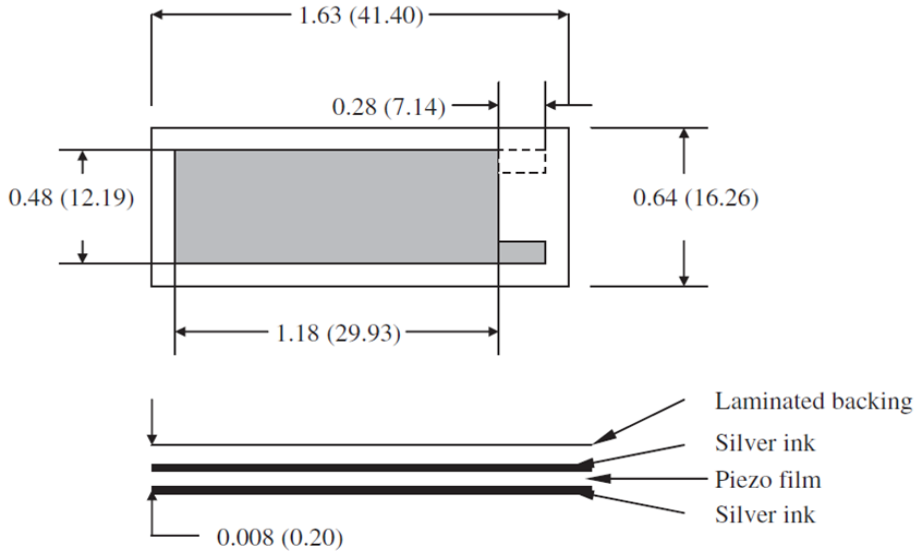


Figure 2. Dimensions of PVDF sensor element (Gu et al. 2005)

2.1.2 Non-contact measuring methods

Non-contact measuring methods, also known as remote sensing, enable the measuring tools to be a few meters or even tens of meters away from the target, which means that the stations of the measuring operations are beyond the limitation of the target's position. Due to this favorable property, non-contact measuring methods may hold more potential to be applied to measuring deflections when considering the operating convenience (Jonckheere et al. 2004). There are a series of methods have been proposed in relevant research such as laser scanning (Monserrat and Crosetto 2008), image/video-based method (Feng and Dai 2014) and total station surveying etc.

Based on the measuring properties of different instruments, these methods can further be divided into active measuring methods and passive measuring methods (Ulaby et al. 1982). Active measuring basically means the measuring instrument itself will emits energy onto the target's surface when measuring that target, while passive measuring means the instrument will not emit any energy to the target throughout the whole measuring process (Sabins 2007). For example, laser scanning is a typical active measuring method. It emits laser rays onto the target to obtain its spatial position information. On the other hand, image/video-based method belongs to the category of passive measuring category. It utilizes digital/industrial cameras to capture image/video streams of the target to recover the spatial information of the target (Elgamal et al. 2003), while emitting no energy onto the target surface in the measuring process.

Laser scanning method: In 2006, Alba and Fregonese presented their work of monitoring deformations of large concrete dams by terrestrial laser scanning (Alba et al. 2006). This method generates 3D point cloud of the target. Based on the point cloud before and after the target deforming, the spatial deflection information of the target can be obtained (Park et al. 2007). However, the main drawback of this method is that the expense of 3D laser scanner that can be used for accurate surveying is normally over thousands of dollars, which has induced this method actually not really practical for the research with relatively low cost to measure the deflections or deformation of civil infrastructure applications.

Total station surveying: This surveying method can also be applied to measure the deflections of civil infrastructures (Maas and Hampel 2006). In this method, several special markers will be placed on the target, and then the total station machine will be operated by professionals to record the spatial coordinates of the markers to acquire the target's positon changing information. However, using total station faces the same problem with applying sensor networks. That is, it cannot achieve full field measurement of deflections for targets.

Visual sensing-based method: Considering the above mentioned unfavourable factors of the conventional methods, image sensing technique is employed to address above existing issues. Image sensing technique, namely, applying image/video sensors (such as digital/video cameras, industrial cameras, etc.) to capture and record the spatial position information of the targets (normally, the output format will be digital image or video streams), and then based on relevant imaging principles of the camera to recover

the real world spatial position information of the target. After obtaining the necessary spatial coordinates, the deflection of the target can be computed as the output (Wahbeh et al. 2003). This method has several advantages when compared to the conventional methods. Firstly, it can generate accurate full-field deflections of the target that the images covered. Secondly, it is very convenient to be operated periodically and applied for different scales of civil infrastructure application (Wang and Cuitiño 2002). As a result, this image sensing-based method can be a highly potential alternative to be applied to measure deflections of civil infrastructures.

The goal of our research is to establish such a visual sensing-based method that can real-time measure the full-field deflections of load bearing members of civil infrastructure applications. The method entails utilizing high-speed video/industrial camera to capture a series of target image streams such that the target's spatial deflection can be real time computed so as to alert the engineers when the deflection is in large scale and may cause an accident.

In our research, the basic principle of the deflection measuring method lies in detecting and matching interest feature points in a series of continuous image frames to obtain the position changing values of the features. Then, based on the location information of the detected features, the in-plane object displacement/deflections can be accurately calculated through keeping tracking those features in the image frames. In specific, the accuracy of this deflection measuring method is entirely associated with the interest points' location in each image frame. Hence, the key task to achieve a high-accuracy measuring method is totally determined by the feature detection and matching

results (Küntz et al. 2006). As a consequence, obtaining the optimal feature detection and matching algorithm has priority over all other tasks in our current research.

2.2 Feature detection and matching algorithms

A series of algorithms have been developed to detect and match feature points along image streams. These algorithms can be categorized into two types (Govender 2009). The first type is feature-based pixel level matching algorithm. In this type, Scale-Invariant Feature Transform (SIFT) proposed by Lowe is known as the most typical algorithm (Lowe 2004). Speeded-Up Robust Features (SURF) is another feature-based matching algorithm proposed by Bay in 2006 (Bay et al. 2006). It inherits the property of scale-invariant features, and its running efficiency has been proved to be higher than SIFT (Luo and Gwun 2009). The reason that we picked these two algorithms to test in our experiments is because previous relevant research has revealed that the SIFT and SURF detectors and descriptors have priority over other detectors and descriptors (Zhu and Davari 2014), such as HOG (Histogram of Oriented Gradients) and GLOH (Gradient Localization Oriented Histogram) etc. (Mikolajczy and Cordelia 2005).

2.2.1 Scale-Invariant Feature Transform (SIFT)

David Lowe proposed the Scale-Invariant Feature Transform (SIFT) algorithm in 1999 (Lowe 1999). This algorithm has been used for object detection, recognition and image matching, etc. It was further improved in 2004. SIFT (Scale-Invariant Feature Transform) operator is a type of local image descriptor, with scale, rotation, translation invariance. It also has certain robustness to changes in illumination, affine transformation

and three-dimensional projection transformation. In Mikolajczyk's comparative experiments of comparing dozens of local invariant descriptors including SIFT and its expansion descriptors. The experimental results revealed that SIFT and its expansion algorithm has shown to have the most robustness in those descriptors (Mikolajczy and Cordelia 2005). The main idea of SIFT algorithm is to find the extreme points in image scale space (not the extreme points on the plane), and then filter the extreme points to find several stable feature points. Finally extract the local characteristics of the image around each stable feature point, and the formation of local descriptors will be used in subsequent matching. The theory of SIFT algorithm solves the problem of scale invariance, that is to say, regardless of the scale size of the same object in the picture, can be extracted as the same feature points by SIFT algorithm.

The features extracted by SIFT algorithm are local features of the image. Those features have scale invariant property to spatial translation, rotation, scale zooming, brightness variation, occlusion and image noise. The algorithm also has certain stability to visual changes of the images and affine transformation.

The feature detecting and matching procedures of Scale-Invariant Feature Transform (SIFT) algorithm can be described as the following four main steps:

- **Image scale space**: generate Gaussian pyramid models for the images.
- **Detection of local extremum** (local maxima or minima): firstly, calculate differential Gaussian pyramid models for images; secondly, extract extremum candidates of the image based on the differential model; thirdly, pick out real

extremums, those that have low contrast values or are poorly localized along edges are removed in this step.

- **Feature descriptor**: 1) firstly, calculate the dominant orientations of each extremum; secondly, generate gradient histogram to represent the gradient direction within the feature point; thirdly, the peak values of gradient histogram represent the dominant orientations of the feature points; 2) specify a N by N window (normally N=16) for each feature points, then generate multidimensional feature descriptors at the central region of the window based on the gradient histograms.

- **Feature matching**: calculate the distances between feature points, those features that have the minimum distance are determined as matched feature pairs.

SIFT descriptor have the following properties (Khan et al. 2011). Firstly, it is invariant to scale transform and spatial rotation due to the features is determined by local maxima or minima across scales and their dominant orientations. Besides, the detected features have illumination invariance. These favorable properties lead SIFT algorithm one of the most powerful algorithms for feature detection and matching in lots of related areas (Nghiem et al. 2007).

Golparvar-Fard et al. presented their work for segmentation and recognition of highway assets using image-based 3D point clouds (Golparvar-Fard et al. 2012). In their method, SIFT algorithm implemented on GPU is applied. Next, using a new multicore implementation, the SIFT features are matched in pairs over the span of Ω consecutive

video frames. An initial solution for the 3D locations of these features points is calculated using Nister's 5-point algorithm. Then, the objective function for the distance between SIFT features and their re-projected 3D points at every iteration is minimized through an optimization process using the multicore sparse bundle adjustment library (Wu et al. 2011). This process results in a sparse point cloud model plus intrinsic and extrinsic camera parameters for each video frame which are fed into the MVS algorithm (Furukawa et al. 2009) to improve density of the sparsely reconstructed model.

Jahanshahi and Masri proposed the adaptive vision-based crack detection method by using 3D scene reconstruction for condition assessment of structures (Jahanshahi and Masri 2012). In the method, SIFT key-points (Lowe 2004) are detected in each image and then matched between all pair of images. The RANSAC algorithm (Fischler and Bolles 1981) is used to exclude outliers. These matches are used to recover focal length, camera center and orientation, and radial lens distortion parameters (two parameters corresponding to a 4th order radial distortion model. Their experimental results reveal the method has good potential to detect cracks for civil structures.

Some advantages and disadvantages of the Scale Invariant Feature Transform (SIFT) algorithm are shown as below:

- ***Advantages:***

1)Feature uniqueness is good, informative, and suitable for extracting and matching rapid massive characteristics in the database

2)Sufficient features, even though a handful of objects in the images, it can also

generate a lot feature points from the images.

3)Relatively fast, Sift optimized matching algorithm can even achieve real-time requirements compare with some global matching algorithms.

4)The extracted features can easily be used to combine with other forms of eigenvectors.

➢ ***Disadvantages:***

1)The running efficiency of the algorithm is still not good when comparing with some real time matching algorithms, such as blocking matching.

2)Sometimes insufficient feature points for non-texture areas of the image.

3)For smoothing edges in the image, it cannot accurately extract the feature points of the object.

2.2.2 Speeded-up Robust Features (SURF)

SURF was proposed by Herbert Bay in 2006. This algorithm employed the Hessian matrix to extract image extremums. Image features are localized by applying a non-maximum suppression schema across image scales (Bay et al. 2008).

➢ Five main procedures involved in SURF algorithm:

1) Generate Hessian matrix for the image to be processed

2) Generate scale space for the image

3) Feature point precise localization based on the generated Hessian matrix in the

scale space

4) Determine dominate orientations for those feature points

5) Feature description and matching

Henssian matrix used in SURF algorithm has excellent stability when extracting local extreme points for the images. However, it is also dependent on the direction of the gradient of the local region of pixels. It is possible to find the incorrect dominant direction in the feature point extraction and matching process. Sift a grayscale algorithm using only the nature properties of the algorithm. It ignores the color information of the images, while Surf's descriptor can take use of the color information in the feature extracting and matching process.

As the related application, SURF algorithm has also been applied to automatically generate sparse 3D points for acquiring civil infrastructure' geometric data in Fathi and Brilakis' paper (Fathi and Brilakis 2011). An automated stereo vision-based method is proposed, as an alternative and inexpensive solution, to producing a sparse Euclidean 3D point cloud of an infrastructure scene utilizing two video streams captured by a set of two calibrated cameras. In this process SURF features are automatically detected and matched between each pair of stereo video frames. 3D coordinates of the matched feature points are then calculated via triangulation. The detected SURF features in two successive video frames are automatically matched and the RANSAC algorithm is used to discard mismatches. They have validated their method a competitive one to recover spatial geometric data for civil infrastructure applications.

The advantages and disadvantages of the Speed up Robust Features (SURF) algorithm are shown as below (Luo and Gwun. 2009):

- Advantages:

1) Comparing with SIFT algorithm, its computing efficiency (running time) is much higher than SIFT algorithm.

2) SURF algorithm also has scale transform and spatial rotation invariance property when extracting and matching the image features.

- Disadvantages:

1) The algorithm is sensitive to illumination variances, which means it has difficulty to process the images under different light condition.

So far, SURF algorithm has been successfully applied in several related research fields, such as object detection and recognition (Duy-Nguyen et al. 2009), 3D reconstruction (Segundo et al. 2012).

2.2.3 Digital Image Correlation (DIC)

Another type of algorithm is pixel based sub-pixel level matching algorithm. Digital image correlation belongs to this type. It has been applied in other applications such as industrial parts deformation detection in mechanical field. Related research has shown the DIC algorithm has great potential in mechanical field (Zhao et al. 2012). However, the performance of this algorithm in civil infrastructure applications still cannot be identified due to the distinct differences between civil engineering and

mechanical fields. For example, speckle patterns are usually applied for DIC algorithm in experimental testing of mechanical applications (Bornert et al, 2009); whereas the speckle patterns are actually not appropriate to be utilized in civil infrastructure applications.

On the other hand, civil infrastructure applications, such as tunnels or dams, are usually in much greater scales than that of mechanical applications. This also may cause the uncertainty of applying DIC algorithm in civil engineering field. As a result, to further identify whether DIC algorithm can still reveal great potential to be utilized to measure deflections of civil infrastructures, we need compare the DIC algorithm with the settings suitable for civil infrastructure applications with conventional feature matching algorithms, such as SIFT and SURF.

Digital Image Correlation (DIC) is an innovative non-contact optical technique for measuring strain and displacement. It employs image registration and tracking techniques to measure the planar or spatial deflection and deformation within a series of continuous image frames. This algorithm has a huge range of potential applications. It may prove to be ideally suited for the study of crack propagation and material deformation in real-world applications, as it has the potential to become a cheap, simple yet accurate solution (McCormick and Lord 2010).

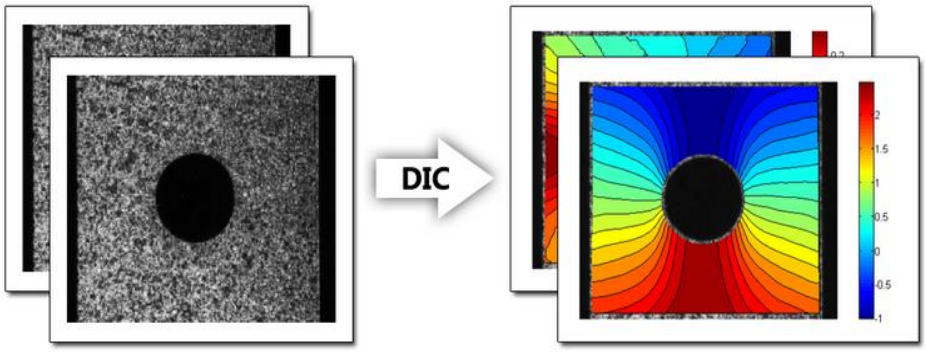


Figure 3. Example of DIC inputs and outputs ("DIC" Ncorr. Web. 20 Mar. 2015.)

1) Correlation criterion

➢ Zero-mean normalized cross-correlation (ZNCC) criteria, which is insensitive to image scale and illumination variance (Taniguchi et al. 2013). The ZNCC criterion is described as bellow.

$$C(p') = 1 - \frac{\sum_{x=-M}^{M}\sum_{y=-M}^{M}[f(x,y) - fm][g(x',y') - gm]}{\sqrt{\sum_{x=-M}^{M}\sum_{y=-M}^{M}[f(x,y) - fm]^2}\sqrt{\sum_{x=-M}^{M}\sum_{y=-M}^{M}[g(x',y') - gm]^2}} \quad \text{(Eq. 1)}$$

Where, f(x, y) and g(x', y') are the corresponding gray values of the deformed reference subsets; x and y are the point coordinates at the center of the reference subset coordinate systems; x' and y' are mapped coordinates of the point (x, y), respectively. $fm = 1/(2M+1)^2 \sum_{x=-M}^{M}\sum_{y=-M}^{M} f(x,y)$ and $gm = 1/(2M+1)^2 \sum_{x=-M}^{M}\sum_{y=-M}^{M} g(x',y')$ are the average gray values of the points in the two subsets; p' is described as the deformation vector, which reveals the relationships between the coordinates (x, y) and coordinates (x', y').

Afterwards, the point (x, y) in the reference subset after deformation is

represented by the first- order shape function shown as below:

$$x' = x + u + \frac{\partial u}{\partial x}x + \frac{\partial u}{\partial y}y \qquad \text{(Eq. 2)}$$

$$y' = y + v + \frac{\partial v}{\partial x}x + \frac{\partial v}{\partial y}y \qquad \text{(Eq. 3)}$$

Where, u and v are the displacement components of reference subset center on x and y directions; the expressions óu/óx, óu/óy, óv/óx and óv/óy are the displacement gradient components; p'= [u, v, óu/óx, óu/óy, óv/óx] is calculated as the corresponding deformation parameter vector (Zhao et al. 2012).

The first-order shape function presented above can be used to handle the situations of translation, rotation, shear, strains and their combinations, and all necessary deflection and deformation information for the measurement in our research can be recovered.

2) Providing initial guess

Gauss-Newton method is applied to search the roots of a function to deal with the issue that an analytic solution may not be available. The issue can be addressed by obtaining the roots of the derivative of a function. Furthermore, its generalization to multivariate optimization can be achieved by replacing the derivative with the gradient, afterwards determining where the norm of the gradient converges to zero (Marquardt 1963).

3) Region of interest (ROI) for DIC

When applying the Digital Image correlation (DIC) algorithm, we need specify a region of interest for the algorithm. Region of interest, namely, is the image region that

we are interested to obtain the feature points. Normally, this region is specified through given the top left and bottom right pixel coordinates of the image that needs to be processed.

The figure below shows the example of specifying the region of interest for the digital image correlation algorithm in our experiment. In this example, we specified the coordinates of the top left point to be (800, 350), and the coordinates of the bottom right point is (1100, 650). Then, actually the width of region of interest is 300 pixels (bottom right X coordinate minus top left X coordinate), and the height of region of interest is 300 pixels (bottom right Y coordinate minus top left Y coordinate). Normally, these coordinate values can be determined by the specific position where the target with interest exactly located in the image.

```
98      //Specify the region of interest(ROI)
99      int TopLeftX=800;
100     int TopLeftY=350;
101     int BottomRightX=1100;
102     int BottomRightY=650;
103     int StepSize=50;
104     int runcount=1;
```

Figure 4. Specification of region of interest (ROI)

2.2.4 Interpolation

Interpolation procedure is needed because after obtaining the sparse displacement field the dense displacement field is generated by interpolating those sparse displacement values. Bilinear interpolation method is adopted in our experiment after getting the sparse deflection/displacement values.

Bilinear interpolation is one of the most typical image interpolation methods in image processing. The principle idea of this algorithm is to compute the linear interpolation values for the target points based on the interpolation function f(x). These interpolation values can be calculated according to the 4 nearest points around the target point by applying the interpolation functions for X and Y directions respectively.

The following figure briefly shows the procedures of bilinear interpolation method:

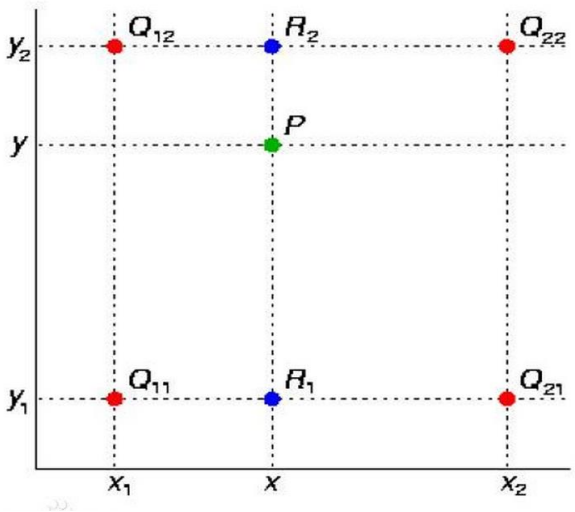


Figure 5. Linear interpolation procedure ("Bilinear Interpolation", Baike. Web. 25 Mar. 2015.)

The green point in the figure is the target point that we need to obtain its interpolation value. The red points are the 4 nearest points around the target point. Then, assume the coordinates for the 4 nearest points are $Q_{11} = (x_1, y_1)$, $Q_{12} = (x_1, y_2)$, $Q_{21} = (x_2, y_1)$ and $Q_{22} = (x_2, y_2)$. We need to calculate the interpolation value for the target point $P = (x, y)$.

The first step, linear interpolation will be conducted for X direction. In specific,

we need get the value for the blue points in the figure. The value of blue points will be calculated based on the two red points around them by given specific metric for the interpolation function. In our experiment, the metric is specified to be Euclidian distance. In other words, the distances between the blue point and its two nearest red points will assigned to be weights when computing the interpolation value.

The following formulas are given:

$$f(R_1) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21}) \quad \text{where} \quad R_1 = (x, y_1) \tag{Eq. 4}$$

$$f(R_2) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22}) \quad \text{where} \quad R_2 = (x, y_2) \tag{Eq. 5}$$

Where, $f(Q_{11})$, $f(Q_{12})$, $f(Q_{21})$ and $f(Q_{22})$ are already known values of the red points. The values of x, x_1 and x_2 are coordinates along the X direction of the image. In this way, the interpolation value for the point R_1 and R_2, which are the blue points shown in the above figure.

Then, the second step is based on the calcualted interpolation vaules of R_1 and R_2 (blue poins). Linear interpolitaon procedure will be conducted once again for the green point. This time, the blue points replece the red poinst in the previous step.

The following formula is used:

$$f(P) \approx \frac{y_2 - y}{y_2 - y_1} f(R_1) + \frac{y - y_1}{y_2 - y_1} f(R_2) \tag{Eq. 6}$$

Where, $f(R_1)$ and $f(R_2)$ are already calculated values from the step one. The values of y, y_1 and y_2 are coordinates along the Y direction of the image. Then, the f(P) is the interpolation value that we need to calculate for the target point P (x, y).

Compared with nearest interpolation algorithm, the processing performance of bilinear interpolation is much better, while compared with bi-cubic interpolation algorithm, the running efficiency of bilinear interpolation algorithm is better (Acharya and Tsai 2007). It is adopted after trade-off between the processing performance and running efficiency in our experiments.

2.2.5 In-plane displacement and strain computation

This procedure can be performed by specifying a region of interest (ROI) and then the displacement data is determined in a grid within the ROI. Afterwards, the displacements data can be either reduced or interpolated to generate a "continuous" displacement field. In our research we apply bilinear interpolation procedure introduced above to obtain dense displacement field.

After obtaining the dense displacement field, in-plane strains of the target can calculated through computing the derivatives of displacement field. Following procedures were applied to compute the strains.

