2007

AUTOMATED HIGH-SPEED MONITORING OF METAL TRANSFER FOR REAL-TIME CONTROL

Zhenzhou Wang
University of Kentucky, zwang5@uky.edu

ABSTRACT OF DISSERTATION

Zhenzhou Wang

The Graduate School
University of Kentucky
2007

AUTOMATED HIGH-SPEED MONITORING OF METAL TRANSFER FOR REAL-TIME CONTROL

ABSTRACT OF DISSERTATION

A dissertation submitted in partial fulfillment of the
requirements for the degree of Doctor of Philosophy in the
College of Engineering
at the University of Kentucky

By
Zhenzhou Wang
Lexington, Kentucky

Director: Dr Yu-Ming Zhang, Professor of Electrical Engineering
Lexington, Kentukcy
2007

ABSTRACT OF DISSERTATION

AUTOMATED HIGH-SPEED MONITORING OF METAL TRANSFER FOR REAL-TIME CONTROL

In the novel Double Electrode Gas Metal Arc Welding (DE-GMAW), the transfer of the liquid metal from the wire to the work-piece determines the weld quality and for applications where the precision is critical, the metal transfer process needs to be monitored and controlled to control the diameter, trajectory, and transfer rate of the droplet of liquid metal. In this doctoral research work, the traditional methods of tracking, Correlation, Least Square Matching (LSM) and Kalman Filtering (KF), are tried first. All of them failed due to the poor quality of the metal transfer image and the variety of the droplet. Then several novel image processing algorithms, Brightness Based Separation Algorithm (BBSA), Brightness and Subtraction Based Separation Algorithm (BSBSA) and Brightness Based Selection and Edge Detection Based Enhancement Separation Algorithm (BBSEDBESA), are proposed to compute the size and locate the position of the droplet. Experimental results verified that the proposed algorithms can automatically locate the droplets and compute the droplet size with an adequate accuracy. Since the final objective is to automatically process the metal transfer in real time, a real time processing system is implemented and the details are described. In traditional Gas Metal Arc Welding (GMAW), the famous laser back-lighting technique has been widely used to image the metal transfer process. Due to laser imaging system’s complexity, it is too inconvenient for practical applications. In this doctoral research work, a simplified laser imaging system is proposed and two effective image algorithms, Probability Based Double Thresholds Separation Algorithm and Edge Based Separation Algorithm, are proposed to process the corresponding captured metal transfer images. Experimental results verified that the proposed simplified laser back-light imaging system and image processing algorithms can be used for real time processing of metal transfer images.

KEYWORDS: GMAW, BSBSA, BBSEDBESA, PBDTSA, EBSA

__________Zhenzhou_Wang________________

__________June 12, 2007_________________

AUTOMATED HIGH-SPEED MONITORING OF METAL TRANSFER FOR REAL-TIME CONTROL

By
Zhenzhou Wang

____________Yuming Zhang_____________
Director of Dissertation

___________Yuming Zhang_____________
Director of Graduate Studies

_____________June 12, 2007____________

DISSERTATION

Zhenzhou Wang

The graduate School

University of Kentucky

2007

AUTOMATED MONITORING OF HIGH-SPEED METAL TRANSFER FOR REAL-TIME CONTROL

DISSERTATION

A dissertation submitted in partial fulfillment of the
requirements for the degree of Doctor of Philosophy in the
College of Engineering
at the University of Kentucky

By
Zhenzhou Wang
Lexington, Kentucky
Director: Dr Yu-Ming Zhang, Professor of Electrical Engineering
Lexington, Kentukcy
2007

ACKNOWLEDGEMENTS

This research is funded by the National Science Foundation under Grant #CMMI-0355324 and the Manufacturing Center of University of Kentucky. I would like to thank my advisor Dr. Yu-Ming Zhang and my dissertation committee members Drs. Sen-ching Samson Cheung, Larry Holloway and Zhong-wei Shen for their guidance, encouragement and support. In addition, I would also thank my colleagues at the Welding Research Laboratory for their help in my life and study.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Background

Image processing plays a critical role in extracting useful information from visual scenes and has been used in welding industry for seam tracking and in welding research for process monitoring[1]–[4]. Because skilled welders make quality welds primarily based on their visual feedback, it is believed that image processing and computer vision will also play a critical role in the development of the next generation intelligent welding machines especially for relatively complex processes such as the most widely used gas metal arc welding (GMAW) where visual information can greatly enhance the capability of process monitoring and control.