- ***Average strain computing***

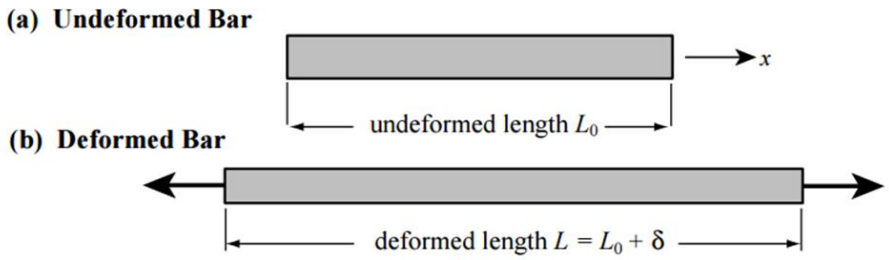


Figure 6. Undeformed and deformed bar configurations to illustrate average strain computation (“Strain”. IAST.Lect04. Web. 26 Mar. 2015)

For an un-loaded bar of length L_0 aligned with the X axis, as shown in the above figure. Regarding the un-deformed bar, also called initial reference or original configuration, the strains of the bar are taken to be zero. This bar is then pulled by applying an axial force. The un-deformed and deformed configurations are shown offset for visualization convenience. In this new configuration, also called deformed or current configuration, the bar's length becomes $L = L_0 + \delta$, where the elongation of the bar is $\delta = L - L_0$. Then, the average axial strain over the whole bar is defined as:

$$\epsilon_{av}^{bar} \stackrel{\text{def}}{=} \frac{L - L_0}{L_{ref}} = \frac{\delta}{L_{ref}} \qquad \text{(Eq. 7)}$$

Where, L_{ref} is the reference length selected for the strain computation. The two conventional choices are $L_{ref} = L_0$ for Lagrangian strains, and $L_{ref} = L$ for Eulerian strains. The former is that commonly applied in solid mechanics and structures. The latter one is usually used fluid mechanics. In our experiment, we specified the L_{ref} to be L_0 for Lagrangian strains.

- ***Point Strain Computing***

The strain at a point is obtained by a limit process. For the un-deformed bar, we mark two points: P and Q separated by a small but finite distance x, as shown in the figure below. Then, the bar is pulled to the deformed configuration as shown in the Figure 7. (b).

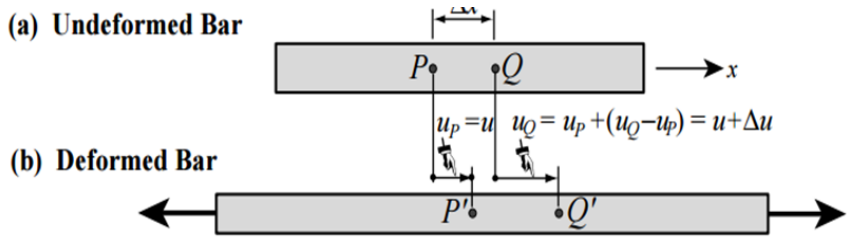


Figure 7. Undeformed and deformed bar configuration for point strain computation

("Strain". IAST.Lect04. Web. 26 Mar. 2015)

The P and Q points have moved to P' and Q'. The axial displacements are $u_P=u$ and $u_Q=u_P+(u_Q-u_P)=u+\Delta u$, respectively. The strain at P can be obtained through taking the limit of the average strain over x as this distance tends to zero. The computation formula is given as:

$$\epsilon_P \overset{\text{def}}{=} \lim_{\Delta x \to 0} \frac{(u + \Delta u) - u}{\Delta x} = \lim_{\Delta x \to 0} \frac{\Delta u}{\Delta x} = \frac{du}{dx} \qquad \text{(Eq. 8)}$$

This formula is also called the strain-displacement equation. It can be applied to compute strains directly by differentiation of the displacement. The formula is shown as below:

$$\epsilon_{xx} = \frac{\partial u}{\partial x} \qquad \text{(Eq. 9)}$$

Where, u is the displacement value of the X direction. ϵ_{XX} is the strain along the same direction. This formula can be used for computing the point strains for each direction of the images.

CHAPTER 3: PROBLEM STATEMENT AND OBJECTIVE

Figure 8 below shows how the feature detection and matching procedure is

applied to measure the deflections and strains.

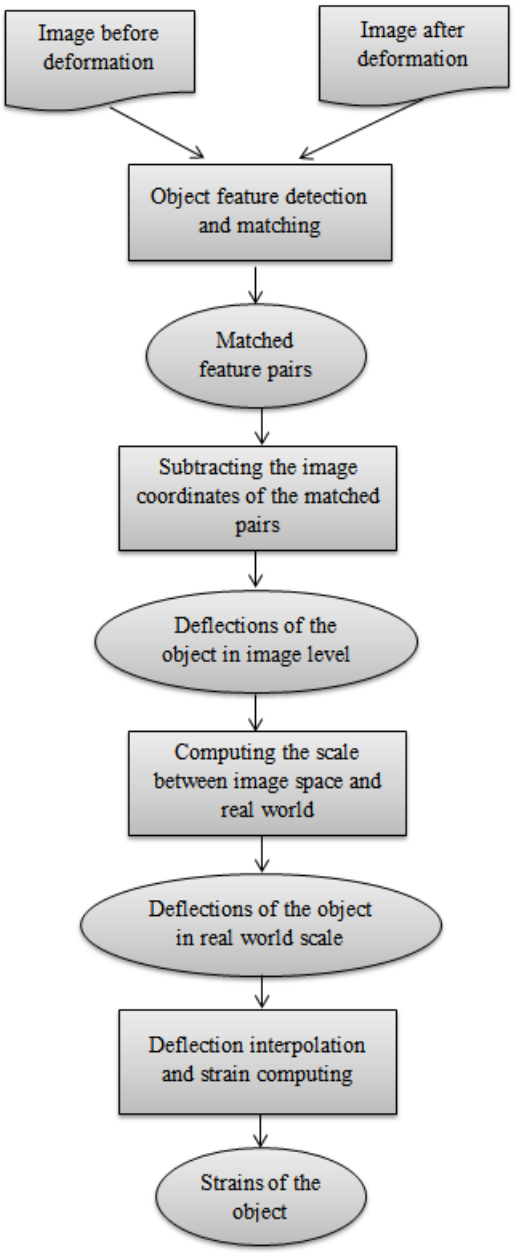


Figure 8. Framework of measuring deflections and strains from images

Different algorithms might be applied in the feature detection and matching procedure, however, the performance of the algorithms has not been compared for

measuring in-plane deflections and strains of civil infrastructures in terms of accuracy and efficiency. To identify the optimal feature detection and matching algorithm for the visual sensing-based method, we need evaluate the three selected feature detection and matching algorithms DIC, SIFT and SURF.

As also introduced in Chapter 2, there are two types of feature detection and matching algorithms can be applied for the measuring task, The first type is feature-based method (SIFT, SURF, etc.). The reason why we selected SIFT and SURF algorithms to test in our experiments is because previous relevant research has revealed that the SIFT and SURF detectors and descriptors have priority over other detectors and descriptors (Zhu and Davari 2014), such as HOG (Histogram of Oriented Gradients) and GLOH (Gradient Localization Oriented Histogram) etc. Another type of algorithm is pixel-based method digital image correlation (DIC). DIC algorithm has shown great potential to measure deformations in mechanical field (Zhao et al. 2012). However, the performance of this algorithm in civil infrastructure applications still has not be identified due to the distinct differences between civil engineering and mechanical fields. Civil infrastructure applications, such as tunnels or dams, are usually in much greater scales than that of mechanical applications. This also may cause the uncertainty of applying DIC algorithm in civil engineering field. Besides, speckle patterns are usually used for DIC algorithm in mechanical applications (Bornert et al, 2009); whereas the speckle patterns are actually not appropriate to be utilized for civil infrastructure applications. As a result, to further identify whether DIC algorithm can still reveal great potential to be utilized to measure deflections and strains of civil infrastructures, we need apply and compare the DIC

algorithm with the settings suitable for civil infrastructure applications with feature-based matching algorithms, such as SIFT and SURF.

Specifically, all the three algorithms can be used in detecting and matching variance that occurs on the surfaces of structure members. However, which one is the most appropriate for implementation of measuring the dynamic deflections and strains of civil infrastructures is unknown in terms of accuracy and efficiency. In order to address this problem, firstly, the algorithms should be implemented with suitable parameter setting for measuring civil infrastructures. After the algorithm implementing procedure, the accuracy and efficiency performance of three algorithms need to be evaluated in a detailed manner. To fill the gap, a series of experiments are designed through using both synthetic images and industrial images as the dataset to test the three algorithms. Next, the accuracy and efficiency comparison work is to be conducted in order to determine the optimal algorithm.

In addition, for the purpose of validating whether the determined optimal feature detection and matching algorithm is applicable to measure deflections and strains for real world civil infrastructures, real world scenario testing needs to be conducted based on the optimal algorithm (it could be determine in the evaluation work that the DIC algorithm is the optimal feature detection and matching algorithm for our case). Therefore, the second section of our research attempts to apply the DIC-based algorithm to measure deflections and strains for real world scenarios and LVDT experiment is designed and conducted to accomplish the research goal (the experimental data is collected in the concrete lab at West Virginia University). In detailed discussion and analysis work is also conducted to

further validate the applicability and its accuracy performance of the proposed DUC-based visual sensing method for measuring deflections and strains of civil infrastructure applications.

CHAPTER 4: EXPERIMENTAL DESIGN

In this section, firstly, a series of experiments are designed and carried out in order to test the performance of the three presented feature detection and matching algorithms. These experiments can be divided into two groups: 1) synthetic image testing; 2) real world scenario testing. For group 1, three images of civil structures are first selected. Then, their corresponding synthetic images include translation subset, rotation subset; illumination changes subset and deformation subset are generated to provide the ground truths. Image processing programs are designed and implemented in MATLAB platform to achieve the translation, rotation, changing illumination and deformation processes.

4.1 Comparison experiments for evaluating feature matching algorithms

➢ ***Overview***

The following flow chart presents the overall procedure of conducting the comparison experiment. As shown in the figure below, digital camera Canon 5D mark III is employed in our experiment to capture experimental dataset. In the figure, the first part is about data collection, namely, applying our digital/industrial cameras to capture a series of images frames of the targets. The second part is data processing, including image feature detection and matching, ground truth generation. The third part is about

experimental results.

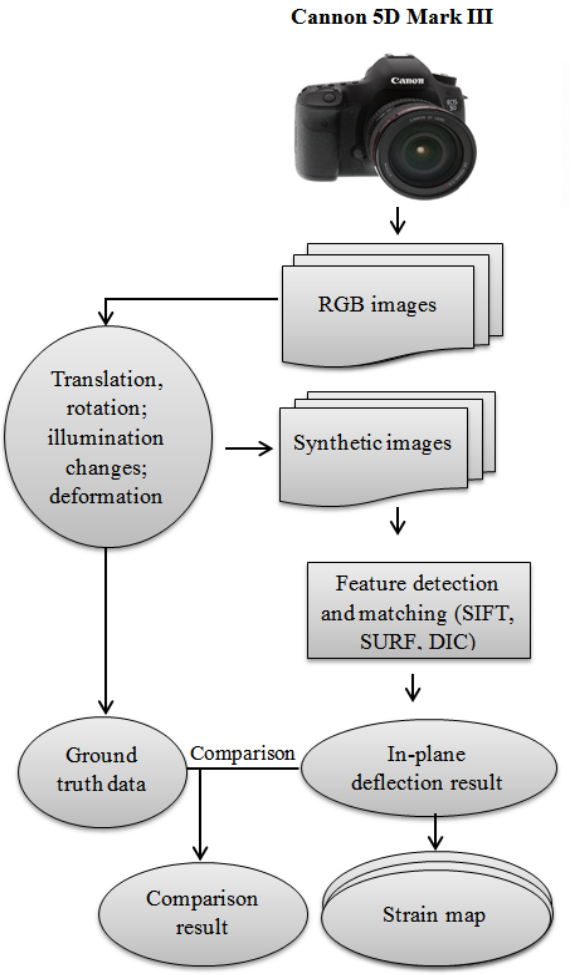


Figure 9. Framework of conducting comparison experiment for the three algorithms

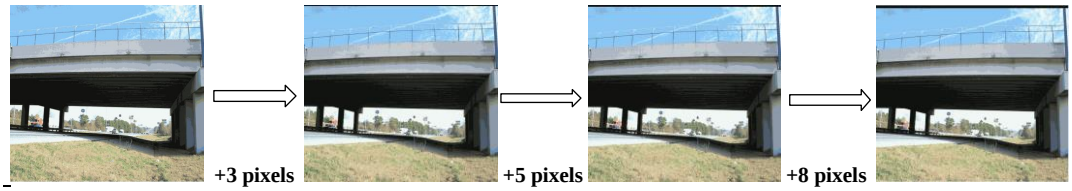
4.1.1 Original digital image dataset

Three images with different scenarios were selected in our research as shown in Figure 10. Image (a) is a part of a concrete bridge in the field; image (b) is a constructed mock-up bridge in the lab; image (c) is an in-building structure at a construction site.

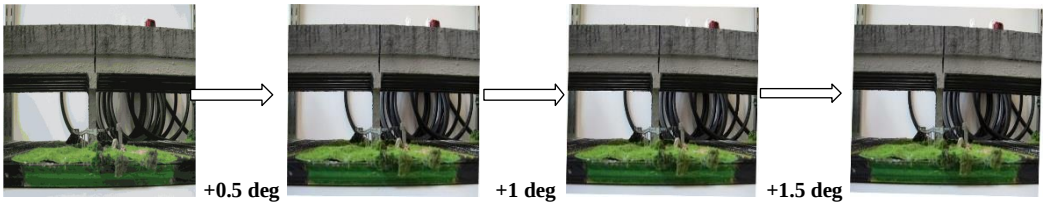
Figure 10. Original image dataset

4.1.2 Synthetic image dataset

The translation subset is designed to test the performance of the algorithms to deal with the in-plane translation (deflection) of the scenarios. The images of translation subset are shown in Figure 11.(a). Rotation subset is designed to test the performance of the algorithms when facing with the in-plane rotation of the scenarios. Figure 11.(b) shows the images of rotation subset. Furthermore, Illumination changes subset is for assessing the algorithm performance in dealing with illumination changes when the pictures are captured under different lighting condition. The illumination changed images is generated by operating Xnview image software based on the synthetic images with 5 pixels deflection. The illumination subset is important since the algorithms are expected to have the ability to measure the scenarios under different light conditions during the daytime. These illumination changes subset images are shown in Figure 11.(c).



(a) Vertical deflection: respectively 3 pixels, 5 pixels 8 pixels deflection of original image



(b) Clockwise rotation: respectively 0.5, 1 and 1.5 degree rotation of original image

(c) Illumination changes: respectively 50 Lex less illumination and 50 Lex more illumination images based on original image

Figure 11. Synthetic Image dataset

The experimental subsets presented above are all about linear transformations of the original dataset. However, as well known, in the real world the observed targets cannot always be as preforming linear transformation. Therefore, the following experimental subset is designed to deal with non-linear transformation scenarios.

The synthetic deformed images are generated by the specified deformation functions:

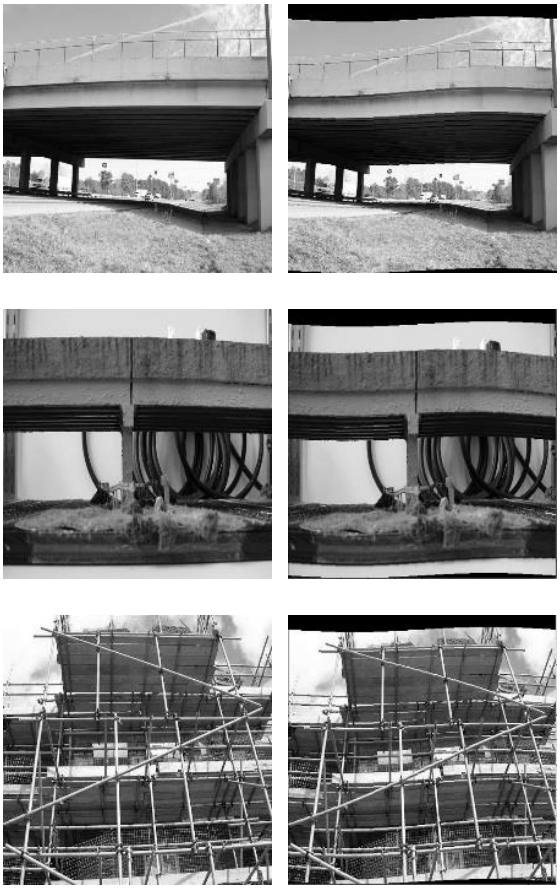
$$\begin{cases} u = x; \\ v = y + [\mu * \sin(2 * Pi * x) / (h - 1)]; \end{cases} \quad \text{(Eq. 10)}$$

$$\begin{cases} u' = x + [\mu * \sin(2 * Pi * y) / (w - 1)]; \\ v' = y; \end{cases} \quad \text{(Eq. 11)}$$

The functions above are employed to add sinusoidal deformation to the original dataset. Where, function (1) is used to add vertical sinusoidal deformation; similarly,

function (2) adds horizontal sinusoidal deformation to the original images.

In this functions, (x, y) are the coordinates of the original image points and (u, v), (u′, v′) are the coordinates of corresponding points in the deformed image. The deformation scale factor μ is set to be 5.0 in this experiment. Pi is the circumference ratio, and h and w are the height and width of the images. The deformed synthetic images are shown in Figure 12.



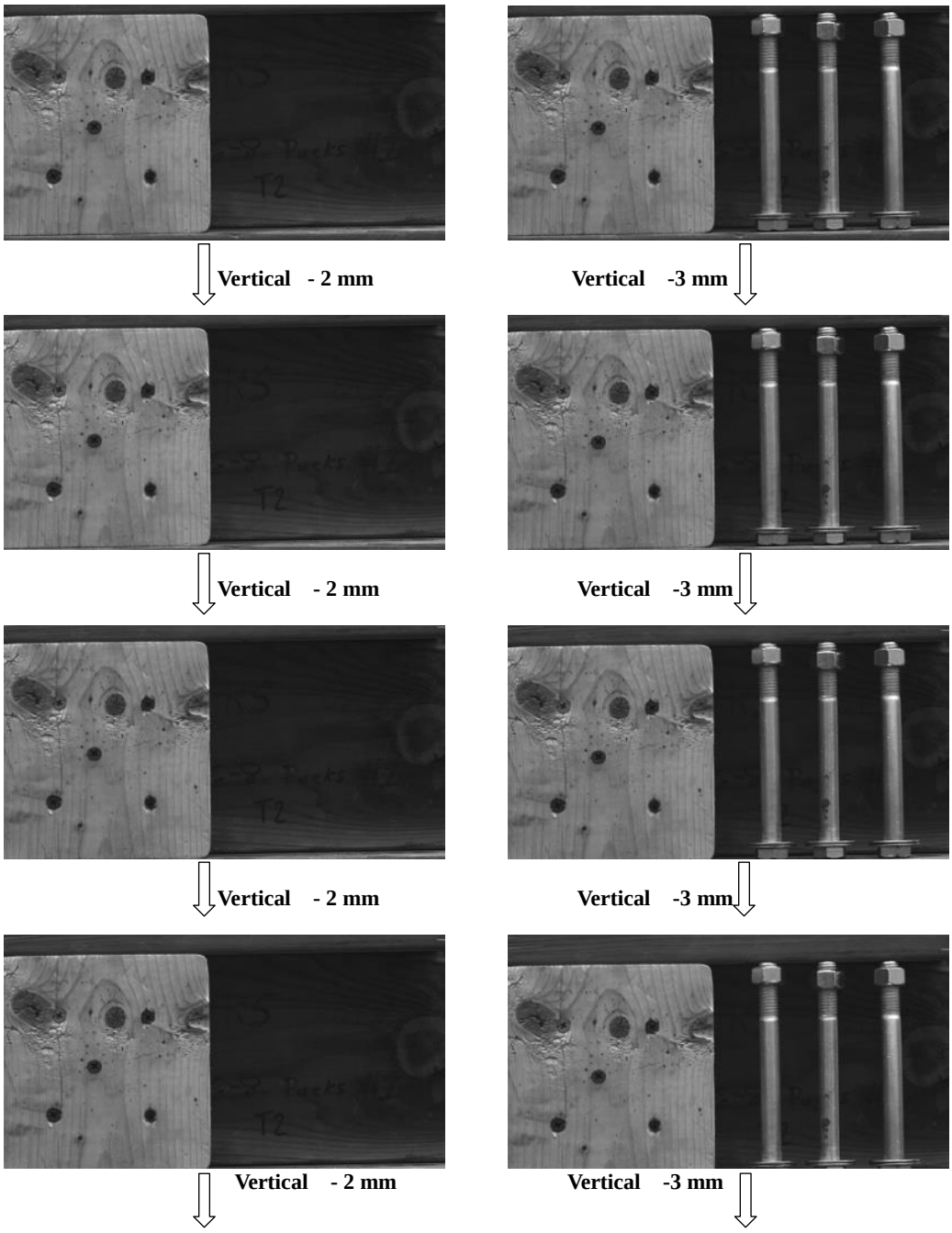
(a) Original images (b) Deformed images

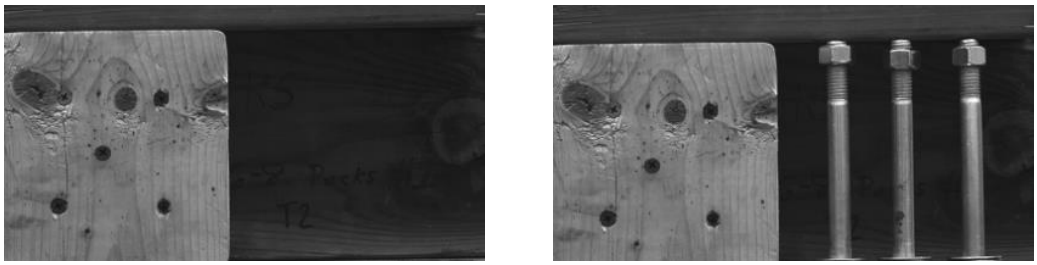
Figure 12. Sinusoidal function deformed image dataset

4.1.3 Real world scenarios testing and deflection measuring

In this experiment, Manta G-233-B industrial camera with a 2/3 inch COMS

sensor and a 50 mm fixed focal length lens were used. We captured two groups of indoor scenario images under operating the servo-hydraulic fatigue testing machine (INSTRON 8501) in our concrete laboratory. The two groups of images captured by the industrial camera are shown in Figure 13.





(a) Group 1 (b) Group 2

Figure 13. Industrial camera images of real world scenarios in concrete lab

4.2 Measuring deflections and strains for real world scenarios

➢ ***Overview***

In this section, several experiments are designed to compute the deflections and strains for real world scenarios by applying images correlation-based algorithm. After computing the deflections and strains, these experimental results will then be used to compare with the ground truth data. Then the performance of our method can be evaluated in this way.

The following flow chart presents the overall procedure of the experiment. As shown in the figure below, industrial camera Manta G-223B is employed in our experiment to capture the experimental dataset. In the figure, the first part is data collection. The second part is data processing, including image feature detection and matching, ground truth recording and the third part is about experimental results.

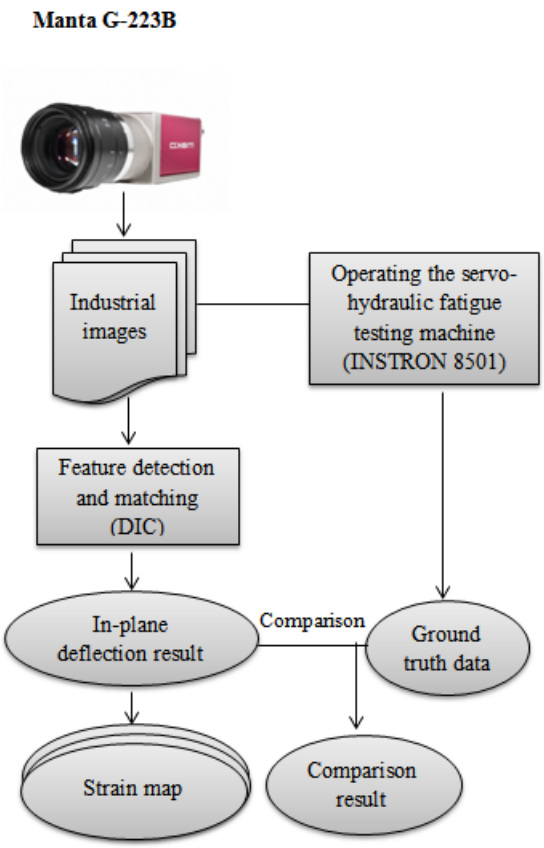


Figure 14. Framework of measuring deflections and strains for real world scenarios

4.2.1 Data collection

To collect the experimental data, we conducted LVDT (Linear Variable Differential Transformer, also called differential transformer, which is a device typically used for measuring linear displacement) experiment with INSTRON concrete compression machine in the structural/concrete lab at West Virginia University. The following figure is an image taking when we are conducting the LVDT experiment in the concrete lab.