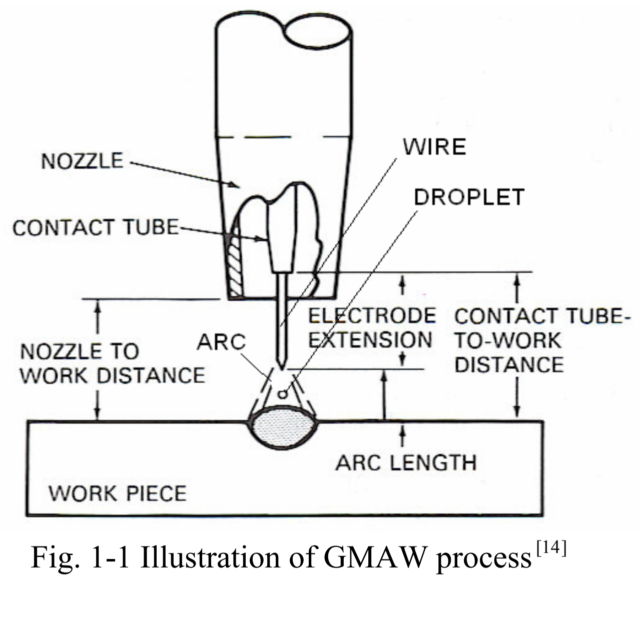


Fig. 1-1 Illustration of GMAW process[14]

Fig. 1-1 illustrates the GMAW process. In this process, the *wire* is fed to the *contact tube* which is connected to the positive terminal of the power supply and is thus positively charged. The *nozzle* restricts the shield gas to a local area surrounding the wire. When the wire touches the negatively charged *work piece* (connected to the negative terminal of the power supply), the tip of the wire is rapidly burnt and an *arc* is ignited in the gap between the wire and the work piece. The arc melts the wire and the melted metal forms a *droplet* at the tip of the wire; after the droplet is detached, a new droplet starts to form and the new cycle starts. This metal melting, droplet forming, and droplet detaching process is referred to as the metal transfer process [5]–[11] . The metal transfer process determines the welding quality in a critical way because the stability of the arc, the amount of the spatters, and the heat and mass inputs are all related to the metal transfer process.

It is apparent that the metal transfer process can be monitored to better understand and control the GMAW process. In fact, to understand and study this relatively rapid process, high speed cameras have been used to image and record the dynamic developments of the droplets [12][13] . Up to date, there are two main methods to acquire the metal transfer images. The first one is to use a camera with frame rate up to 2000-4000 frames per second and aimed at the metal transfer directly. Since the light intensity entering into the camera is proportional to the camera's exposure time, the effect of the strong arc light can be eliminated. Proper filters are used to eliminate the unwanted elements' effect further. Consequently, the clear metal transfer images can be obtained with high frequency.

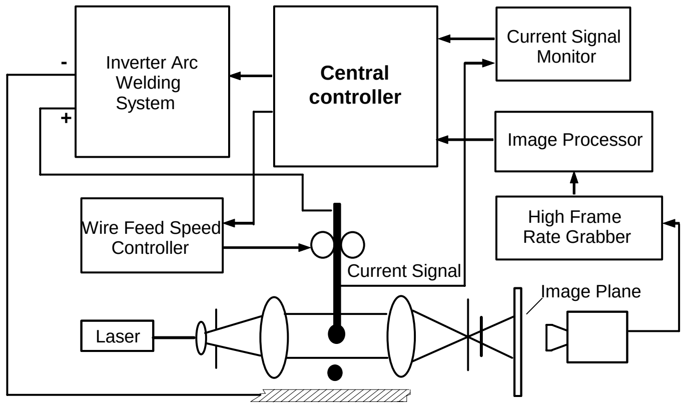


Fig. 1-2 Laser back-lighting technique[14]

Another widely used method of obtaining clear metal transfer images is named as laser back-light technique. Its functional diagram is shown in Fig.1-2. The laser is projected by the left lens to travel toward droplet/wire and then reaches the image plane as a set of parallel lights. Lights blocked by droplet/wire will not reach the right lens and the image plane. The rest of laser lights will illuminate the image plane and be viewed by the camera. It is known that the intensity of arc light quickly decays as the travel distance increases. If the distance from the arc to the right lens is sufficient, the arc light will be weaker than the parallel laser lights on the image plane. Consequently, the camera can image the geometry of droplet and wire. Fig. 1-2 shows a sequence of metal transfer images captured using the laser back-lighting imaging system with an average current of 165A. It is seen that the acquired image is quite clear and easy to be analyzed. This technique is quite important especially before the availability of digital high speed video cameras. At that time, the film cameras are widely used to capture the metal transfer images and analyzed by human eyes directly. As a result, the requirement for the quality of the images on the imaging screen is very high. With the spread of digital cameras and development of image processing techniques, the requirement for the quality of the direct imaging becomes low because image processing has robust processing ability and can process low quality images with high accuracy. Moreover, due to the complexity of the back-light imaging technique, it is very inconvenient to implement it. So it is too complex for practical use. As a result, a much easy implementing laser imaging system to image the metal transfer process is proposed in this research work.