Figure 15. Industrial camera set up and LVDT configuration

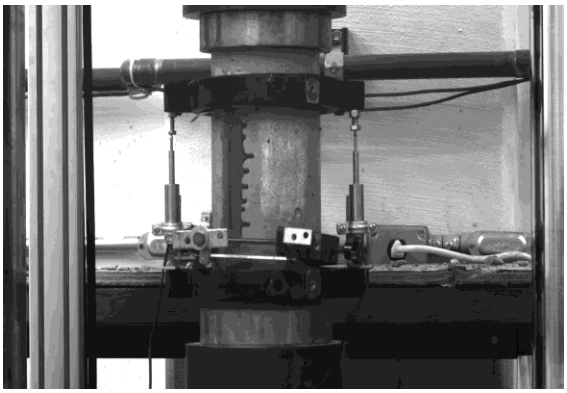
The data collection procedures mainly includes industrial camera (Manta G-223B) set up and configuration, LVDT installation and recording configuration, and INSTRON compression machine configuration and operation as well. The image data captured by the industrial camera will be instantly transmitted and stored in the laptop's hard drive.

Figure below is an image of the testing concrete sample with LVDT devices installed on it. This concrete testing sample was made by technician from WVU structural group. The LVDT devices (two LVDT devices used for the concrete sample, as shown in the figure below, left and right sides of the sample were installed one device, respectively) have been installed onto the testing sample, and then the concrete sample was placed on the INSTRON hydraulic machine for compression experiment.

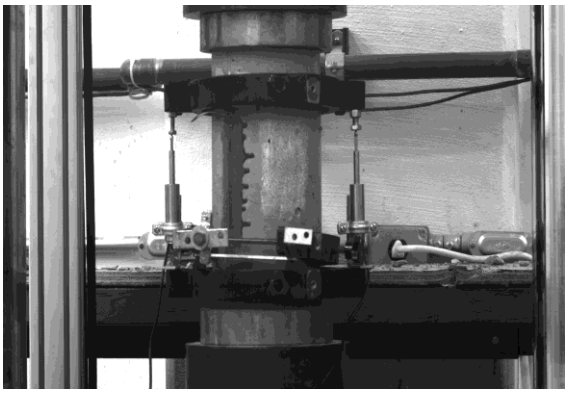
Figure 16. Experimental specimen and LVDT equipment

4.2.2 Industrial image dataset

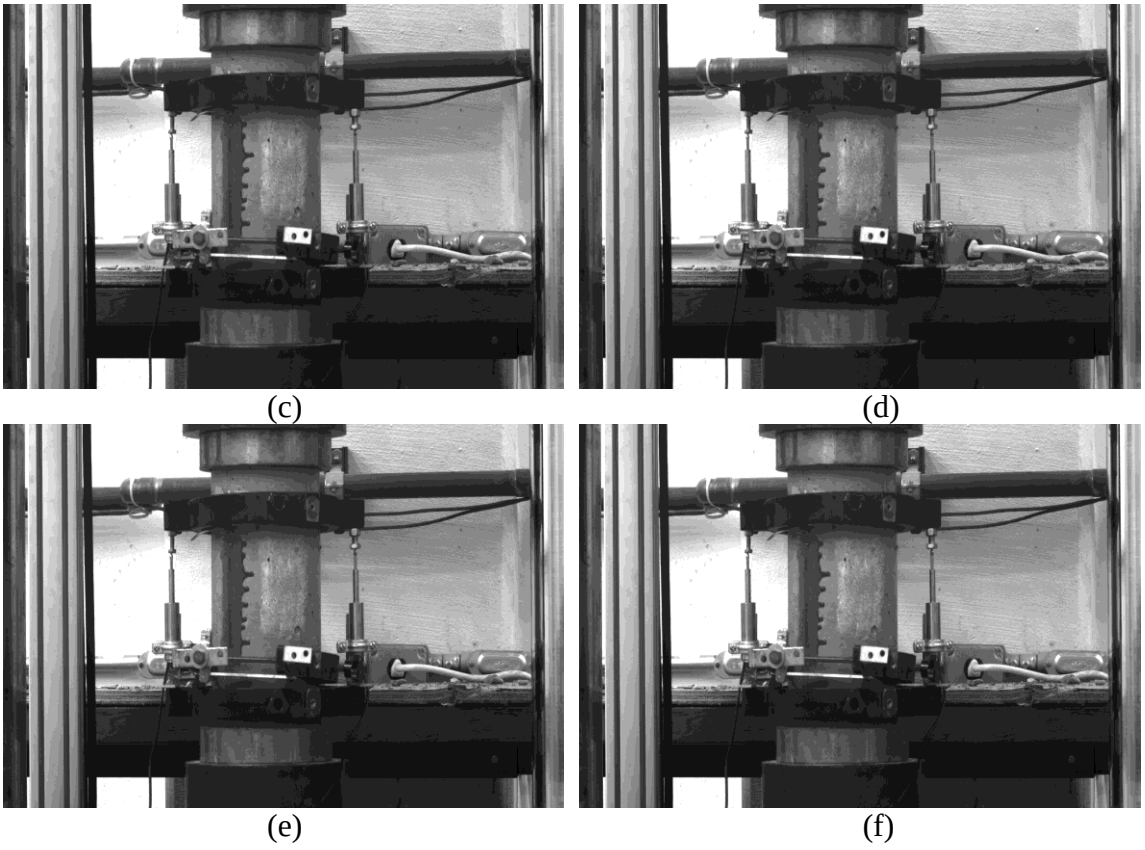
The images of concrete testing sample as shown in the figure blow are single-view images captured by the industrial camera during the experiments. From image (a) to image (f), the concrete sample was compressed by the machine by specifying 0.0008 inch displacement between each two neighboring images.



(a)



(b)



(c) (d)

(e) (f)

Figure 17. Real world industrial image dataset of concrete sample

4.2.3 Data pre-processing

After obtaining the industrial image dataset shown in last section, firstly, we applied the image correlation-based algorithm to detect and matching the corresponding features in the image frames. Then, based on the positions of the feature points, deflection/displacement can be calculated for each pair of corresponding feature points.

The following figure is the operation interface of the image correlation-based algorithm. It is built upon Dr. Zhao's previous work (Zhao et al. 2012). In this algorithm, different region of interest (ROI) and related local parameters that are suitable for processing the images of civil infrastructure applications are adjusted in the experiments.

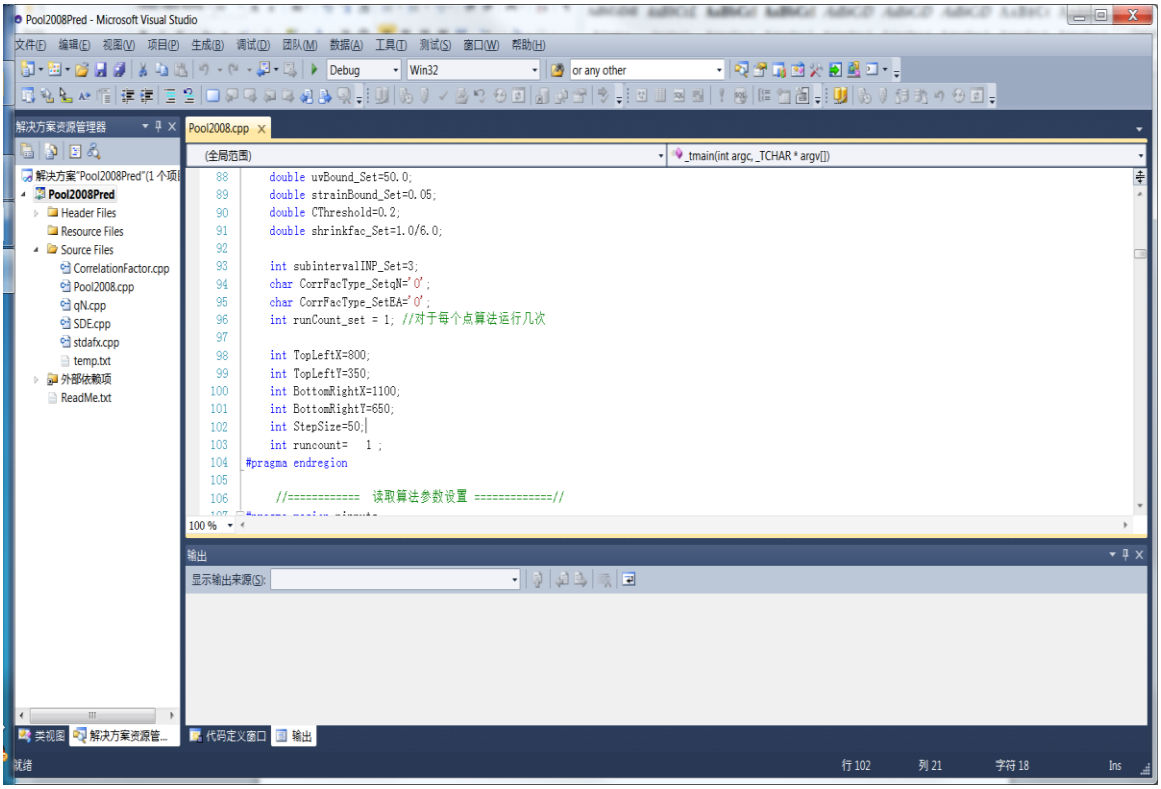


Figure 18. Image correlation-based algorithm operation interface

The figure below shows the processing window of the image correalation-based algorithm. From the figure, we can see it is a win32 console program, in which totally 49 points of interest (POI) are specified for the current processing precedure. The algorithm attempts to search the correspodning feature poins for each point of interest iteratively. Corresponding features, namely, is the feature points in defferent images that correspons to the same object points in the real world.

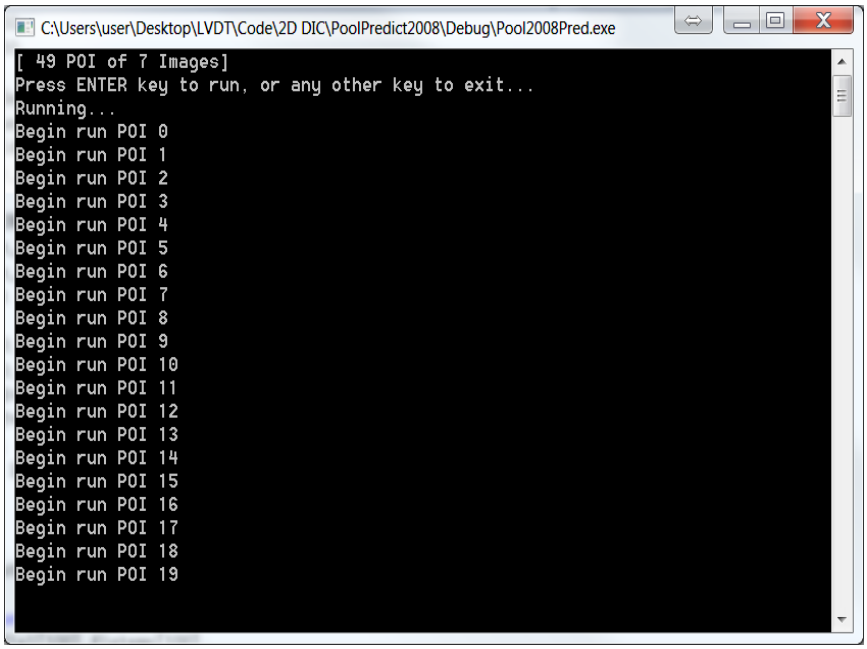


Figure 19. Processing window of the image correlation-based algorithm

4.2.4 In-plane displacement calculation

After we get the corresponding features for the image frames, the deflection/ displacement values can be calculated based on the positions of corresponding feature pairs located in defferent image frames.

As presented in section 2.2.3, dispalcement values can be calculated by substracting the image coordinates of corresponding feature points in the images. Here, we obtain the displaceent results are pixel displacement of the object in the image coordinate system. However, after specifying the camera capture distance and foucal length of camera lens, the image-scale pixel displacement can be reclaculated in real world scale.

The figure shown below presents the Matlab program implemented to calculate the sparse displacement field based on the processing result obtained by the image correaltion-based algorithm.

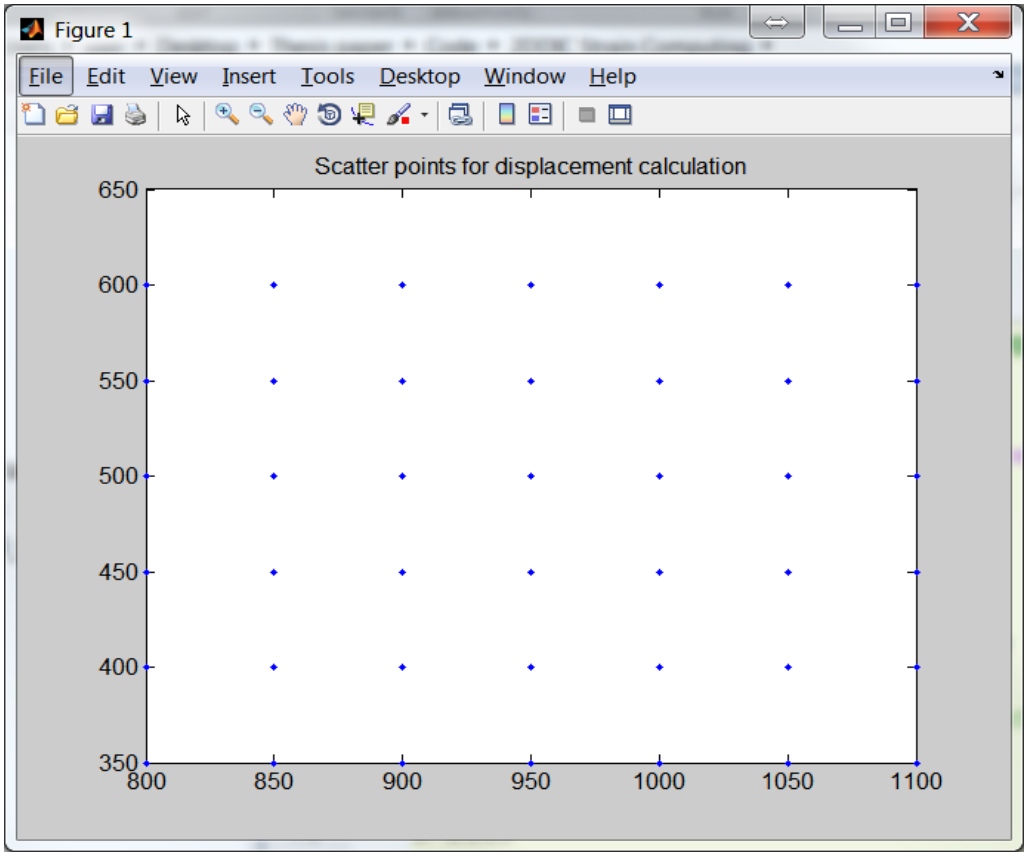


Figure 20. Scatter points for displacement calculation

The scatter points shown in the above figure are the previously introduced points of interest (POI). Thus, after the image correlation processing procedure, we got the corresponding feature points in deffrent image for those POIs.

Then, the displacement values can be calculated based on the corresponding features. The calculated results can be visualized as displacement arrow map. As shown in the figure below.

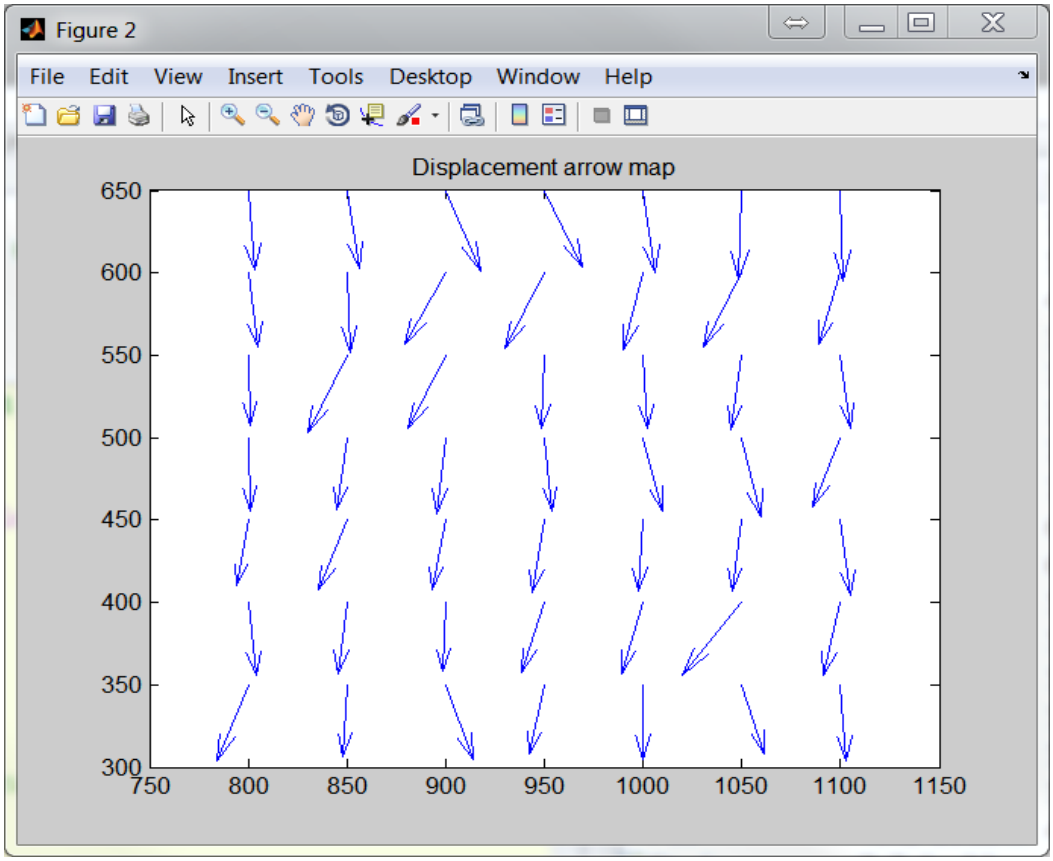


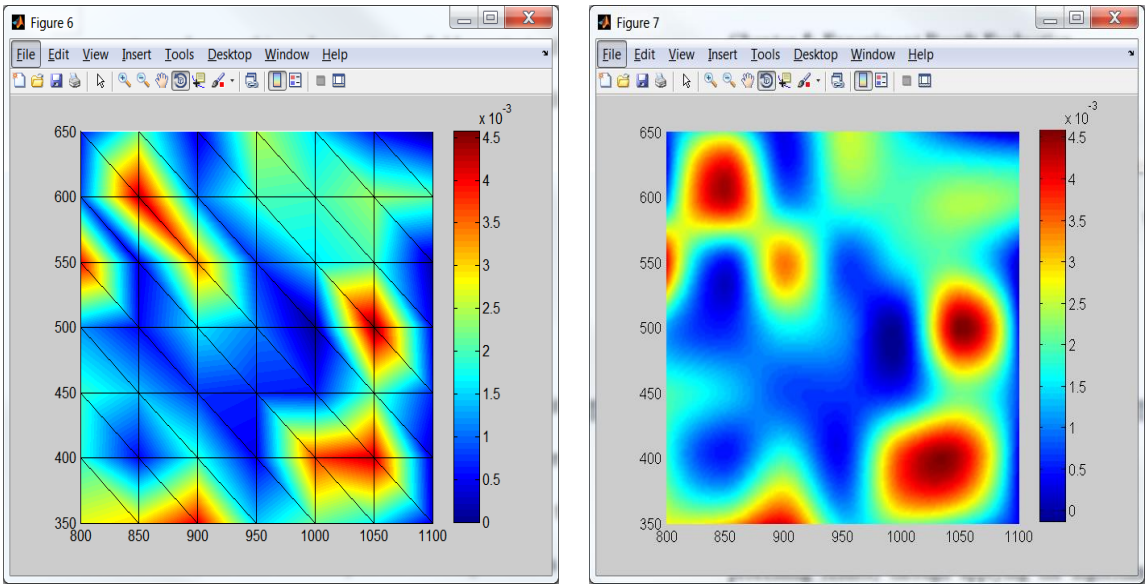
Figure 21. Displacement arrow map

the origins of the arrows are the points of interest (POIs), the direction of the arrows refer to the directions where dispalcement happening, and the lengths of the arrows are displacement values. Therefire, this arrow map actually is a vector diagram for representing in-plane displacement field, in which the arrow derections are the directions of displacement vectors and the leagths of the arrows represent the quantity value of displacement vectors.

4.2.5 In-plane strain computation

After obtaining the sparse displacement field, actually sparse in-plane can also be

computed based on those displacement values. However, in order to generate much more accurate and smoothing dense strain field, optimized interpolation method is implemented and employed in the strain computation procedure. The figure below shows the intuitive difference before and after applying the interpolation algorithms for the strain optimization work.



(a) Before (b) After

Figure 22. Strain maps before and after applying interpolation algorithm

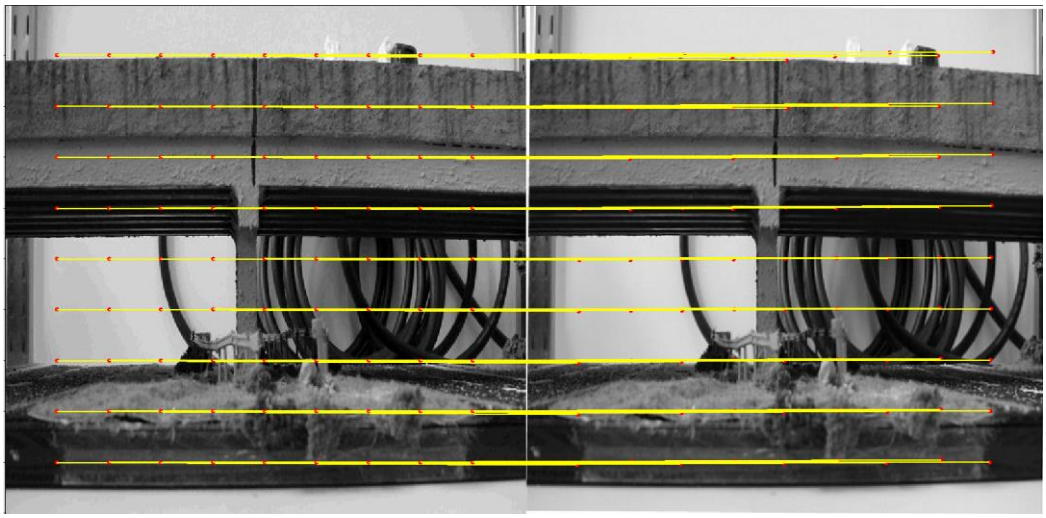
As we can see in the above figure, the strain map getted before the interpolation procesure has sharp edges, which means the strain values in the map is not changing smoothly. In this case, the strain values obtained for the points other than the points of interest (POIs) are not accruate. On the other hand, the strain map generated after the interpolation procedure has very smoothing edges, in outher words, the strain values in the map change gradually and continuously, which is more reasonabe when accounting the real wrorld strain disctributions.

CHAPTER 5: EXPERIMENT RESULT AND EVALUATION

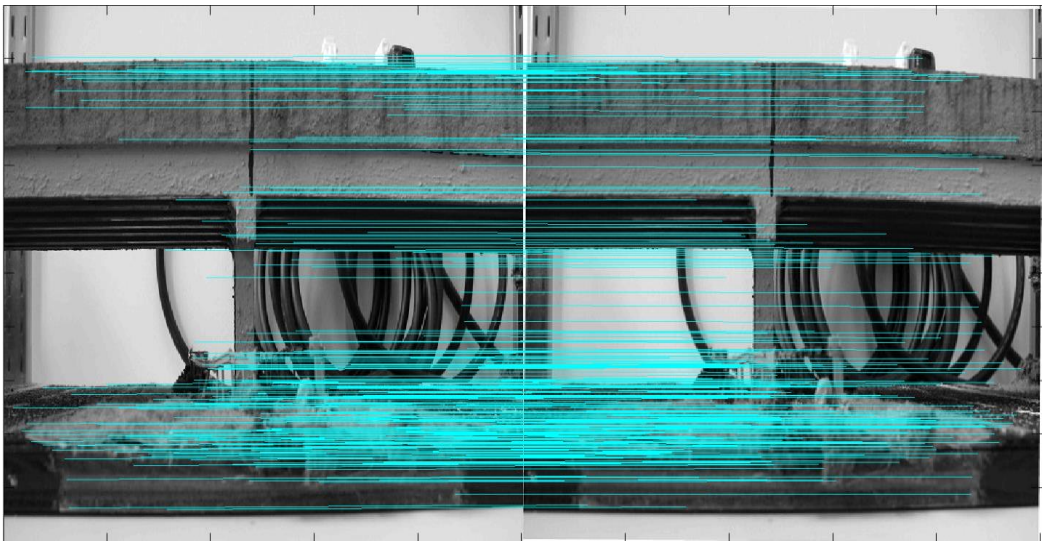
5.1 Accuracy comparison of digital image groups

The accuracy criterion for comparing the three algorithms is based on calculating the absolute differences (errors) between the measured transformed values (experimental processing results) through applying the algorithms and their corresponding ground truths. Besides, the time efficiency of the three algorithms is also evaluated by recording and comparing the algorithms' running time when processing different groups of testing scenarios.

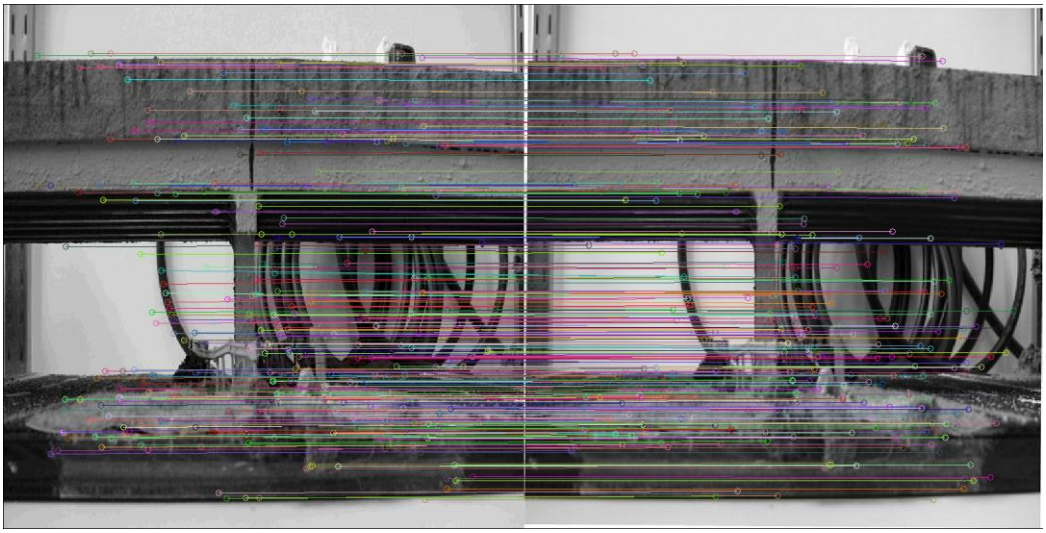
The figure below shows the feature detection and matching results of the three algorithms, from which we can also observe that the region of interest (ROI) can be specified in the DIC algorithm. While the SIFT and SURF algorithms only automatically selected a series of random features in the images.



(a) DIC processing result



(b) SIFT processing result



(c) SURF processing result

Figure 23. Processing results of DIC, SIFT and SURF algorithms for rotation group

5.1.1 Measurement accuracies of linear deflected scenarios

To compare the accuracy performance of the algorithms, this section shows the statistical results of measuring accuracy. The following table shows the accuracy of the three algorithms when processing linear deflected scenarios, including translation, rotation and illumination changing groups. The data in the table is error percentages that

are calculated through dividing absolute errors by the corresponding ground truth.

Table 2. Error percentages of different image scenes: (a) Translation; (b) Rotation; (c) Illumination changes

Translation (%)	3 pixels			5 pixels			8 pixels		
	Scene 1	Scene 2	Scene 3	Scene 1	Scene 2	Scene 3	Scene 1	Scene 2	Scene 3
DIC	0.0311	0.0003	0.5587	0.0045	0.0001	0.0124	0.0362	0.0259	0.0339
SIFT	0.1252	0.2914	1.4079	0.5825	0.1743	0.8446	0.4822	0.0890	0.5279
SURF	12.5417	8.0829	8.8976	5.6888	4.3751	5.0000	0.5072	0.2331	0.0586

(a)

Rotation (%)	0.5 degree			1 degree			1.5 degree		
	Scene 1	Scene 2	Scene 3	Scene 1	Scene 2	Scene 3	Scene 1	Scene 2	Scene 3
DIC	17.6293	15.9105	16.7992	17.545	15.1788	16.3858	16.7685	14.8734	15.3340
SIFT	25.3056	24.3408	25.8251	25.397	23.9278	24.3075	24.5719	23.1111	23.2874
SURF	24.8862	25.0558	25.5584	24.0299	22.6295	24.1035	23.8335	22.0336	22.6595

(b)

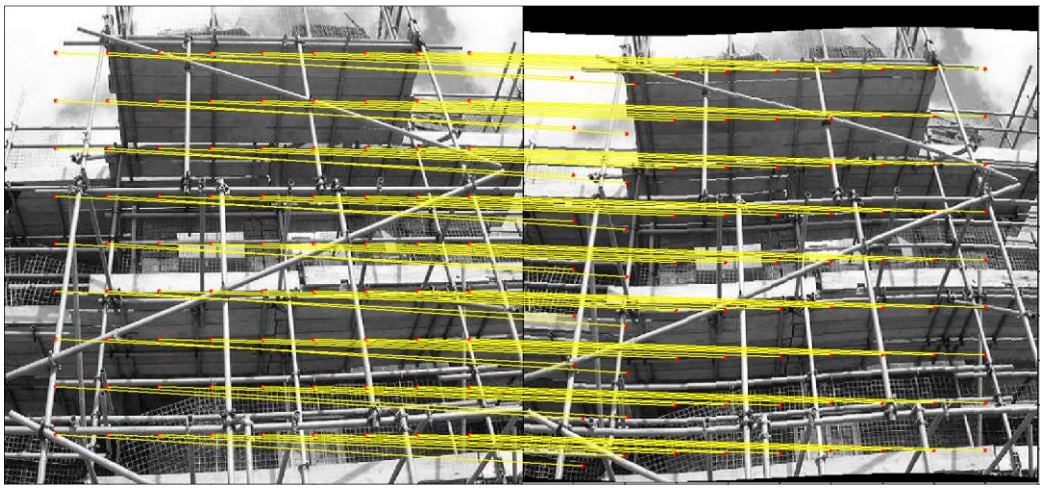
Illumination (%)	Less-Illumination			Normal-Illumination			More-Illumination		
	Scene 1	Scene 2	Scene 3	Scene 1	Scene 2	Scene 3	Scene 1	Scene 2	Scene 3
DIC	0.0043	0.0813	0.0023	0.0045	0.0001	0.0124	0.0045	0.0001	0.014
SIFT	0.5825	0.1803	0.8446	0.5825	0.1744	0.8446	0.5825	0.1803	0.8446
SURF	6.1731	4.3751	5.6578	5.6888	4.3751	5.0000	5.7403	4.3751	5.219

(c)

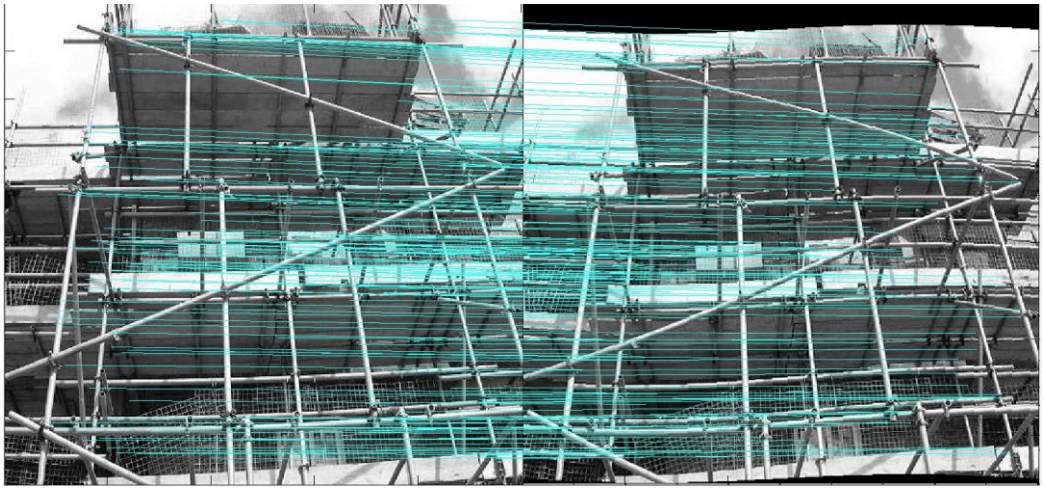
5.1.2 Measurement accuracies of non-linear deformed scenarios

Above three groups of experiments are designed for linear deflected scenarios. However, deflections of civil infrastructure applications in our real world can not always be considered as scenraios only with linear deflcetions. Therefore, anohter group of experimetn is conducted to testing the performace of processing non-linear case. Here we use the deormed image dataset, which has presented in 4.1.2 section.

The following figure presnets an example of the processing results of the threee feature detection and mathcing algorithms for the deformed scenarios.



(a) DIC processing result



(b) SIFT processing result

(c) SURF processing result

Figure 24. Processing results of DIC, SIFT and SURF algorithms for deformation group

The experimental processing results of the three different deformed scenes are shown in the table below. The data in the table is error percentages calculated through dividing the errors by the ground truths.

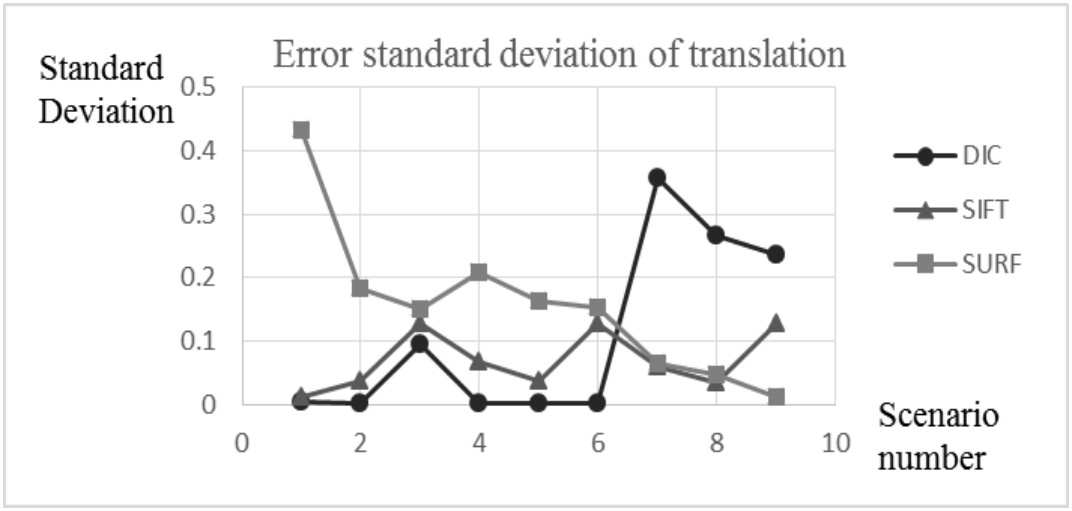
Table 3. Error percentages of the three algorithms for different deformed scenarios

Deformation (%)	Scene 1	Scene 2	Scene 3	Average
DIC	17.5012	18.8390	18.5891	18.3098
SIFT	19.9427	26.1856	27.3290	24.4858
SURF	19.5472	25.6559	26.3965	23.8665

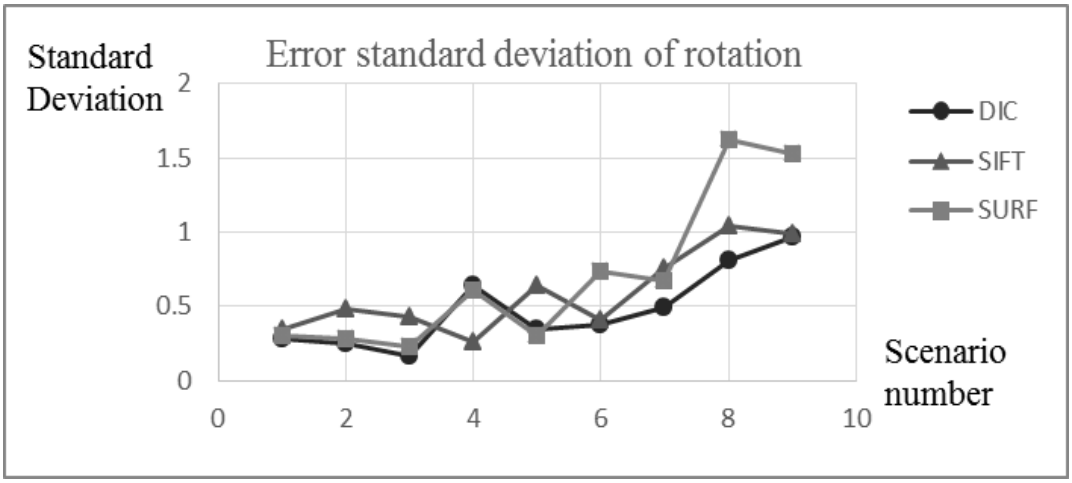
(d)

5.2 Measuring error distributions

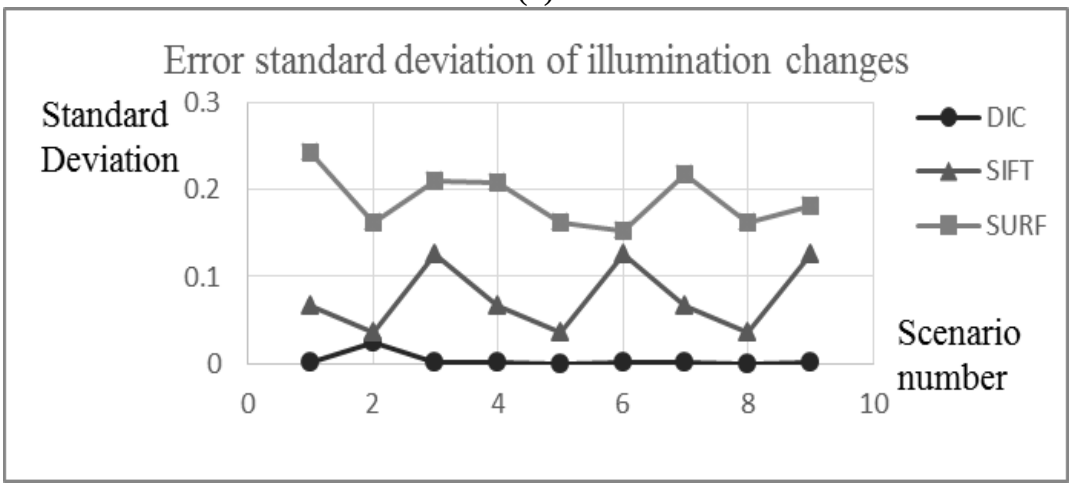
The measuring error distributions (standard deviation (ð) of the errors) are observed and calculated for each experimental subset. This procedure aims at reflecting the deviation that lies in the errors. It can help to evaluate the stability of the algorithms in terms of their processing errors. The results are shown in Figure 25.



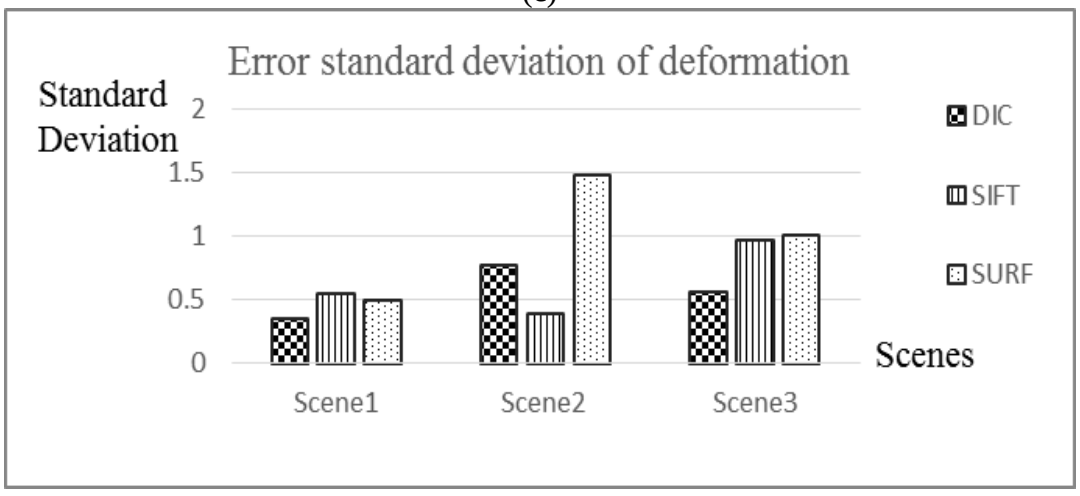
(a)



(b)



(c)



(d)

Figure 25. Feature point error distributions standard deviation: (a) Translation; (b) Rotation; (c) Illumination changes; (d) Deformation

5.3 Efficiency comparison

The time efficiency of the algorithms is also taken into account to evaluate the overall performance of the feature detection and matching algorithms. The running time of the algorithms was presented in Table 4. The unit of the data is in second.

Table 4: Running time of the three algorithms for different image scenarios: (a) Translation; (b) Rotation; (c) Illumination changes

Translation (sec)	3 pixels			5 pixels			8 pixels		
	Scene 1	Scene 2	Scene 3	Scene 1	Scene 2	Scene 3	Scene 1	Scene 2	Scene 3
DIC	5.979	4.894	4.148	5.733	5.285	4.584	12.356	12.253	12.435
SIFT	1.046	1.05	1.023	1.045	1.342	1.277	1.025	1.062	1.543
SURF	0.982	0.981	0.961	0.983	0.958	0.957	0.982	0.975	0.960

(a)

Rotation (sec)	0.5 degree			1 degree			1.5 degree		
	Scene 1	Scene 2	Scene 3	Scene 1	Scene 2	Scene 3	Scene 1	Scene 2	Scene 3
DIC	11.446	11.666	9.982	11.687	11.788	11.927	12.396	12.408	12.396
SIFT	0.99	0.98	0.991	1.035	1.083	1.053	1.042	1.098	1.022
SURF	0.941	0.959	0.944	0.982	0.955	0.997	0.995	0.997	0.994

(b)

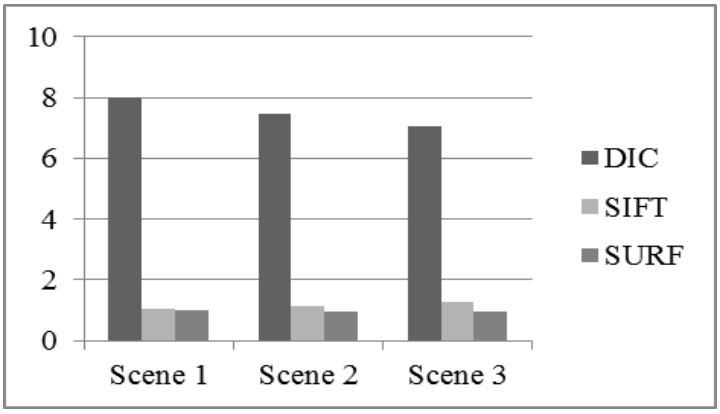
Illumination (sec)	Less illumination			Normal illumination			More illumination		
	Scene 1	Scene 2	Scene 3	Scene 1	Scene 2	Scene 3	Scene 1	Scene 2	Scene 3
DIC	5.733	5.285	4.584	4.974	5.261	5.986	5.898	4.962	5.365
SIFT	1.0446	1.3417	1.277	1.1256	1.235	1.338	1.261	1.325	1.167
SURF	0.983	0.952	0.958	0.978	0.951	0.952	0.979	0.956	0.951

(c)

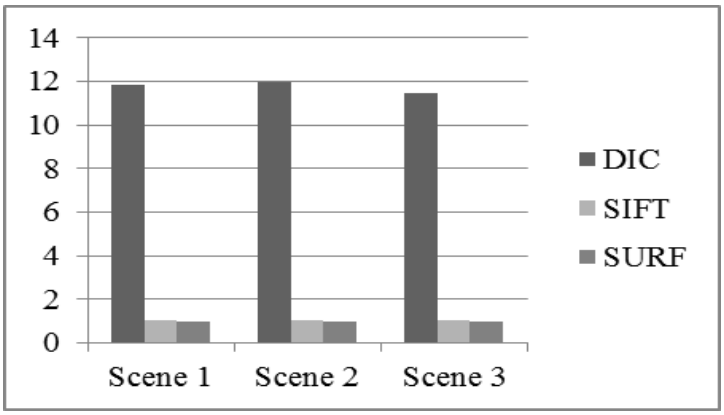
The figure below visualized the data in above table by histograms. From the figure, we can observe that the running time values of SURF algorithm are the smallest among the three algorithms. Both SIFT and SURF algorithms' running time for these three scenarios is around 1 second, which actually can be considered as real-time or near real-time processing.

However, for the DIC algorithm, the running time of rotation group is over 10

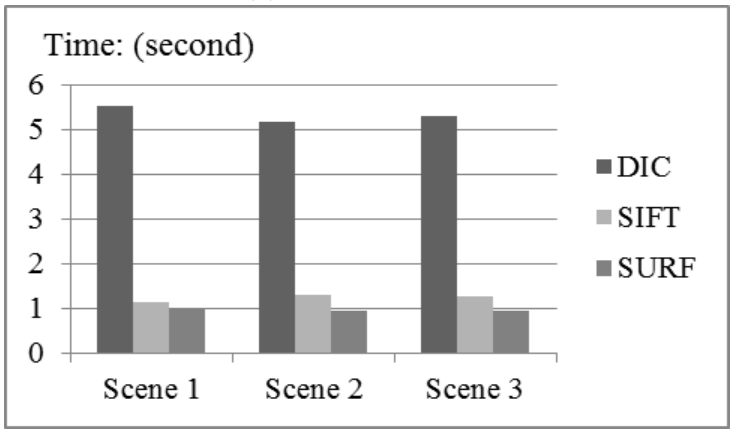
seconds. For other groups, DIC algorithm also needs over 5 seconds to process the experimental data. Therefore, the efficiency of SIFT and SURF algorithms are much higher than that of DIC algorithm from these comparison results.



(a) Translation scenarios



(b) Rotation scenarios



(c) Illumination changing scenarios

Figure 26. Efficiency comparisons of the three algorithms for (a) Translation scenarios, (b) Rotation scenarios, and (c) Illumination changing scenarios

5.4 Error estimation in real world scale

Based on the imaging principle of optical camera, the pixel-based errors can be converted into real world scale by specifying different image resolutions at the given camera capture distances.