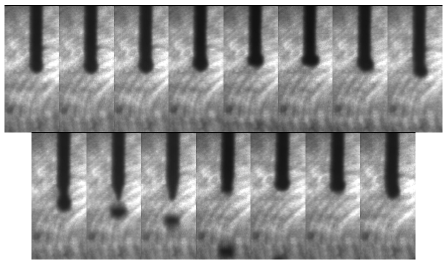


Fig. 1-3 GMAW metal transfer process with 165 A of average current

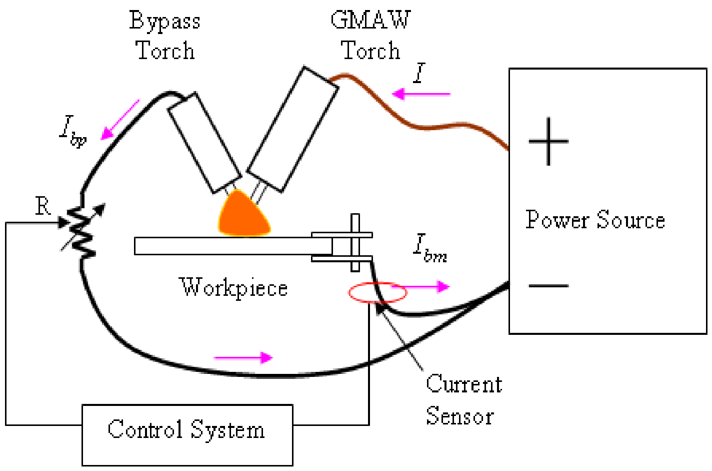


Fig. 1-4 DE-GMAW system

The novel DE-GMAW process[15] has been implemented by adding a non-consumable tungsten electrode to a conventional GMAW system to form a bypass loop as shown in Fig.1-4. Because of the bypass loop, only part of the current which melts the electrode flows to the base metal. The melting current I is thus divided into two branches: bypass current I_{bp} from the GMAW wire to the bypass electrode and base metal current I_{bm} from the GMAW wire to the base metal (Fig.1-4). While the total (melting) current I can be controlled by adjusting the wire feed speed, the University of Kentucky has developed a control system[15] to adjust the base metal current at a desired level through adjusting the resistance of the variable power resistor (Fig.1-4). As a result, the base metal current and bypass current can be controlled freely and independently. This controllability gives DE-GMAW a capability to reduce the base metal input (determined primarily by the base metal current) while the deposition rate (determined by the wire feed speed or primarily by the total current) and the capability to control the metal transfer process by adjusting the distribution of the total current in the base metal and bypass currents. Experiments have revealed for example that when a fixed total current below the so-called transition current which is approximately 225 amps is used, the transfer changes from the undesirable globular mode to the desired spray mode after the bypass current increases to 50 amps. In addition, the diameter of the droplets decreases as the bypass current increases but the transfer rate (number of the droplets per second) increases. The stability and metal transfer of DE-GMAW can thus be accurately monitored using high speed images and be controlled by adjusting the current distribution or the waveform of the current distribution using the high speed images as the feedback. Hence, the problem of controlling the size of the droplet (thus the transfer rate of the droplet because the melting rate is given) in real-time arises correspondingly. An effective image processing algorithm which can automatically compute the size and position of droplets is thus needed for advanced control of metal transfer in DE-GMAW. Since the frame rate required to capture the metal transfer video[16]–[19] is too high for current frame grabber, the real time processing goal can not be achieved now. As a result, the automated image processing of this kind of images is treated as the foundation for the next generation intelligent welding control. Although the real time processing system of DE-GMAW metal transfer process is not implemented, analysis has been performed to demonstrate the feasibility of integrating the developed automated image processing algorithms to the real time processing system for the next generation intelligent welding control.

On the contrary, the metal transfer produced by GMAW has a less frequent drop rate. A frame grabber with 420 frames per second rate can meet this drop rate and accordingly the information of the detach rate of the droplet can be obtained. Dr Yu-Ming Zhang of University of Kentucky had performed the real time analysis of this kind of metal transfer images using the laser back-light imaging technique. In this research work, a different imaging processing based methodology is proposed to image the metal transfer process of GMAW. The corresponding automated image processing algorithms have also been developed. The real time processing system is implemented by integrating the proposed image processing algorithms. Experiments verified that the real time processing can locate the droplet and compute its size and detach rate automatically with adequate accuracy.

1.2 Objectives

As discussed in the previous section, there are two main objectives of this research work. The first objective is to develop the automated image processing algorithms for the metal transfer process of DE-GMAW to establish the foundation for the next generation intelligent welding control. Fig. 1-5 shows a typical metal transfer image in DE-GMAW with two droplets existing simultaneously. Because a high frame rate must be used for metal transfer process monitoring, the resolution of the resultant image is typically low and the effective area of image (on the sensor chip) is relatively small although the image can be read by careful readers. As can be seen, the central part of the image is the welding arc, on which a bright droplet is identifiable. The image processing algorithms to be developed in this research work needs to locate the droplet in the image (*droplet position*) and calculate *its area* in the image as the *droplet size*. After the *droplet position* is computed, the *detach rate* and *detach angle* can be determined easily.