The digital camera employed in our experiments is Canon 5D mark III which equips with a 36×24 mm full-frame CMOS sensor. The camera lens adopted in the experiments has a 30 mm fixed focal length. The capturing distance is set to be 5 meters. By specifying different image resolutions (5760×3840, 2880×1920, 1920×1280, 720×480), the errors in real world scale can be calculated based on the pixel-based errors reported in section 5.1. As shown in Table 5, the errors are in millimeter scale.

Table 5. Real world scale error estimation (camera capture distances: 5 m)

Resolution: 5760×3840				
Real-world error estimation (mm)	Translation	Rotation	Illumination	Deformation
DIC	0.0038	0.4293	0.0009	0.5992
SIFT	0.0258	0.6413	0.0279	0.8013
SURF	0.2575	0.6128	0.2738	0.7810
Resolution: 2880×1920				
Real-world error estimation (mm)	Translation	Rotation	Illumination	Deformation
DIC	0.0076	0.8586	0.0019	1.1983
SIFT	0.0516	1.2826	0.0558	1.6025
SURF	0.5150	1.2256	0.5476	1.5620
Resolution: 1920×1280				
Real-world error estimation (mm)	Translation	Rotation	Illumination	Deformation
DIC	0.0114	1.2879	0.0028	1.7975
SIFT	0.0773	1.9240	0.0837	2.4038
SURF	0.7726	1.8383	0.8214	2.3430
Resolution: 720×480				

Real-world error estimation (mm)	Translation	Rotation	Illumination	Deformation
DIC	0.0304	3.4344	0.0074	4.7934
SIFT	0.2062	5.1306	0.2233	6.4102
SURF	2.0601	4.9022	2.1903	6.2481

Based on the error data presented in the above table, the statistical histograms are generated to reveal the error estimation for the three algorithms intuitively. The bin-plot of the estimation errors is shown in Figure 27.

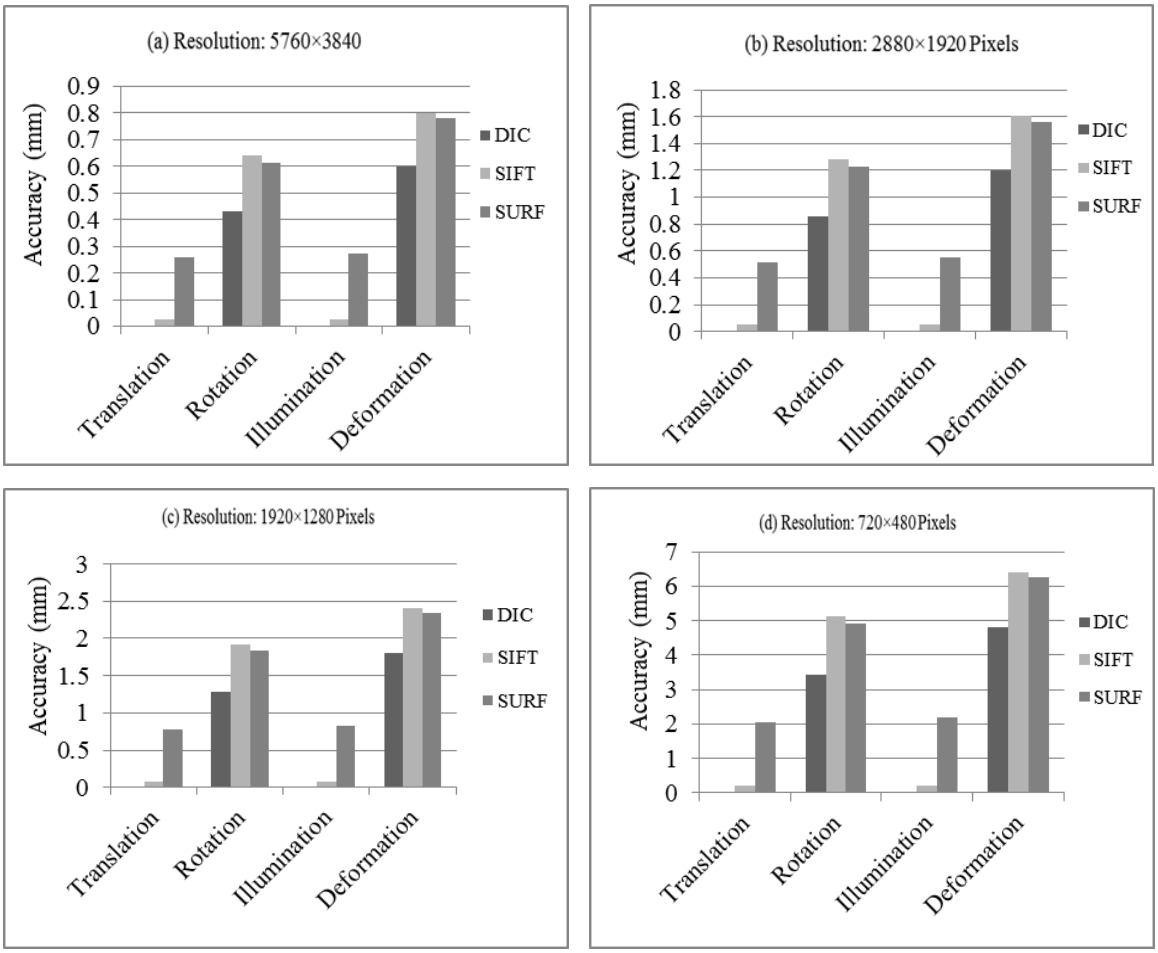
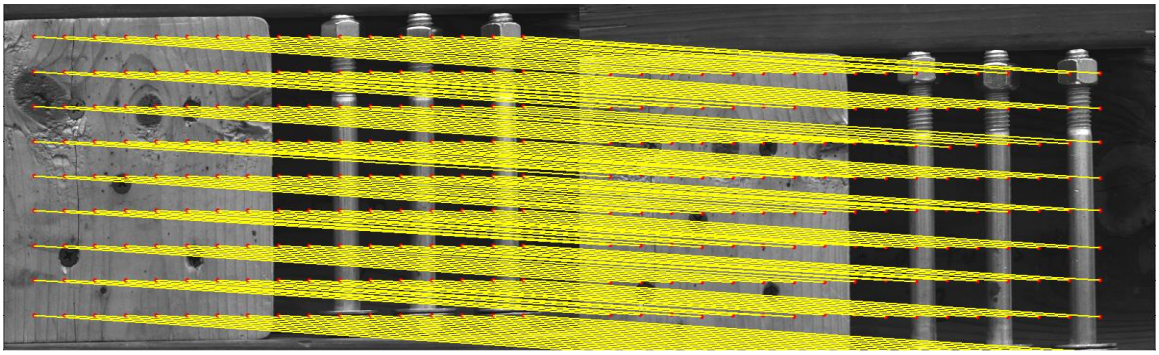


Figure 27. Real world scenario testing results of the three algorithms for synthetic images with different resolutions

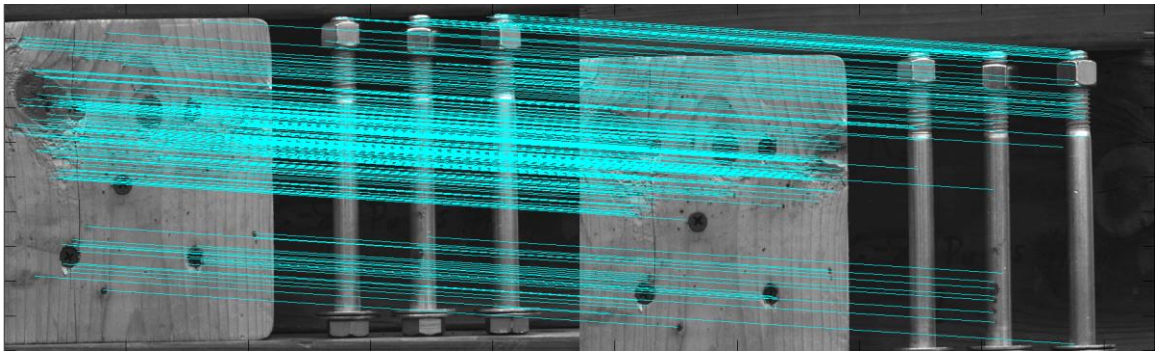
5.5 Real world scenario testing

Two groups of experiments were conducted in our concrete laboratory at WVU to further test the three feature detection and matching algorithms. The instrument employed in the experiments is Manta G-233-B industrial camera with a 2/3 inch COMS sensor, and a 50 mm fixed focal length lens was used. The image dataset has been presented in 4.1.3 section. The images in group 1 are taken with 15.4 pixel deflection (equals to 2 mm displacement in real world) between each two neighboring images, and the images in the second group 2 are taken with 22.9 pixel deflection (3 mm real world displacement interval). All the testing images used in the experiments have a 2048*1088 resolution.

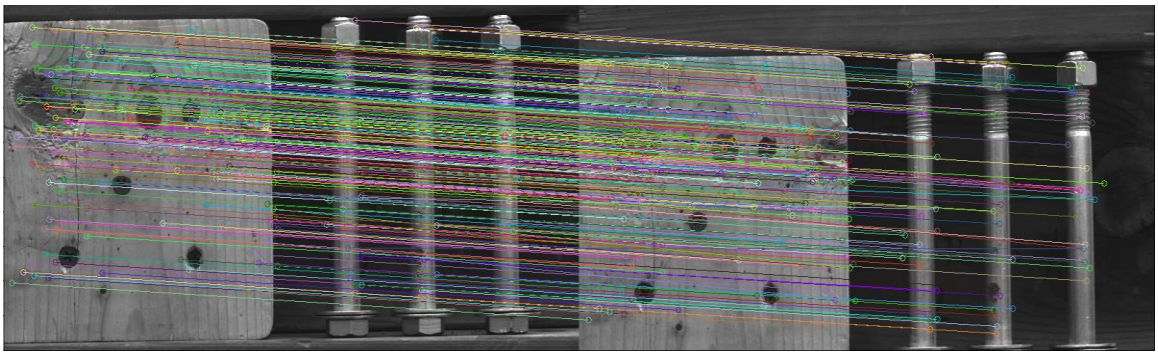
The figure below shows the examples of feature detection and matching results of the three algorithms for industrial image groups.



(a) DIC processing result



(b) SIFT processing result



(c) SURF processing result

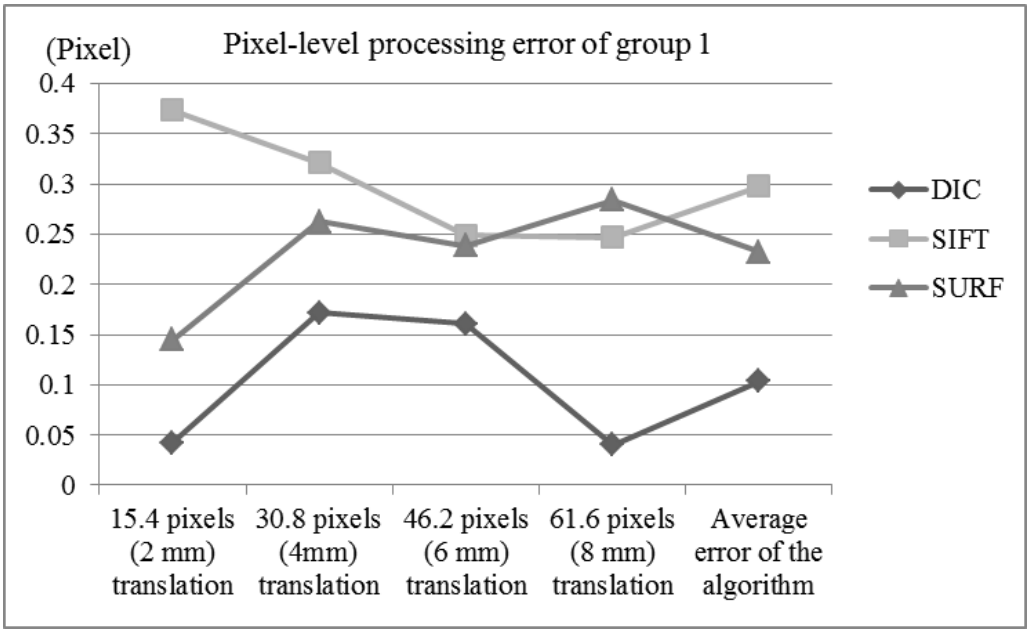
Figure 28. Processing results of DIC, SIFT and SURF algorithms for deformation group

The three feature detection and matching algorithms were respectively employed to process the two groups of industrial images. The processing results are presented in Table 6. The data in the table is pixel-based processing errors.

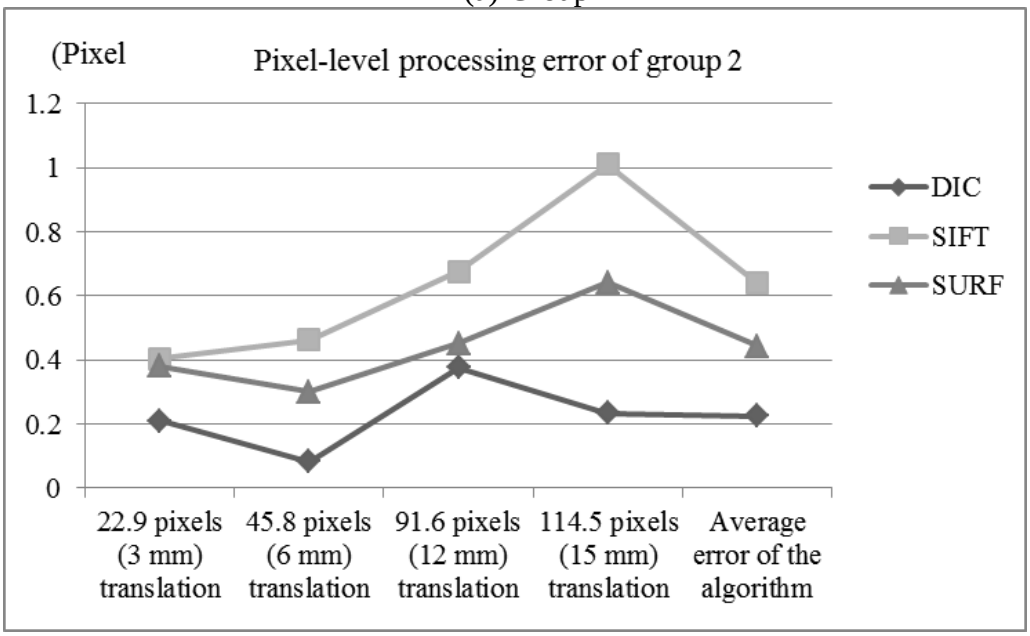
Table 6. Pixel-level errors of the algorithms for industrial images

Group 1					
Average Error (pixel)	15.4 pixels (2 mm) translation	30.8 pixels (4 mm) translation	46.2 pixels (6 mm) translation	61.6 pixels (8 mm) translation	Average Error of the algorithm
DIC	0.0421	0.1716	0.1605	0.0400	0.1036
SIFT	0.3734	0.3212	0.2488	0.2470	0.2976
SURF	0.1452	0.2624	0.2388	0.2840	0.2326
Group 2					
Average Error (pixel)	22.9 pixels (3 mm) translation	45.8 pixels (6 mm) translation	91.6 pixels (12 mm) translation	114.5 pixels (15 mm) translation	Average Error of the algorithm
DIC	0.2112	0.0831	0.3764	0.2334	0.2260
SIFT	0.4052	0.4626	0.6760	1.0112	0.6388
SURF	0.3817	0.3016	0.4509	0.6426	0.4442

Then, to compare the error performance more intuitively, the line char of avergae erros of the three algorithms is reproted in the figure below.



(a) Group 1



(b) Group 2

Figure 29. Pixel-level processing errors of the three algorithms for industrial images of real world scenario

Based on imaging principle, the capturing distance of the industrial images is calculated to be 0.95 meter. Additionally, by specifying different camera capture

distances, the pixel-based errors in Table 6 can be re-calculated into real world in millimeter scale. Table 7 presents the computing results.

Table 7. Real world-scale errors of the three algorithms for processing industrial images

Group 1	(Real camera capture distance is 0.95 m)				
Average error (mm)	Scenario 1	Scenario 2	Scenario 3	Scenario 4	Total Average
DIC	0.0055	0.0223	0.0208	0.0052	0.0135
SIFT	0.0485	0.0417	0.0323	0.0321	0.0386
SURF	0.0189	0.0341	0.03	0.0369	0.0302
(Estimated camera capture distance is 5 m)					
Average error (mm)	Scenario 1	Scenario 2	Scenario 3	Scenario 4	Total Average
DIC	0.0301	0.1226	0.1146	0.0286	0.074
SIFT	0.2667	0.2294	0.1777	0.1764	0.2126
SURF	0.1037	0.1874	0.1706	0.2028	0.1661
(Estimated camera capture distance is 10 m)					
Average error (mm)	Scenario 1	Scenario 2	Scenario 3	Scenario 4	Total Average
DIC	0.0605	0.2465	0.2305	0.0575	0.1488
SIFT	0.5364	0.4614	0.3573	0.3547	0.4275
SURF	0.2086	0.3769	0.343	0.4079	0.3341
(Estimated camera capture distance is 30 m)					
Average error (mm)	Scenario 1	Scenario 2	Scenario 3	Scenario 4	Total Average
DIC	0.1821	0.7418	0.6937	0.173	0.4476
SIFT	1.6139	1.38832	1.0753	1.0674	1.2862
SURF	0.6277	1.1342	1.032	1.2274	1.0053
(Estimated camera capture distance is 50 m)					
Average error (mm)	Scenario 1	Scenario 2	Scenario 3	Scenario 4	Total Average
DIC	0.3036	1.237	1.1569	0.2885	0.7465
SIFT	2.6915	2.3153	1.7932	1.7801	2.145
SURF	1.0468	1.8915	1.7211	2.0469	1.6766
Group 2	(Real camera capture distance is 0.95 m)				
Average error (mm)	Scenario 5	Scenario 6	Scenario 7	Scenario 8	Total Average
DIC	0.0277	0.0109	0.0493	0.0306	0.0296
SIFT	0.0531	0.0606	0.0886	0.1325	0.0837
SURF	0.05	0.0395	0.0591	0.0842	0.0582

(Estimated camera capture distance is 5 m)					
Average error (mm)	Scenario 5	Scenario 6	Scenario 7	Scenario 8	Total Average
DIC	0.1522	0.0599	0.2712	0.1682	0.1629
SIFT	0.292	0.3333	0.4871	0.7286	0.4602
SURF	0.275	0.2173	0.3249	0.463	0.3201
(Estimated camera capture distance is 10 m)					
Average error (mm)	Scenario 5	Scenario 6	Scenario 7	Scenario 8	Total Average
DIC	0.306	0.1205	0.5453	0.3382	0.3275
SIFT	0.5871	0.6703	0.9795	1.4651	0.9255
SURF	0.553	0.437	0.6534	0.931	0.6436
(Estimated camera capture distance is 30 m)					
Average error (mm)	Scenario 5	Scenario 6	Scenario 7	Scenario 8	Total Average
DIC	0.9209	0.3625	1.6409	1.0176	0.9855
SIFT	1.7666	2.01692	2.9474	4.4085	2.7849
SURF	1.6641	1.3148	1.966	2.8014	1.9366
(Estimated camera capture distance is 50 m)					
Average error (mm)	Scenario 5	Scenario 6	Scenario 7	Scenario 8	Total Average
DIC	1.5357	0.6045	2.7364	1.6971	1.6434
SIFT	2.9461	3.3636	4.9153	7.3519	4.6442
SURF	2.7752	2.1927	3.2786	4.6718	3.2296

Based on the computing results shown in the above table, the line chart of the real world scale average errors for the three algorithms has been generated and shown in the figure below.

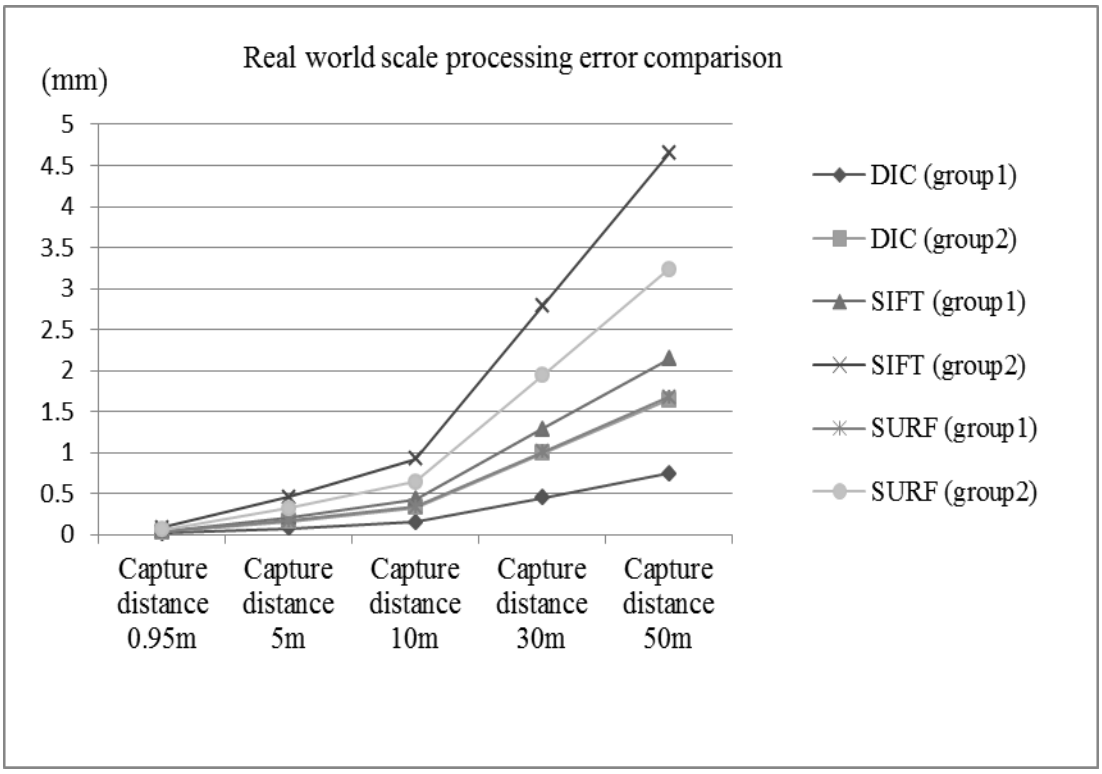


Figure 30. Real world scale processing errors of the three algorithms

In the figure above, the processing average errors of the three algorithms (DIC, SIFT and SURF) are presented for images of group 1 and group 2, separately. The camera capture distances are specified to be the 5m, 10m, 30m and 50m for estimating the processing errors when the camera station is from different distances to the target.

5.6 Measuring deflections and strains for real world scenarios

The image dataset of real world scenarios used for measuring deflections and strains has been presented in 4.2.2 section.

The methodology proposed in 2.2.3 section has been implemented in MATLAB

platform to generate a user interface for the deflection and strain computing process. The figure below shows the user interface.