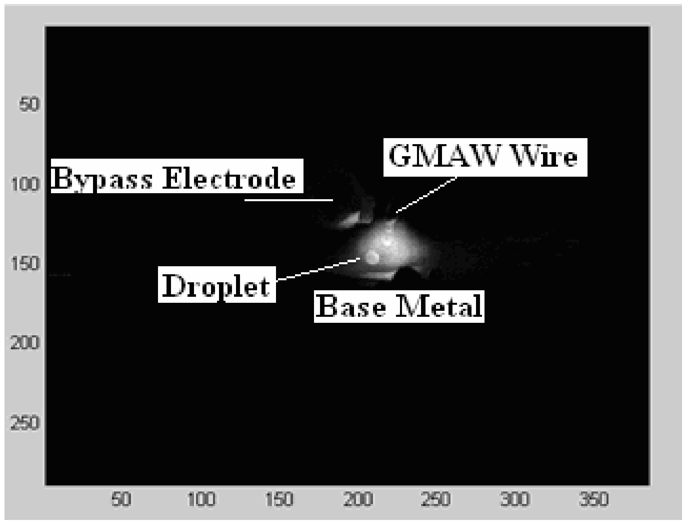


Fig. 1-5 Typical metal transfer in DE-GMAW

The second objective is to develop automated image processing algorithms for the GMAW metal transfer process and implement the real time processing system to perform on-line analysis. Although the well-known laser back-light technique can acquire clear metal transfer images, it is too complex for practical use. In this research work, a image processing based methodology of acquiring the metal transfer images is proposed and used. Fig. 1-6 shows three typical metal transfer images captured by the proposed methodology. The first image is named as insignificant tapering with droplet, the second one is named as insignificant tapering without droplet and the third one is named as significant tapering. The developed automated image processing algorithms should not only compute the position and size of the droplet, but also compute the length and average width of the insignificant tapering and significant tapering. All these information can be used as feed back in the later control of the welding current to achieve desirable welding quality.

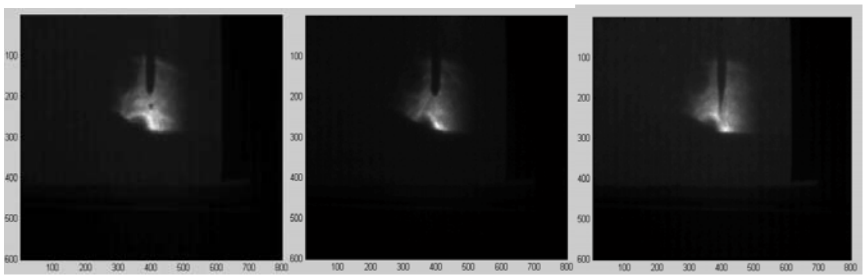


Fig. 1-6 Typical metal transfer images of GMAW

1.3 Challenges

There are three main challenges of this research. First, the quality of the captured image is relatively poor. Many factors including strong arc light and welding plasma make it impossible to capture very clear images of the metal transfer. Also, the resolution of the arc area or the droplet is relatively low. The quality of captured image is further reduced. In addition, the quality of the captured image also varies with the welding parameters. This makes it difficult for automatic processing. As a result, without proper enhancement technique, it is very difficult to separate the droplet from the background with adequate accuracy. Hence, proper enhancement is needed. However, there are a lot of enhancement techniques and an effective one must be found to improve the quality of the captured image. Second, there are several characteristic parameters (position, size and detaching rate of the droplet) which are to be calculated in real time and the computation for all of them is time-consuming. The image algorithms must be optimized or appropriately designed to be as fast as possible. It seems that tracking is the fastest algorithm to compute the position of the droplet, but all the tested tracking algorithms failed due to variety of the droplet and poor quality of the captured image. Moreover, the size of the droplet varies from frame to frame and it also needs to be computed. This contradicts with tracking algorithms. Third, since the computed characteristic parameters are used for feedback control, it must be accurate enough. Specific complex algorithms are needed to solve the specific problems. These proposed algorithms are required to be both robust and fast enough.

1.4 Organization

This dissertation is organized as follows. In Chapter 2, three existing traditional tracking methods, Correlation, Least Square Matching and Kalman Filtering, are reviewed and experiments are conducted to evaluate their effectiveness and defectiveness. In Chapter 3, a novel automated image processing algorithm named Brightness Based Separation Algorithm is proposed and evaluated. Experimental results are analyzed. In Chapter 4, a second automated algorithm named Brightness and Subtraction Based Separation Algorithm is proposed and evaluated. Experimental results are illustrated to demonstrate the effectiveness of this original algorithm. In Chapter 5, the third novel automated image processing algorithm named Brightness Based Selection and Edge Detection Based Enhancement Separation Algorithm is proposed and evaluated. Experimental results are also illustrated to show that this proposed algorithm is very effective in locating the droplet and computing its size. In Chapter 6, the processing results using the developed automated software are analyzed. In Chapter 7, the proposed image processing based methodology of monitoring the GMAW metal transfer process is evaluated and the corresponding image processing algorithms are proposed. In chapter 8, the implemented real time processing system is evaluated. Finally, in chapter 9, conclusions are drawn and contributions are described.

CHAPTER 2

REVIEW OF TRADITIONAL METHODS OF TRACKING

By observing the DE-GMAW metal transfer images, it is seen that the droplet always detaches in a specific track, which is determined mainly by the gravity of the droplet and the blow force of the by-pass electrode. Moreover, the droplet has a higher grayscale value than most other parts. So an effective tracking algorithm may locate the position of the droplet very fast with adequate precision. In this research work, several famous tracking techniques are applied. In the following sessions, they will be discussed in detail.

2.1 Correlation

Correlation is a traditional way of tracking which needs to define a model first. Then the model can be tracked in real time by correlation whose equation is given in Eq. (2-1) below:

$$f(x,y)\circ h(x,y)=\frac{1}{MN}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}h^{*}(m,n)\,f(x+m,y+n) \tag{2-1}$$

Where h is the function of the defined model and h^{*} denotes its complex conjugate. In our case, h is a real function (image) thus $h^{*}=h$. f is the captured image that has metal transfer droplets. M and N indicate the size of the defined model. Since the droplet varies a little bit from frame to frame, the model is difficult to define. In this work, the author choose $M=10$ and $N=10$.

To experiment, the author define the model as a disk with radius equal to 5 as shown in the center part of Fig. 2-1 based on observing the average size of the droplets. Fig.2-2 shows the resultant image of correlation. According to the correlation theory[20][21], the point where the correlation response reaches the peak (132,214) is the position of the droplet. Unfortunately, since the droplet might not be the brightest part of the complete image and the brightness among the droplet is not uniformly distributed, this method does not result in an adequate accuracy. For the metal transfer image shown in Fig.1-2 the correlation peak is at (132, 214), while the observed position of the bottom droplet is at (143, 215) and the observed position of the top droplet is at (130, 218). This computed peak position does not match either of the droplets, so it fails.

It is possible that the edge correlation method[22][23] might improve the accuracy. However, due to the variety of the droplet's shape and size, it is also impossible to define finite models to track the droplet effectively. In addition, the resolution of the captured image is relatively poor. Hence, the detected edge information is too little for accurate tracking.

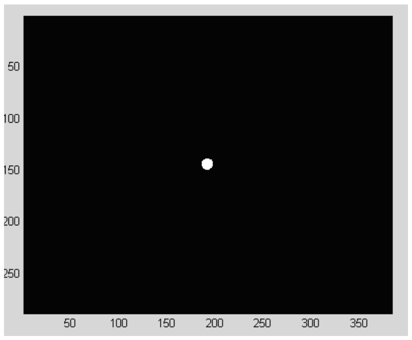


Fig.2-1 Defined droplet model

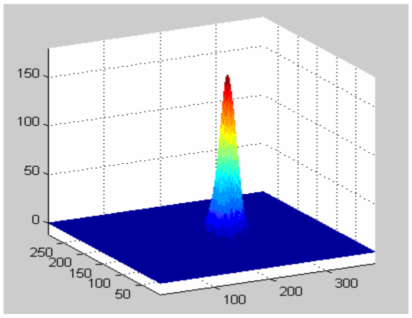


Fig.2-2 Correlation response

2.2 Least Square Matching (LSM) Tracking Algorithm

Least Square Matching Algorithm is the most accurate image matching technique. It has been widely used in tracking moving objects in sequential images due to its simplicity and effectiveness. In this research work, Least Square Matching Algorithm is tried to track the droplet's positions in sequential images.

2.2.1 Identification of the initial position of the droplet

Before tracking the droplet, the initial position of the droplet should be computed with enough accuracy. The following identification algorithm is used to find the initial position of the droplet and it can be divided into three steps.