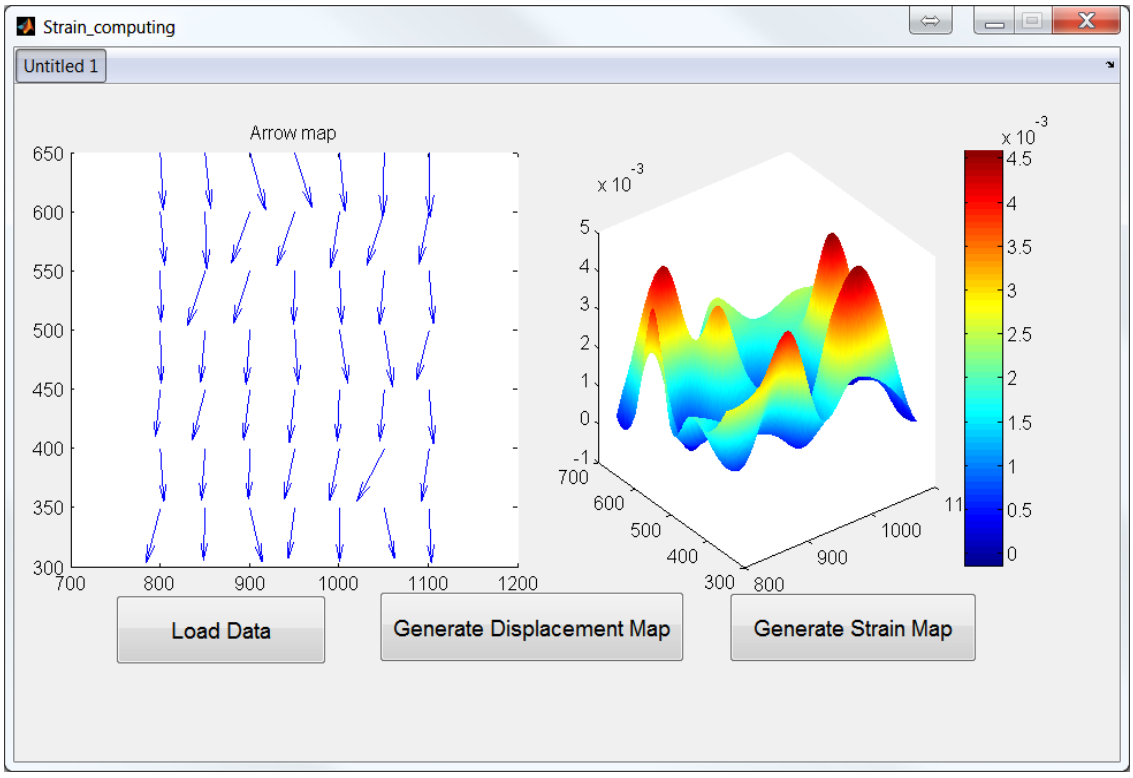


Figure 31. User interface for computing in-plane deflections and strains

➢Deflection measuring results

The industrial image dataset reported in section 4.1.3 was used for measuring in-plane deflections. There are 2 groups of image data, the ground truth of group 1 is 2 mm (15.4 pixel image-based deflection) real world vertical deflection between each neighboring images, and for group 2, the ground truth is 3 mm real world deflection (22.9 pixel image-based deflection) along vertical direction.

The average errors for deflection measuring are presented in the table below. Both the pixel-based errors and real world scale errors are reported.

Table 8. Average errors of measuring deflections for real world scenarios

Average error for deflection						
Group 1						
Ground Truth		15.4 pixels (2 mm)	30.8 pixels (4 mm)	46.2 pixels (6 mm)	61.6 pixels (8 mm)	Average
DIC	Average err (pixel)	0.0421	0.1716	0.1605	0.04	0.1036
	Average err (mm)	0.0055	0.0223	0.0208	0.0052	0.0135
Group 2						
Ground Truth		22.9 pixels (3 mm)	45.8 pixels (6 mm)	91.6 pixels (12 mm)	114.5 pixels (15 mm)	Average
DIC	Average err (pixel)	0.2112	0.0831	0.3764	0.2334	0.226
	Average err (mm)	0.0301	0.1226	0.1146	0.0286	0.074

Table 9. Accuracy of measuring deflections for real world scenarios

Deflection measuring accuracy						
Error percentages	Ground truth 2 mm	Ground truth 4 mm	Ground truth 6 mm	Ground truth 8 mm	Average Error percentages (%)	Accuracy
Group 1	0.2772	0.5516	0.3444	0.0591	0.3081	0.9969
Group 2	0.8986	0.1673	0.3868	0.2084	0.4153	0.9958

➢ Strain measuring results

After the deflection measuring testing, we conducted another several groups of experiments to test the performance of the proposed method to measure in-plane strains for real world scenarios. The industrial image dataset has been presented in 4.2.2 section. In this experiment, we use LVDT (Linear Variable Differential Transformer) to record the ground truth data which will be used to evaluate the processing results of the experiment

The figure below shows the processing results. The displacement maps as shown in Figure 32 reflect the sparse displacement field of the moving surface members of the target. There are 49 (7 by 7) points are specified to compute the displacement. The directions of the arrows represent the directions of the displacement of these points, and the length of the arrows represents the quantity value of the displacement.

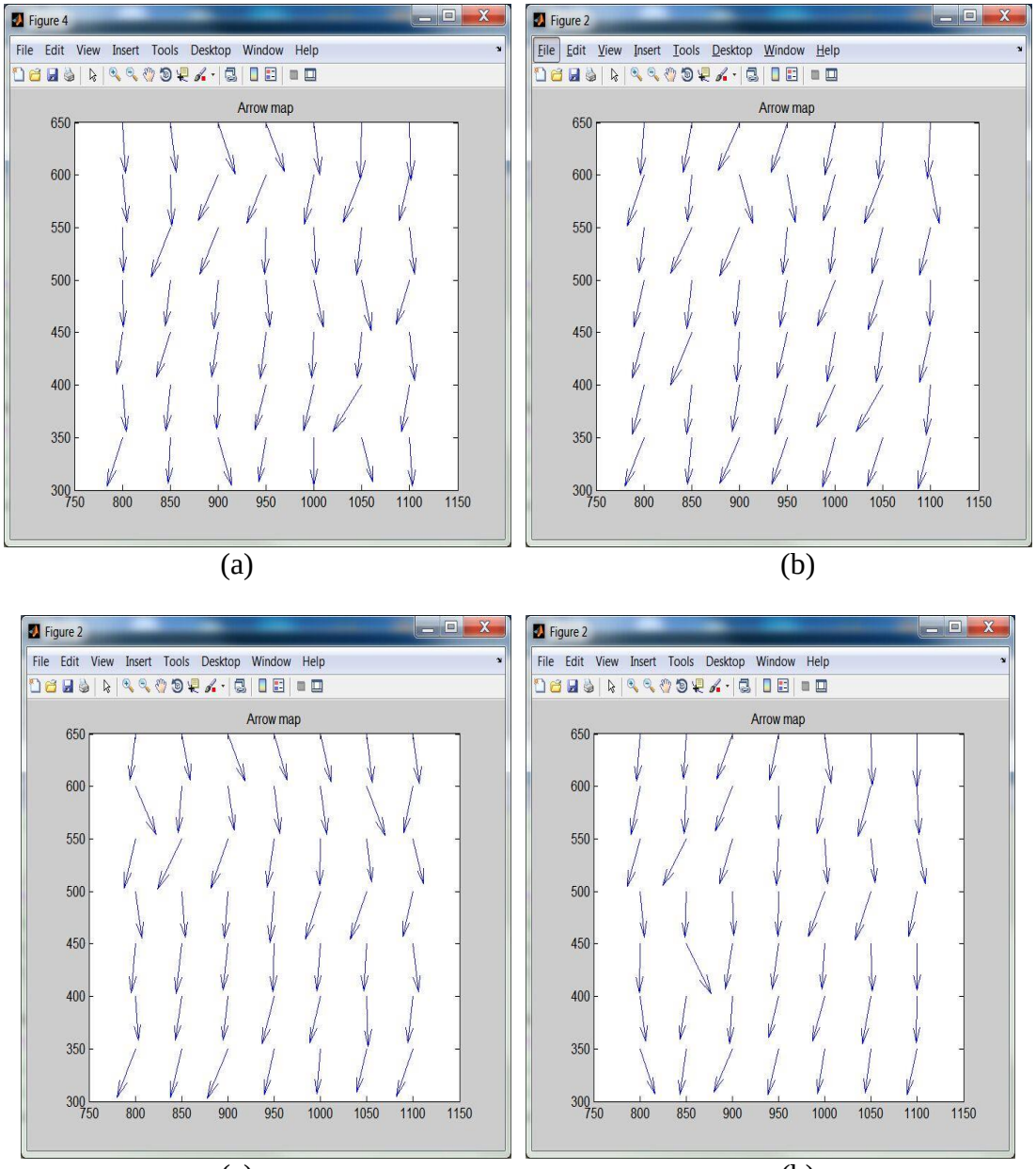
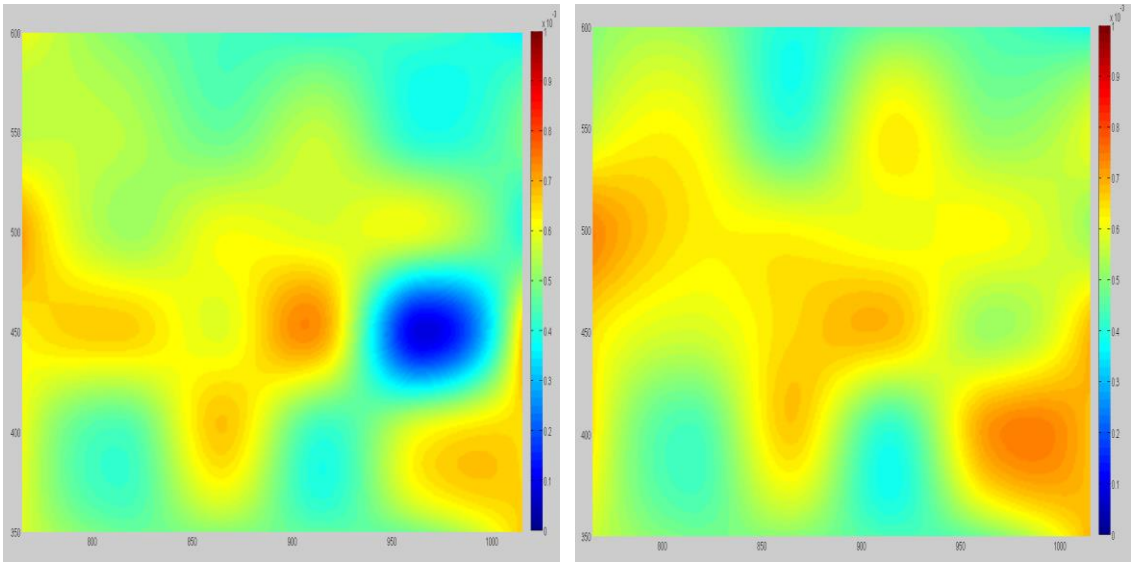


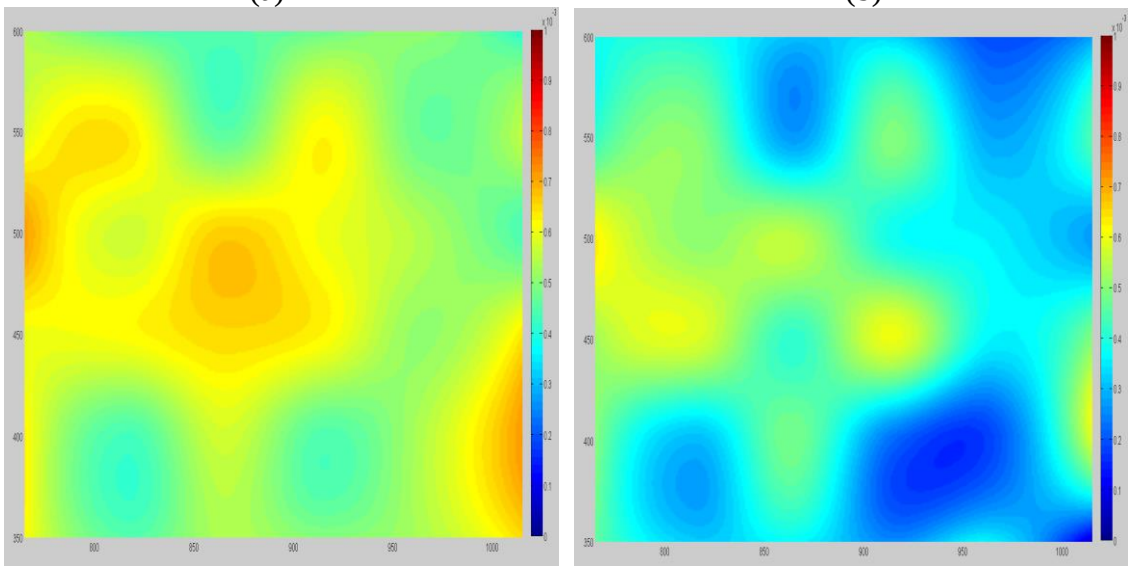
Figure 32. Displacement maps for measuring the real world scenarios

After computing the displacement field for the sparse points, we need to conduct the interpolation procedure as described in section 4.1.3, and bilinear interpolation is adopted in our method. Then, strain maps can be generated based on the displacement

filed after interpolation. The figure below show the final strain maps after interpolation for the real world testing scenarios.



(a) (b)



(a) (b)

Figure 33. Strain maps for measuring the real world scenarios

➢Measuring accuracy

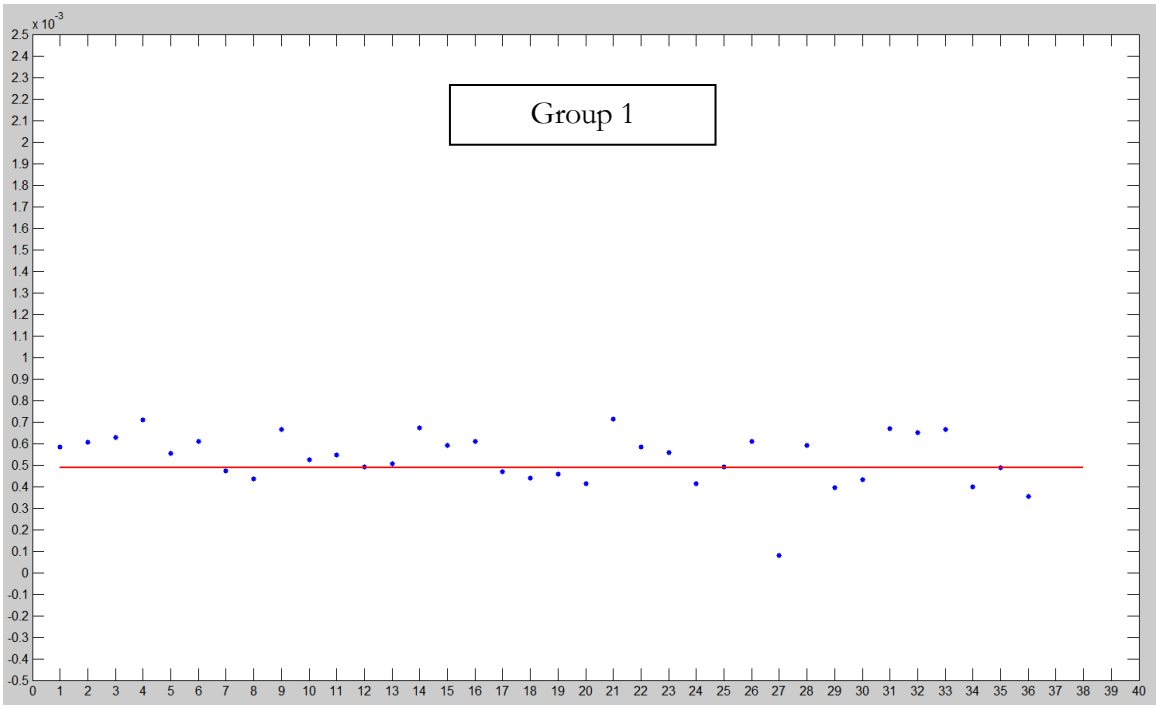
The processing results were compared with the ground truth data that are

recorded by the LVDT devices. As shown in the industrial image dataset in section 4.2.2, the LVDT devices were install on the left and right side of the testing sample. Therefore, we compared our processing results with the ground truths from LVDT devices. 4 groups of experiments were conducted in this section. The following table shows the experimental results.

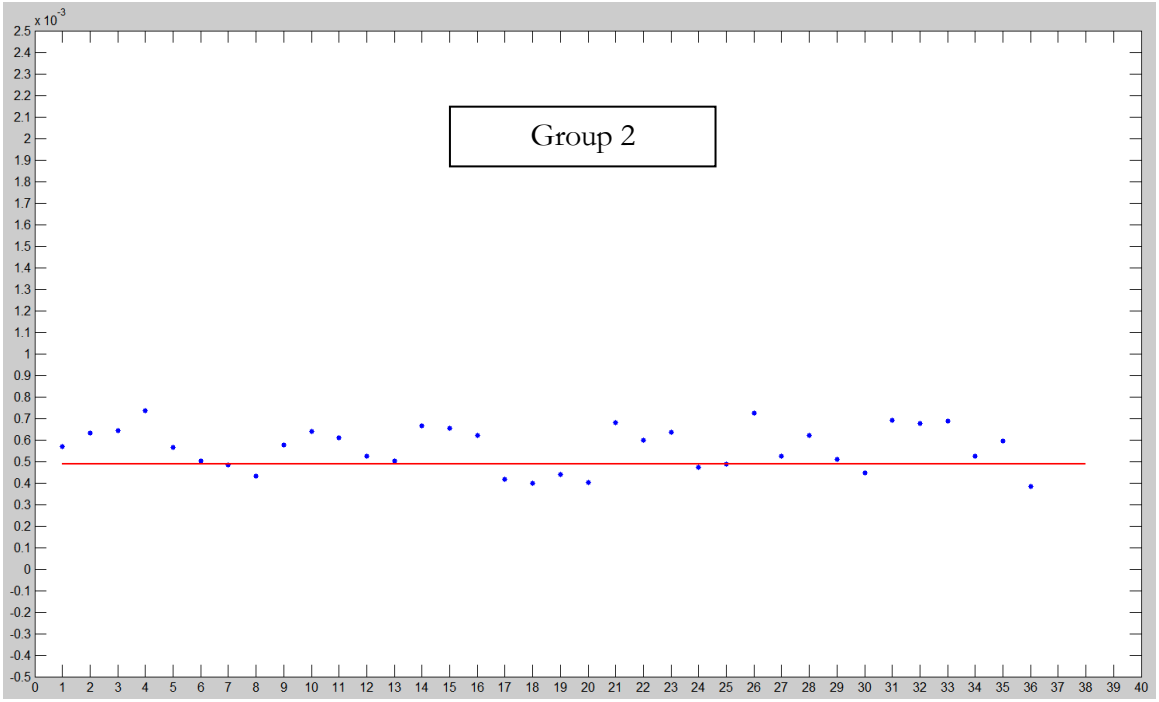
Table 10. LVDT experimental results

LVDT experiment	Group 1	Group 2	Group 3	Group 4	Average
LVDT values (inch)	0.00401	0.004	0.004	0.004	0.004
Strain ground truth	0.000501	0.0005	0.0005	0.0005	0.0005
Loading stress (psi)	2166.08	2210.37	2159.17	2055.88	2147.88
Young's modulus (Kpsi)	4321.36	4420.74	4318.34	4111.76	4293.05
Average strain (measured)	0.000529	0.000562	0.000555	0.000406	0.000513
Error percentages	0.0562	0.1249	0.1104	0.1879	0.1198
Accuracy	0.9438	0.8751	0.8896	0.8121	0.8802

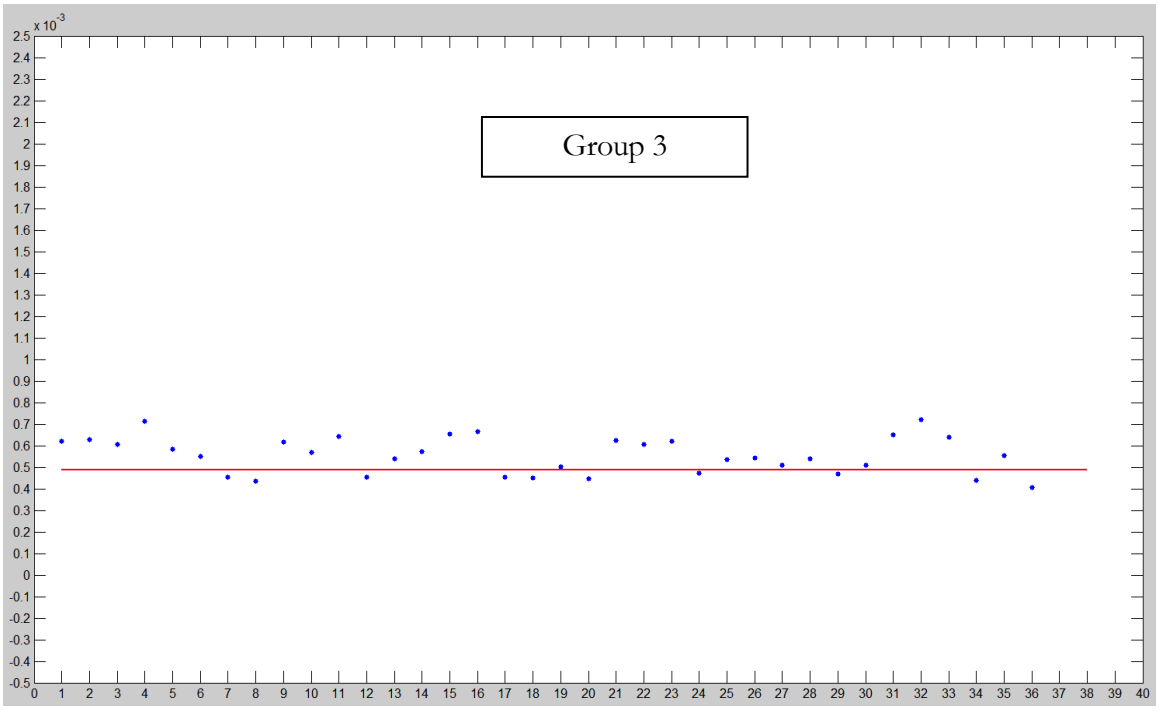
The figure below shows the accuracy distribution of measuring the in-plane strains for the real world scenarios. The ground truth of the strains recorded by the LVDT device is 5×10^{-4} for each single group of industrial images. The blue points plotted in the figure below are the measured point-strain values along the vertical direction of the images. The red line is the average strain ground truth. This figure is used for representing the accuracy distribution of the calculated strains.



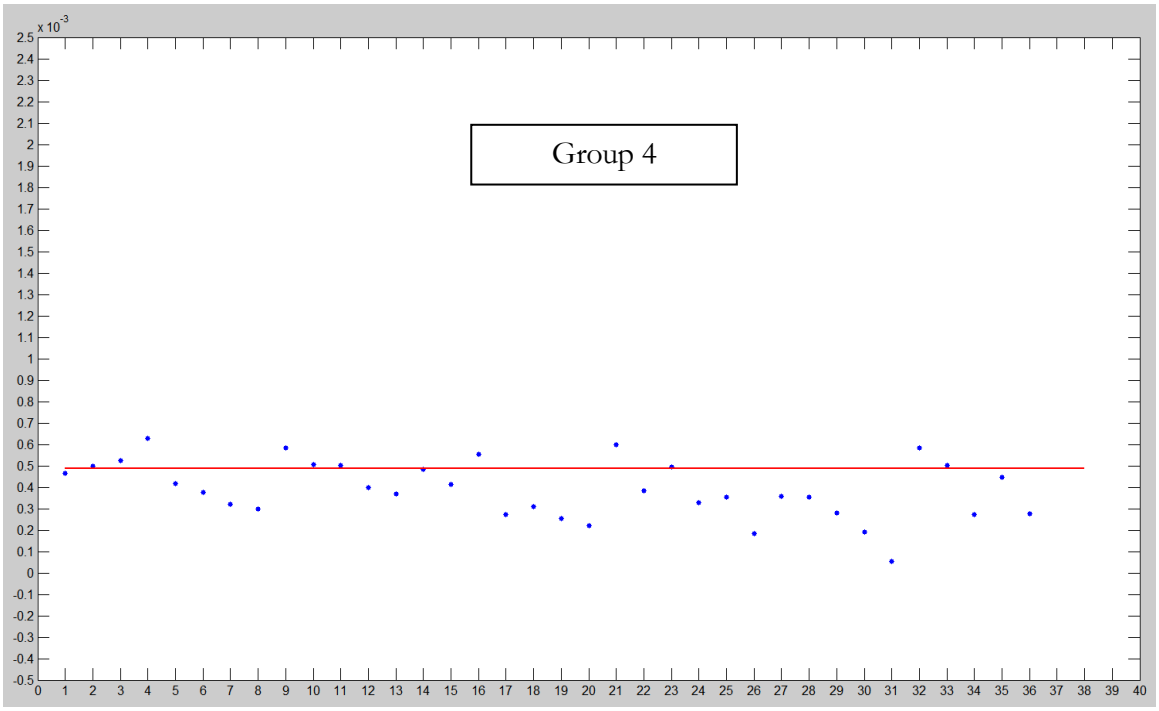
(a)



(b)



(c)



(d)

Figure 34. Point strain distribution: (a) group 1, (b) group 2, (c) group 3, (d) group 4

To quantitatively compare the in-plane strain measuring accuracy, firstly, the error percentages of the experimental results were computed by applying the following formula:

$$\text{Error percentage} = (\text{Measuring value} - \text{Ground truth})/\text{Ground truth} \quad \text{(Eq. 12)}$$

Then, the accuracy of the algorithm can be calculated based on the formula give as below:

$$\text{Measuring accuracy} = 1 - \text{Error percentage} \quad \text{(Eq. 13)}$$

The table below reports the strain measuring accuracy and the error standard deviation of the 4 experimental groups.

Table 11. Strain measuring accuracy for the 4 groups of experiments

Strain measuring accuracy						
Strains	Group 1	Group 2	Group 3	Group 4	Average	Ground truth
	0.000529	0.000562	0.000555	0.000406	0.000513	0.000501
Accuracy	Group 1	Group 2	Group 3	Group 4	Average accuracy	
	0.9425	0.8765	0.8909	0.8111	0.8802	
Error STD	Group 1	Group 2	Group 3	Group 4	Average standard deviation	
	0.000125	0.000101	0.000085	0.000133	0.000111	

As shown in the table above, the average accuracy of measuring strains for these groups of experiments is around 88%. The average standard deviation value of the measuring errors is around 1.11×10^{-4} while the ground truth of average strain is recorded as 5.01×10^{-4}.

5.7 Integrated user interface for measuring in-plane deflections and strains

We have implemented an integrated user interface for the deflection and strain measuring procedures with Matlab GUI programing, from camera initial parameter configuration to data collection, data processing, and to the generation of displacement maps and strain maps. The figure below shows the integrated user interface.

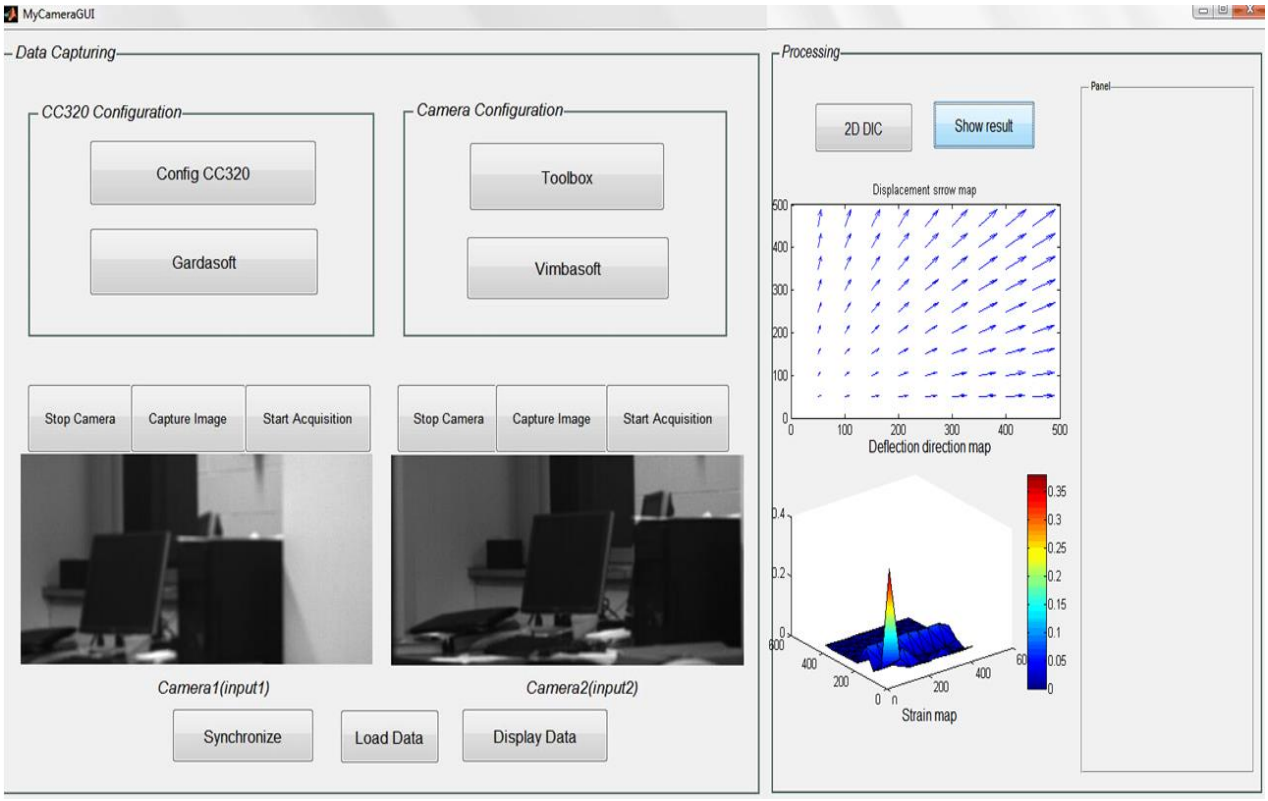


Figure 35. Integrated user interface for measuring in-plane deflections and strains

In the user interface shown above, there are three main parts. The first part is about camera initialization parameters configuration. We also integrate the external time trigger CC320 into our measuring system for highly precise dual camera-synchronization. The second part includes data collection (two adaptive windows for image capturing) and the captured image data visualization. The third part is about data processing (deflections and strains computation) and result visualization (generation of displacement maps and strain maps).

CHAPTER 6: DISCUSSION AND ANALYSIS

6.1 Algorithm accuracy performance

From Table 2 and Table 3, we can see that DIC achieved the best accuracy performance among all of the testing groups including translation, rotation, illumination changes and deformation. SIFT led to more accurate measurements than SURF in translation and illumination changes scenarios. In terms of the rotation and deformation scenarios, the performance of SIFT and SURF is not as good as that of DIC algorithm.

6.2 Algorithm efficiency performance

The running time of the algorithms processing different datasets is shown in Table 4. All the images processed in the efficiency experiments have the resolution of 501×501 pixels. From Table 4, the three algorithms' running time for different scenarios (translation, rotation, and illumination change) has little difference. This result indicates that scenario differences have relatively small influence on the time efficiency of the algorithms.

However, the running time of the three algorithms when processing the same scenario is quite different. From the results in Table 4, SURF achieved the best time efficiency among these three algorithms. SIFT is the second. The time complexity of DIC is not as good as SURF and SIFT. The average running time of SURF algorithm is around 0.9 second and the running time of SIFT algorithm is around 1 second. The running time of SURF and SIFT can be considered to be real time or near real time. However, the running time of DIC varies with different scenarios; it is around 5 seconds in illumination

change group, and around 12 seconds in rotation and 8 pixel translation groups.

6.3 Error estimation in real world scale

In Table 5, the image pixel errors were computed into real-world metric measurements. As for translation subset, when the camera capture distance is 5 m from the target with a 5760×3840 image resolution, the error in real world scale is approximately 0.004 mm for DIC algorithm. With the image resolution reducing, the real world-scale error increases to 0.03 mm with a 720×480 resolution. Similar results can be observed in rotation, illumination changes and deformation sunsets. As a result, regarding the same algorithm, the real world accuracy is positively correlated with image resolution.

In addition, comparing the real-world accuracy between different algorithms, we can see that DIC algorithm achieved the best accuracy performance among the three. SURF obtained the worst accuracy performance. Especially, when the image resolution is reduced to 720×480 pixels, the real world-scale error of SURF algorithm for dealing with translation scenarios is over 2 mm that is much higher than that of DIC and SIFT.

In particular, all of the three algorithms seem to have difficulties in processing the rotation and deformation subsets when the images have relatively small resolution. For example, in deformation subset, when the image resolution is 720×480, the accuracy performances of all three algorithms are over 4 mm. As a consequence, these algorithms still call for further improvement to overcome this limitation.

6.4 Real world scenario testing

From the results shown in Table 5, the DIC algorithm achieved the best accuracy performance among the three algorithms for processing the real world scenarios. It can obtain an average accuracy within 0.5 pixel that is much higher than that of SURF and SIFT. SURF algorithm obtained the second best accuracy results. SIFT revealed the worst accuracy performance among the three algorithms. The average error of SIFT is around 3 times of result of DIC algorithm.

In terms of the real world scale estimation of the average errors shown in Table 6 and Table 7, the DIC algorithm still reveals superiority among the three algorithms, SURF algorithm is the second, while the accuracy performance of SIFT algorithm actually is not as good as DIC and SURF. For instance, when the capturing distance is 50 meter, the average error of SIFT in real world scale is even over 7 mm that is considered to be meaningless for our research task.

6.5 Measuring deflections and strains

In section 5.6, to further validate the feasibility and measuring accuracy performance of our proposed image correlation-based algorithm, several groups of experiments were conducted. From Table 8 and Table 9, we can see the accuracy of measuring in-plane deflections is 99.69% for the first group experiment, and a 99.58% accuracy performance of the second group. The first group experiment used the dataset with a 2 mm (ground truth) interval between each neighboring industrial image, and the second group experiment has a 3 mm ground truth interval. These experimental results of the two groups show the algorithm achieved relatively higher accuracy when processing

those scenarios with smaller deflection ground truths.

From the data in Table 10, we can see the average accuracy for measuring strains of real world scenarios is 88.03%, and the average standard deviation is 1.11×10^{-4}, which is used for evaluating the distribution of measuring errors. The experimental results in section 5.6 reveals our proposed method can achieve high accuracy to measure in-plane deflections and strains for real world scenarios.

CHAPTER 7: CONCLUSION AND FUTURE WORK

This thesis proposes to apply visual sensing techniques to measure in-plane deflections and strains for civil infrastructure applications. It entails applying image correlation-based algorithm to automatically detect and matching image features which are used to computing the deflections and strains. A series of experiments were conducted to compare and evaluate those three feature detection and matching algorithms (DIC, SIFT, SURF) in order to obtain the optimal one using for our measuring goal. What is more, several experiments were also conducted for the purpose of further validating the applicability of the DIC-based method for real world scenarios. The experimental results reveal that the proposed DIC-based visual sensing method has achieved highly accurate measuring performance.

As the main contributions of this research, we studied and evaluated the three feature detection and matching algorithms (DIC, SIFT and SURF) for measuring in-plane deflections and strains of civil infrastructure applications. The gap of the accuracy and efficiency performance of applying DIC algorithm to measure in-plane deflections and

strains for civil infrastructures has been clearly identified through designing and implementing the experiments of both synthetic image group and real-world laboratory scenario group along with comparing to SIFT and SURF algorithms. Also, after finishing the evaluation work of the algorithms, the accuracy of measuring real-world scale deflections and strains by applying DIC algorithm is tested through conducting the LVDT experiment in lab environment. The experimental results reveal that the method can achieve around 88% measuring accuracy performance. So far, we have also developed an user interface for the DIC-based deflection and strain measuring method. It integrates all the involved procedures including data collection (configuring and controlling the Manta G-223B industrial cameras), data storage and transmission, data visualization, data processing (image feature detection and matching), in-plane deflection & strain computing and output visualization (generating displacement maps and strain maps). This user interface enables the users a better environment to facilitate their efficiency when applying the visual sensing-based method.

The visual sensing-based method for measuring in-plane deflections and strains is relatively novel in civil infrastructure health and safety monitoring field. It has several competitive advantages when comparing with other method (such as wire/wireless sensor networks). Firstly, this method can generate full-field measurement results (it can obtain continuous deflections and strains for the whole object). Besides, the cost-effective equipment and much more convenient set-up procedures will enable engineers to operate periodically and apply for different scales of civil infrastructure applications. As a result, this visual sensing-based method is a highly potential alternative to be applied to measure

deflections and strains for civil infrastructure applications.

However, the time efficiency of DIC is not as good as expected. It impedes the effectiveness of applying this method to acquire real-time deflection and strain measuring results. Therefore, as the future work, the time efficiency of DIC algorithm will be further investigated in detail and improved by reducing the computing complexity of the algorithm. Also, this method has not been applied to measure three dimensional real world scenarios, and the accuracy performance for 3D scenarios is still undetermined. In addition, some research work is also demanded to identify relations between the measured deflection/strain values and the safety/integrity situations of civil infrastructure applications so that our proposed deflection and strain measuring method can be finally employed to successfully address the widely existed health and safety issues of civil structures/infrastructures.

References

Acharya, T., and Tsai, P. S. (2007). Computational foundations of image interpolation algorithms. ACM Ubiquity, 8(42).

Alba, M., Fregonese, L., Prandi, F., Scaioni, M., & Valgoi, P. (2006). Structural monitoring of a large dam by terrestrial laser scanning. International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, 36(5), 6.

Barnes, J., Rizos, C., Kanli, M., Small, D., Voigt, G., Gambale, N., and Lamance, J. (2004). Structural deformation monitoring using locata. In 1st FIG International Symposium on Engineering Surveys for Construction Works and Structural Engineering.

Bay, H., Tuytelaars, T., and Van Gool, L. (2006). Surf: Speeded up robust features. Springer Berlin Heidelberg, In Computer Vision–ECCV 2006. 404-417.

Berardino, P., Fornaro, G., Lanari, R., and Sansosti, E. (2002). A new algorithm for surface deformation monitoring based on small baseline differential SAR interferograms. *Geoscience and Remote Sensing, IEEE Transactions on*,*40*(11), 2375-2383.

Bornert, M., Brémand, F., Doumalin, P., Dupré, J. C., Fazzini, M., Grédiac, M. and Wattrisse, B. (2009). Assessment of digital image correlation measurement errors: methodology and results. Experimental mechanics, 49(3), 353-370.

Chang, P. C., Flatau, A., and Liu, S. C. (2003). Review paper: health monitoring of civil

infrastructure. Structural health monitoring, 2(3), 257-267.

Chintalapudi, K., Fu, T., Paek, J., Kothari, N., Rangwala, S., Caffrey, J., and Masri, S. (2006). Monitoring civil structures with a wireless sensor network. Internet Computing, IEEE, 10(2), 26-34.

Dai, F., Feng, Y., and Hough, R. (2014). Photogrammetric error sources and impacts on modeling and surveying in construction engineering applications. Visualization in Engineering, 2(1), 2.

Dargie, W. and Poellabauer, C. (2010). Fundamentals of wireless sensor networks: theory and practice. John Wiley & Sons.

Fathi, H., & Brilakis, I. (2011). Automated sparse 3D point cloud generation of infrastructure using its distinctive visual features. Advanced Engineering Informatics, 25(4), 760-770.

Feng, Y., and Dai, F. (2014) Evaluation of Stereo Matching Algorithms for Temporary Structure Monitoring. In Computing in Civil and Building Engineering (pp. 1206-1213). ASCE.

Furukawa, Y., Curless, B., Seitz, S. M., and Szeliski, R. (2009). “Reconstructing building interiors from images.” IEEE 12th Int. Conf. on Computer Vision, IEEE, 80–87.

Golparvar-Fard, M., Balali, V., and de la Garza, J. M. (2012). Segmentation and recognition of highway assets using image-based 3D point clouds and semantic Texton

forests. Journal of Computing in Civil Engineering, 29(1).

Govender, N. (2009). Evaluation of feature detection algorithms for structure from motion.

Jahanshahi, M. R., and Masri, S. F. (2012). Adaptive vision-based crack detection using 3D scene reconstruction for condition assessment of structures.Automation in Construction, 22, 567-576.

Jonckheere, I., Fleck, S., Nackaerts, K., Muys, B., Coppin, P., Weiss, M., and Baret, F. (2004). Review of methods for in situ leaf area index determination: Part I. Theories, sensors and hemispherical photography. Agricultural and forest meteorology, 121(1), 19-35.

Juan, L, and Oubong G. (2009). A comparison of sift, pca-sift and surf. International Journal of Image Processing (IJIP) 3.4: 143-152

Karbhari, V. M., and Zhao, L. (2000). Use of composites for 21st century civil infrastructure. Computer methods in applied mechanics and engineering,185(2), 433-454.

Khan, N. Y., Brendan M., and Geoff W. (2011). "Sift and surf performance evaluation against various image deformations on benchmark dataset." Digital Image Computing Techniques and Applications (DICTA), 2011 International Conference on. IEEE.

Kim, S., Pakzad, S., Culler, D., Demmel, J., Fenves, G., Glaser, S., and Turon, M. (2007, April). Health monitoring of civil infrastructures using wireless sensor networks.

In Information Processing in Sensor Networks, 2007. IPSN 2007. 6th International Symposium on (pp. 254-263). IEEE.

Küntz, M., Jolin, M., Bastien, J., Perez, F., and Hild, F. (2006). Digital image correlation analysis of crack behavior in a reinforced concrete beam during a load test. Canadian Journal of Civil Engineering, 33(11), 1418-1425.

Li, H. N., Li, D. S., and Song, G. B. (2004). Recent applications of fiber optic sensors to health monitoring in civil engineering. *Engineering structures*, *26*(11), 1647-1657.

Lowe, D. G. (1999). Object recognition from local scale-invariant features. InComputer vision, 1999. The proceedings of the seventh IEEE international conference on (Vol. 2, pp. 1150-1157). Ieee.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. International journal of computer vision, 60(2), 91-110.

Lynch, J. P., and Kenneth J. L. (2006). A summary review of wireless sensors and sensor networks for structural health monitoring. Shock and Vibration Digest 38.2: 91-130

M. A. Fischler, R. C. Bolles, (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography, Commun. ACM 24 (6) 381–395.

Marquardt, D. W. (1963). An algorithm for least-squares estimation of nonlinear parameters. Journal of the Society for Industrial & Applied Mathematics, 11(2), 431-441.

Maas, H. G., and Uwe H. (2006). Photogrammetric techniques in civil engineering material testing and structure monitoring. Photogrammetric Engineering & Remote Sensing 72.1: 39-45

McCormick, N., and Lord, J. (2010). Digital image correlation. Materials today,13(12), 52-54.

Mikolajczyk, K., and Cordelia S. (2005). A performance evaluation of local descriptors. Pattern Analysis and Machine Intelligence, IEEE Transactions on 27.10: 1615-1630

Nghiem, A. T., Bremond, F., Thonnat, M., and Valentin, V. (2007, September). ETISEO, performance evaluation for video surveillance systems. In Advanced Video and Signal Based Surveillance, 2007. AVSS 2007. IEEE Conference on (pp. 476-481). IEEE.

Park, H. S., Lee, H. M., Adeli, H., and Lee, I. (2007). A new approach for health monitoring of structures: terrestrial laser scanning. Computer Aided Civil and Infrastructure Engineering, 22(1), 19-30.

Rainieri, C., Fabbrocino, G., and Cosenza, E. (2011). Integrated seismic early warning and structural health monitoring of critical civil infrastructures in seismically prone areasRoberts, G. W., Meng, X., & Dodson, A. H. (2004). Integrating a global positioning system and accelerometers to monitor the deflection of bridges. Journal of Surveying Engineering, 130(2), 65-72.. Structural Health Monitoring, 10(3), 291-308.

Roberts, G. W., Meng, X., and Dodson, A. H. (2004). Integrating a global positioning system and accelerometers to monitor the deflection of bridges. Journal of Surveying

Engineering, 130(2), 65-72.

Sabins, F. F. (2007). Remote sensing: principles and applications. Waveland Press.

Segundo, Mauricio Pamplona, et al. "Automating 3D reconstruction pipeline by surf-based alignment." Image Processing (ICIP), 2012 19th IEEE International Conference on. IEEE, 2012.

Taniguchi, Ayako, et al. (2013). "Automated assessment of small bowel motility function based on three-dimensional zero-mean normalized cross correlation."Biomedical Engineering and Informatics (BMEI), 2013 6th International Conference on. IEEE.

Ta, Duy-Nguyen, et al. (2009). "Surftrac: Efficient tracking and continuous object recognition using local feature descriptors." Computer Vision and Pattern Recognition, CVPR.

Udd, E. (1998). Early efforts to initiate the field of fiber optic smart structures at Mcdonnell Douglas. In 5th Annual International Symposium on Smart Structures and Materials (pp. 12-18). International Society for Optics and Photonics.

Wahbeh, A. M., John P. C., and Sami F. M. (2003). A vision-based approach for the direct measurement of displacements in vibrating systems. Smart Materials and Structures 12.5: 785

Wang, Y., and Alberto M. C. (2002). Full-field measurements of heterogeneous deformation patterns on polymeric foams using digital image correlation. International

Journal of Solids and Structures 39.13: 3777-3796

Wu, C., Agarwal, S., Curless, B., and Seitz, S. M. (2011). “Multicore bundle adjustment.” Conf. on Computer Vision and Pattern Recognition (CVPR), IEEE, 3057–3064.

Young, W. C., and Budynas, R. G. (2002). Roark's formulas for stress and strain (Vol. 7). New York: McGraw-Hill.

Zhao, J. Q., Zeng, P., Lei, L. P., and Ma, Y. (2012). Initial guess by improved population-based intelligent algorithms for large inter-frame deformation measurement using digital image correlation. Optics and Lasers in Engineering,50(3), 473-490.

APPENDIX 1 Original code of deflection and strain computation and user interface

```
function varargout = Strain_computing(varargin)
% Author: Youyi Feng, 03-25-2015
% Department of Civil and Environmental Engineering
% Email: yofeng@mix.wvu.edu
% STRAIN_COMPUTING MATLAB code for Strain_computing.fig
%      STRAIN_COMPUTING, by itself, creates a new STRAIN_COMPUTING or raises the existing
%     singleton*.
%
%      H = STRAIN_COMPUTING returns the handle to a new STRAIN_COMPUTING or the handle to
%     the existing singleton*.
%
%      STRAIN_COMPUTING('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in STRAIN_COMPUTING.M with the given input arguments.
%
%      STRAIN_COMPUTING('Property','Value',...) creates a new STRAIN_COMPUTING or raises the
%      existing singleton*.  Starting from the left, property value pairs are
```

```
%      applied to the GUI before Strain_computing_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Strain_computing_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES
% Edit the above text to modify the response to help Strain_computing
% Last Modified by GUIDE v2.5 04-Apr-2015 16:28:43


% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Strain_computing_OpeningFcn, ...
                   'gui_OutputFcn',  @Strain_computing_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
```

```
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end


if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Strain_computing is made visible.
function Strain_computing_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Strain_computing (see VARARGIN)
```

```
% Choose default command line output for Strain_computing

handles.output = hObject;

% Update handles structure

guidata(hObject, handles);

% UIWAIT makes Strain_computing wait for user response (see UIRESUME)

% uiwait(handles.figure1);



% --- Outputs from this function are returned to the command line.

function varargout = Strain_computing_OutputFcn(hObject, eventdata, handles)

% varargout  cell array for returning output args (see VARARGOUT);

% hObject    handle to figure

% eventdata  reserved - to be defined in a future version of MATLAB

% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure

varargout{1} = handles.output;



% --- Executes on button press in loaddata.

function loaddata_Callback(hObject, eventdata, handles)

% hObject    handle to loaddata (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
load 1.txt;


% --- Executes on button press in axes1.
function arrowmap_Callback(hObject, eventdata, handles)
% hObject    handle to axes1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
[col1,col2,col3,col4,col5,col6,col7, col8]=...
   textread('1.txt','%f %f %f %f %f %f %f %f',-1);
%
%% Compute the displacement matrix
POI=[col2,col3]; % Point of interest with displacement vector
DIS=sum((col2.*col2)+(col3.*col3),2);  % compute the displacement matrix
%% Plot the scatter points for displacement
% Define the order of the points
%% To find 81*81 interest points
%
```

```
%******************Set m,n values for m*n points************%

m=7; % determined for 81*81 points %?50?450????5????

n=7;

No_x = repmat(800:(300/(m-1)):1100,1,n); %generate the x coordinates

A=zeros(1,m*n);

for i=1:m

A(:,((i-1)*m+1):((i-1)*m+m))=50*(i-1)+350; %generate the y coordinates

end

No_y = A;

%

%% plot the scatter points and displacement

%{

figure;

plot(No_x,No_y, '.');

title ('Diaplacement scatter points')

%}

%Plot the arrow map

axes(handles.axes1);

No_x1=No_x+(col2)';
```

```
No_y1=No_y+(col3)';

quiver(No_x,No_y,(col2)',(col3)');

title ('Arrow map')



% --- Executes on button press in axes2.

function strainmap_Callback(hObject, eventdata, handles)

% hObject    handle to axes2 (see GCBO)

% eventdata  reserved - to be defined in a future version of MATLAB

% handles    structure with handles and user data (see GUIDATA)

%% For 81*81 case

%

tic

[col1,col2,col3,col4,col5,col6,col7, col8]=...

   textread('1.txt','%f %f %f %f %f %f %f %f',-1);

POI=[col2,col3]; % Point of interest with displacement vector

DIS=sum((col2.*col2)+(col3.*col3),2);



%******************Set m,n values for m*n points************%

m=7; % determined for 81*81 points %?50?450????5????
```

```
n=7;

No_x = repmat(800:(300/(m-1)):1100,1,n); %generate the x coordinates

A=zeros(1,m*n);

for i=1:m

A(:,((i-1)*m+1):((i-1)*m+m))=50*(i-1)+350; %generate the y coordinates

end

No_y = A;

%******************Set J,K values for J*K points************%

% n*n displacement matrix

J=7;

K=7;

Dis_x= (reshape(col2,J,K))';

Dis_y= (reshape(col3,J,K))';

% displacement matrix for point P

Dis_x_p= Dis_x(:,1:J-1);

Dis_y_p= Dis_y(1:K-1,:);

% displacement matrix for point Q

Dis_x_q= Dis_x(:,2:J);

Dis_y_q= Dis_y(2:K,:);
```

```
% compute displacement micro increasement Delta_u;

%(Displacement of point Q minus thant of point P

X_Delta_u= Dis_x_q - Dis_x_p;

Y_Delta_u= Dis_y_q - Dis_y_p;

% calculate the point strains for each point

D_pq= 50; % the distance interval of points P and Q

X_strain_0 = X_Delta_u/D_pq;

Y_strain_0 = Y_Delta_u/D_pq;

% Complement the strain matrix to n*n dimension to fit the n*n points

A=zeros(J,1);

B=zeros(1,K);

X_strain= [X_strain_0, A];

Y_strain= [Y_strain_0; B];

% Compute the point strains matrix

Strain= ((X_strain.^2)+(Y_strain.^2)).^0.5;
```

APPENDIX 2 Integrated user interface original code

```
function varargout = MyCameraGUI(varargin)
% Author: Youyi Feng
% Department of Civil Engineering and Environment, WVU
% Email: yofeng@mix.wvu.edu
% MYCAMERAGUI MATLAB code for MyCameraGUI.fig
%      MYCAMERAGUI, by itself, creates a new MYCAMERAGUI or raises the existing
%      singleton*.
%
%      H = MYCAMERAGUI returns the handle to a new MYCAMERAGUI or the handle to
%      the existing singleton*.
%
%      MYCAMERAGUI('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in MYCAMERAGUI.M with the given input arguments.
%
%      MYCAMERAGUI('Property','Value',...) creates a new MYCAMERAGUI or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before MyCameraGUI_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to MyCameraGUI_OpeningFcn via varargin.
```

```
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES
% Edit the above text to modify the response to help MyCameraGUI
% Last Modified by GUIDE v2.5 27-Feb-2015 10:30:41
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @MyCameraGUI_OpeningFcn, ...
                   'gui_OutputFcn',  @MyCameraGUI_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
```

```
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before MyCameraGUI is made visible.
function MyCameraGUI_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to MyCameraGUI (see VARARGIN)


% Choose default command line output for MyCameraGUI


handles.output = hObject;
handles.vid1=videoinput('gige',1);
triggerconfig(handles.vid1,'hardware','DeviceSpecific','DeviceSpecific');
%triggerconfig(handles.vid1,'manual');
handles.vid1.FramesPerTrigger = Inf;
%triggerconfig(handles.vid1,'manual');
```

```
handles.vid2=videoinput('gige',2);

triggerconfig(handles.vid2,'hardware','DeviceSpecific','DeviceSpecific');

%triggerconfig(handles.vid2,'manual');

handles.vid2.FramesPerTrigger = Inf;

%camera2trigger=triggerconfig(handles.vid2)



guidata(hObject, handles);

% UIWAIT makes MyCameraGUI wait for user response (see UIRESUME)

% uiwait(handles.MyCameraGUI);

uiwait(handles.MyCameraGUI);



% --- Outputs from this function are returned to the command line.

function varargout = MyCameraGUI_OutputFcn(hObject, eventdata, handles)

% varargout  cell array for returning output args (see VARARGOUT);

% hObject    handle to figure

% eventdata  reserved - to be defined in a future version of MATLAB

% handles    structure with handles and user data (see GUIDATA)

%handles.output = hObject;

%varargout{1} = handles.output;

% Get default command line output from handles structure
```

```
%varargout{1} = handles.output;



% --- Executes when user attempts to close MyCameraGUI.
function MyCameraGUI_CloseRequestFcn(hObject, eventdata, handles)
% hObject    handle to MyCameraGUI (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
delete(hObject);
delete(imaqfind);
close all;
clear all;
% Hint: delete(hObject) closes the figure
%delete(hObject);
% --- Executes on button press in startStopCamera.
function startStopCamera_Callback(hObject, eventdata, handles)
% hObject    handle to startStopCamera (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
axes(handles.cameraAxes);
vidRes1=get(handles.vid1,'VideoResolution');
```

```
nBands1=get(handles.vid1,'NumberOfBands');
set(handles.vid1,'ReturnedColorSpace','rgb');
himage1=imshow(zeros(vidRes1(2),vidRes1(1),nBands1));

%preview(handles.vid1,himage1);

if strcmp(get(handles.startStopCamera,'String'),'Start Camera')
    % Camera is off. Change button string and start camera.
    set(handles.startStopCamera,'String','Stop Camera')
    %start(handles.vid1)
    preview(handles.vid1,himage1);
    set(handles.startAcquisition,'Enable','on');
    set(handles.captureImage,'Enable','on');
else
    % Camera is on. Stop camera and change button string.
    set(handles.startStopCamera,'String','Start Camera')
    stop(handles.vid1)
    set(handles.startAcquisition,'Enable','off');
    set(handles.captureImage,'Enable','off');
end
```

```
%}

% --- Executes on button press in captureImage.

function captureImage_Callback(hObject, eventdata, handles)

% hObject    handle to captureImage (see GCBO)

% eventdata  reserved - to be defined in a future version of MATLAB

% handles    structure with handles and user data (see GUIDATA)

frame = get(get(handles.cameraAxes,'children'),'cdata'); % The current displayed frame

save('testImage1.mat', 'frame');



figure;

imshow(frame);

disp('Frame saved to file "testImage.mat"');



% --- Executes on button press in startAcquisition.

function startAcquisition_Callback(hObject, eventdata, handles)

% hObject    handle to startAcquisition (see GCBO)

% eventdata  reserved - to be defined in a future version of MATLAB

% handles    structure with handles and user data (see GUIDATA)

%axes(handles.cameraAxes);
```

```
if strcmp(get(handles.startAcquisition,'String'),'Start Acquisition')
    % Camera is not acquiring. Change button string and start acquisition.
    set(handles.startAcquisition,'String','Stop Acquisition');

    %stop(handles.vid1);
    %stoppreview(handles.vid1);
    %triggerconfig(handles.vid1,'hardware','DeviceSpecific','DeviceSpecific');
    start(handles.vid1);
    %trigger(handles.vid1);
    Initial_frameslogged = handles.vid1.FramesAcquired
    %wait (handles.vid1, 30);
    %disp('Waiting for CC320 trigger commands.....')

else
    % Camera is acquiring. Stop acquisition, save video data,
    % and change button string.
    stop(handles.vid1);
    disp('Saving captured video...');
    videodata = getdata(handles.vid1);
    save('testvideo1.mat', 'videodata');
```

```
    %save('testvideo.avi', 'videodata');
    disp('Video saved to file "testvideo1.mat"');
    frameslogged = handles.vid1.FramesAcquired
    % start(handles.vid3); % Restart the camera
    set(handles.startAcquisition,'String','Start Acquisition');
end

% --- Executes on button press in startStopCamera2.
function startStopCamera2_Callback(hObject, eventdata, handles)
% hObject    handle to startStopCamera2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%handles.output2 = hObject;

axes(handles.cameraAxes2);
vidRes2=get(handles.vid2,'VideoResolution');
nBands2=get(handles.vid2,'NumberOfBands');
set(handles.vid2,'ReturnedColorSpace','rgb');
himage2=imshow(zeros(vidRes2(2),vidRes2(1),nBands2));
```

```
% preview(handles.vid2,himage2);
if strcmp(get(handles.startStopCamera2,'String'),'Start Camera')
    % Camera is off. Change button string and start camera.
    set(handles.startStopCamera2,'String','Stop Camera')
    %start(handles.vid2)
    preview(handles.vid2,himage2);
    set(handles.startAcquisition2,'Enable','on');
    set(handles.captureImage2,'Enable','on');
else
    % Camera is on. Stop camera and change button string.
    set(handles.startStopCamera2,'String','Start Camera')
    stop(handles.vid2)
    set(handles.startAcquisition2,'Enable','off');
    set(handles.captureImage2,'Enable','off');
end


% --- Executes on button press in captureImage2.
function captureImage2_Callback(hObject, eventdata, handles)
% hObject    handle to captureImage2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)
frame2 = get(get(handles.cameraAxes2,'children'),'cdata'); % The current displayed frame
save('testImage2.mat', 'frame2');
figure;
imshow(frame2);
disp('Frame saved to file "testImage.mat"');

% --- Executes on button press in startAcquisition2.
function startAcquisition2_Callback(hObject, eventdata, handles)
% hObject    handle to startAcquisition2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

if strcmp(get(handles.startAcquisition2,'String'),'Start Acquisition')
    % Camera is not acquiring. Change button string and start acquisition.
     set(handles.startAcquisition2,'String','Stop Acquisition');
     %triggerconfig(handles.vid1,'hardware','DeviceSpecific','DeviceSpecific');
    start(handles.vid2);
    Initial_frameslogged = handles.vid2.FramesAcquired
    %wait (handles.vid2,20);
```

```
else
    % Camera is acquiring. Stop acquisition, save video data,
    % and change button string.
    stop(handles.vid2);
    disp('Saving captured video...');
    videodata = getdata(handles.vid2);
    save('testvideo2.mat', 'videodata');
    %save('testvideo.avi', 'videodata');
    disp('Video saved to file "testvideo2.mat"');
    frameslogged = handles.vid2.FramesAcquired
    % start(handles.vid2); % Restart the camera
    set(handles.startAcquisition2,'String','Start Acquisition');
end


% --- Executes on button press in cameraset1.
function cameraset1_Callback(hObject, eventdata, handles)
% hObject    handle to cameraset1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
imaqtool;


% --- Executes on button press in cameraset2.
function cameraset2_Callback(hObject, eventdata, handles)
% hObject    handle to cameraset2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
system('C:\Users\user\Desktop\EXE\Win32\VimbaViewer.exe');


% --- Executes on button press in configtrigger1.
function configtrigger1_Callback(hObject, eventdata, handles)
% hObject    handle to configtrigger1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
 f=figure('Name','CC320 Configuration','Num','off','Units','norm');
% Add the browser object on the right
jObject = com.mathworks.mlwidgets.html.HTMLBrowserPanel;
[browser,container] = javacomponent(jObject, [], f);
set(container, 'Units','norm', 'Pos',[0.3,0.05,0.65,0.9]);
```

```
% Add the URLs listbox on the left
urls = {'http://192.168.1.5/general.cgi'};
hListbox = uicontrol('style','listbox', 'string',urls, ...
       'units','norm', 'pos',[0.05,0.05,0.2,0.9], ...
       'userdata',browser);

% Set the listbox's callback to update the browser contents
cbStr=['strs = get(gcbo,"string"); ' ...
     'url = strs{get(gcbo,"value")};' ...
     'browser = get(gcbo,"userdata"); ' ...
     'msg=["<html><h2>Loading " url " - please wait"];'...  % no need for </h2></html>
     'browser.setHtmlText(msg); pause(0.1); drawnow;'...
     'browser.setCurrentLocation(url);'];
set(hListbox,'Callback',cbStr);
% --- Executes on button press in configtrigger2.
function configtrigger2_Callback(hObject, eventdata, handles)
% hObject    handle to configtrigger2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
 system('C:\Users\user\Desktop\EXE\CC320\GardasoftMaint.exe'); % invoke the Gardasoft;
```

```
% --- Executes on button press in synchronize.
function synchronize_Callback(hObject, eventdata, handles)
% hObject    handle to synchronize (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if strcmp(get(handles.synchronize,'String'),'Synchronization')
    % Camera is not acquiring. Change button string and start acquisition.
    set(handles.synchronize,'String','Stop');
    tic
    start(handles.vid1);
    toc
    start(handles.vid2);

else
    tic
    stop(handles.vid1);
    toc
    stop(handles.vid2);
    disp('Saving captured video...');
```

```
    videodata1 = getdata(handles.vid1);
    save('testvideo1.mat', 'videodata1');
    %save('testvideo.avi', 'videodata');
    frameslogged = handles.vid1.FramesAcquired
    disp('Video saved to file "testvideo1.mat"');

    disp('Saving captured video...');
    videodata2 = getdata(handles.vid2);
    save('testvideo2.mat', 'videodata2');
    %save('testvideo.avi', 'videodata');
    frameslogged = handles.vid2.FramesAcquired
    disp('Video saved to file "testvideo2.mat"');
    close all;

end

% --- Executes on button press in initialdata.
function initialdata_Callback(hObject, eventdata, handles)
% hObject    handle to initialdata (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)

%save video 1

load ('testvideo1.mat'); % read video file/ use "whos" command to see the variables

[h,w,g,N_frame]=size(videodata1); % N_frame is the number of frames (images) in this video

% testing show the imgae

 filename = 'film1';

writerObj = VideoWriter( [filename '.avi'] );

N=10; %set the rates that the video has

writerObj.FrameRate = N;

open(writerObj);

%figure;

for ii = 1: N_frame

   frame = videodata1(:,:,:,ii);

   %imshow(frame);

   f.cdata = frame;

   f.colormap = [];

   writeVideo(writerObj,frame);

end

close(writerObj);

%save video 2
```

```
load ('testvideo2.mat'); % read video file/ use "whos" command to see the variables

[h,w,g,N_frame]=size(videodata2); % N_frame is the number of frames (images) in this video

% testing show the imgae

 filename = 'film2';

writerObj2 = VideoWriter( [filename '.avi'] );

N=10; %set the rates that the video has

writerObj2.FrameRate = N;

open(writerObj2);

%figure;

for ii = 1: N_frame

   frame = videodata2(:,:,:,ii);

   %imshow(frame);

   f.cdata = frame;

   f.colormap = [];

   writeVideo(writerObj2,frame);

end

close(writerObj2);



% save image data for DIC, SIFT, SURF to process

%save left image
```

```
load('testvideo1.mat');

Img1=videodata1(:,:,:,1);

Img1=imresize(Img1,[500,500]);

imwrite(Img1,'C:\Users\user\Desktop\employedcode\2DDIC\Image_0.bmp'); % Save data into DIC folder path

imwrite(Img1,'C:\Users\user\Desktop\employedcode\SIFT\Image_0.bmp'); % Save data into SIFT folder path

imwrite(Img1,'C:\Users\user\Desktop\employedcode\SURF\TestImages\Image_0.bmp'); % Save data into SURF folder path

%save right image

load('testvideo2.mat');

Img2=videodata2(:,:,:,1);

Img2=imresize(Img2,[500,500]);

imwrite(Img2,'C:\Users\user\Desktop\employedcode\2DDIC\Image_1.bmp');% Save data into DIC folder path

imwrite(Img2,'C:\Users\user\Desktop\employedcode\SIFT\Image_1.bmp'); % Save data into SIFT folder path

imwrite(Img2,'C:\Users\user\Desktop\employedcode\SURF\TestImages\Image_1.bmp'); % Save data into SURF folder path




% --- Executes on button press in readinput.

function readinput_Callback(hObject, eventdata, handles)

% hObject    handle to readinput (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%{
load('testvideo1.mat');
load('testvideo2.mat');
axes(handles.figure1);
imshow(videodata1(:,:,:,1)); % show single left image
axes(handles.figure2);
imshow(videodata2(:,:,:,1));    % show single right image
%}
%
%Read and show the video data in processing box
mov1 = VideoReader( 'film1.avi' );
mov2 = VideoReader( 'film2.avi' );
for i=1:mov1.NumberOfFrames
   img1 = read( mov1, i );
   img2 = read( mov2, i );
   axes(handles.figure1);
   imshow(img1);
   axes(handles.figure2);
```

```
    imshow(img2);
end
%
%{
axes(handles.figure2);
mov2 = VideoReader( 'film2.avi' );
for i=1:mov2.NumberOfFrames
    img2 = read( mov2, i );
    imshow(img2);
end
%}


% --- Executes on button press in DIC.
function DIC_Callback(hObject, eventdata, handles)
% hObject    handle to DIC (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
open('C:\Users\user\Desktop\employedcode\2DDIC\Pool2008Pred.exe');
disp('2D DIC is running, please wait.....'");
%system('C:\Users\user\Desktop\EXE\Win32\VimbaViewer.exe');
```

```
%[x,y]=textread('C:\Users\user\Desktop\employedcode\2DDIC\Image_1.bmp_xps.txt','%f%f%*[^\n]','delimiter',';','headerlines',1);



% --- Executes on button press in showresult.

function showresult_Callback(hObject, eventdata, handles)

% hObject    handle to showresult (see GCBO)

% eventdata  reserved - to be defined in a future version of MATLAB

% handles    structure with handles and user data (see GUIDATA)

test=importdata('C:\Users\user\Desktop\employedcode\2DDIC\Image_1.bmp_xps.txt');

X_dis=test.data(:,2);

Y_dis=test.data(:,3);

[a,b]=size(test.data(:,2));

formatSpec = 'Obtained %d interest points from the inputs. \n';

fprintf(formatSpec,a);



col2=X_dis;

col3=Y_dis;

%visulize the processing result

POI=[col2,col3]; % Point of interest with displacement vector

DIS=sum((col2.*col2)+(col3.*col3),2);  % compute the displacement matrix
```

```
%% To find 81*81 interest points
%
%% Plot the scatter points for displacement
% Define the order of the points
% To find 9*9 interest points
%
%******************For 9*9 points************%
No_x = 50*[1,2,3,4,5,6,7,8,9,1,2,3,4,5,6,7,8,9,...
    1,2,3,4,5,6,7,8,9,1,2,3,4,5,6,7,8,9....
    1,2,3,4,5,6,7,8,9,1,2,3,4,5,6,7,8,9,...
    1,2,3,4,5,6,7,8,9,1,2,3,4,5,6,7,8,9,...
    1,2,3,4,5,6,7,8,9,];
No_y = 50*[1,1,1,1,1,1,1,1,1,2,2,2,2,2,2,2,2,2,...
    3,3,3,3,3,3,3,3,3,4,4,4,4,4,4,4,4,4,...
    5,5,5,5,5,5,5,5,5,6,6,6,6,6,6,6,6,6,...
    7,7,7,7,7,7,7,7,7,8,8,8,8,8,8,8,8,8,...
    9,9,9,9,9,9,9,9,9];
%
%******************Set m,n values for m*n points************%
```

```
%{

m=81; % determined for 81*81 points %?50?450????5????

n=81;

No_x = repmat(50:(400/(m-1)):450,1,n); %generate the x coordinates

A=zeros(1,m*n);

for i=1:m

A(:,((i-1)*m+1):((i-1)*m+m))=5*(i-1)+50; %generate the y coordinates

end

No_y = A;

%}

%% plot the scatter points and displacement

%figure;

%plot(No_x,No_y, '.');

%title ('Diaplacement scatter points')

%Plot the arrow map

No_x1=No_x+(col2)';

No_y1=No_y+(col3)';

axes(handles.arrowmap);

quiver(No_x,No_y,No_x1,No_y1);

rotate3d on
```

```
title ('Displacement srrow map')

%Triangulation of the scatter points and draw displacement map

% use delaunay triangulation

tri=delaunay(No_x,No_y);

z=DIS;

%{

figure;

trimesh(tri,No_x,No_y,z,...

   'FaceColor','interp',...

   'FaceLighting','phong',...

   'EdgeColor','k');

grid off;

colorbar;

title ('Displacement map')

%}

%% Compute Strain from displacement



% n*n displacement matrix

Dis_x= (reshape(col2,9,9))';

Dis_y= (reshape(col3,9,9))';
```

```
% displacement matrix for point P

Dis_x_p= Dis_x(:,1:8);

Dis_y_p= Dis_y(1:8,:);

% displacement matrix for point Q

Dis_x_q= Dis_x(:,2:9);

Dis_y_q= Dis_y(2:9,:);

% compute displacement micro increasement Delta_u;

%(Displacement of point Q minus thant of point P

X_Delta_u= Dis_x_q - Dis_x_p;

Y_Delta_u= Dis_y_q - Dis_y_p;

% calculate the point strains for each point

D_pq= 50; % the distance interval of points P and Q

X_strain_0 = X_Delta_u/D_pq;

Y_strain_0 = Y_Delta_u/D_pq;

% Complement the strain matrix to n*n dimension to fit the n*n points

A=zeros(9,1);

B=zeros(1,9);

X_strain= [X_strain_0, A];

Y_strain= [Y_strain_0; B];

% Compute the point strains matrix
```

```
Strain= ((X_strain.^2)+(Y_strain.^2)).^0.5;

% use delaunay triangulation and draw strain map

tri=delaunay(No_x,No_y);

z_1=reshape((Strain)',1,81);

axes(handles.strainmap);

trimesh(tri,No_x,No_y,z_1,...

   'FaceColor','interp',...

   'FaceLighting','phong',...

   'EdgeColor','k');

grid off;

colorbar;

rotate3d on

title ('Strain map');



%% For 81*81 case

%{

%******************Set J,K values for J*K points************%

% n*n displacement matrix

J=81;

K=81;
```

```
Dis_x= (reshape(col2,J,K))';

Dis_y= (reshape(col3,J,K))';

% displacement matrix for point P

Dis_x_p= Dis_x(:,1:J-1);

Dis_y_p= Dis_y(1:K-1,:);

% displacement matrix for point Q

Dis_x_q= Dis_x(:,2:J);

Dis_y_q= Dis_y(2:K,:);

% compute displacement micro increasement Delta_u;

%(Displacement of point Q minus thant of point P

X_Delta_u= Dis_x_q - Dis_x_p;

Y_Delta_u= Dis_y_q - Dis_y_p;

% calculate the point strains for each point

D_pq= 50; % the distance interval of points P and Q

X_strain_0 = X_Delta_u/D_pq;

Y_strain_0 = Y_Delta_u/D_pq;

% Complement the strain matrix to n*n dimension to fit the n*n points

A=zeros(J,1);

B=zeros(1,K);

X_strain= [X_strain_0, A];
```

```
Y_strain= [Y_strain_0; B];

% Compute the point strains matrix

Strain= ((X_strain.^2)+(Y_strain.^2)).^0.5;

% use delaunay triangulation and draw strain map

tri=delaunay(No_x,No_y);

z_1=reshape((Strain)',1,J*K);

figure;

trimesh(tri,No_x,No_y,z_1,...

   'FaceColor','interp',...

   'FaceLighting','phong',...

   'EdgeColor','k');

grid off;

colorbar;

title ('Strain map');

%
```