Step1: Preprocessing

This step is to separate the objects of interest completely from the background which is completely dark. To this end, the image is binarized using the following equation:

$$f^{'}(x,y)=\begin{Bmatrix} 1, & if & f(x,y)>T \\ 0, & if & f(x,y)<T \end{Bmatrix} \tag{2-2}$$

where the threshold T is determined by the histogram of the image $f(x,y)$. Fig.2-3 shows the histogram of the example image to be binarized and the marked place indicates the threshold used to separate the image. The lowest point next to the right part of the peak is chosen as the threshold in this case. Fig. 2-4 shows an original image in (a) and its binarized image in (b). The binarized image is then filtered to eliminate unwanted noises to ease the computation of the original position of the droplet. In this step, the largest bright area is selected as S and then the filtered image $\hat{f}(x,y)$ is determined as

$$\hat{f}(x,y)=\begin{Bmatrix} f^{'}(x,y), & if & (x,y)\in S \\ 0 \quad\quad\quad , & if & (x,y)\notin S \end{Bmatrix} \tag{2-3}$$

The filtered image is shown in Fig. 2-4(c).

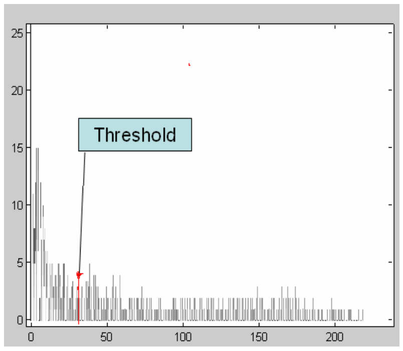


Fig. 2-3 Histogram of example image

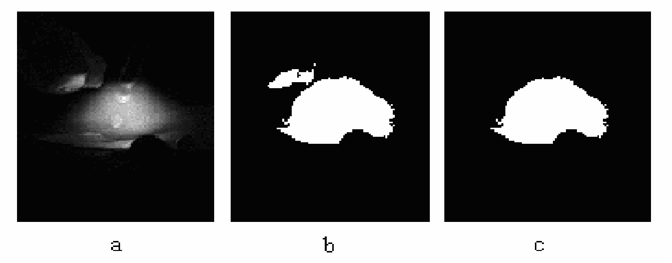


Fig. 2-4 Pre-processing of an example image a).original image. b). after binarizing c). after filtering

Step 2: Identification of the wire tip

The droplet is formed at and detached from the tip of the wire. In the image, it is located at the top of the welding arc. To identify it, the coordinate of the topmost point of the welding arc needs to be calculated. Since the image has been binarized, the computation appears straightforward. Its position, denoted as p, can be computed using the equation:

$$p = (\min(x), y), \; for \quad (\hat{f}(x-1, y) = 0) \,\&\, (\hat{f}(x, y) = 1) \tag{2-4}$$

The computed coordinate for the image in Fig. 1-5 is (122, 224). Since the arc almost remains constant during the welding, this coordinate can be used for the whole video sequence.

Step 3: Identification of the initial position of the droplet

After the position of the tip of the wire is identified, a specific value is added to it based on experiments and observation to acquire the initial position of the droplet. The author has chosen the specific value as (13, -9), then the initial position of the droplet becomes (135, 215).

2.2.2 Least Square Matching[24]–[28] (LSM) Tracking

Fig. 2-5 shows eleven adjacent images captured during the DE-GMAW metal transfer process. The first frame is the image when the droplet is just detached and the following frames record the travel of the detached droplet. In each frame, there are two droplets. The upper one is the new droplet which is just produced and being developed to detach from the wire tip. The lower one is the droplet which has been detached and is traveling. The objective of tracking is to find out the position of the same droplet in different frames (from its birth till its evanishment), then the detach rate of the droplet can be known easily.

As can be seen, in addition to the travel from the wire toward the work-piece, the droplet also shifts in horizontal direction. In conventional GMAW, the droplet travels from the anode wire toward the cathode work-piece because of the detaching electromagnetic force pointing from the anode toward the cathode combined with the gravity of the droplet. In DE-GMAW, both the work-piece and the bypass electrode are cathode and there are two components of detaching electromagnetic forces pointing from the wire to the bypass electrode and work-piece respectively. Hence, in DE-GMAW, a horizontal motion can be observed for the traveling droplet. By observation, the distance of the droplet in two adjacent frames is always less than 2 pixels in x direction and y direction respectively. Based on these features, a search region of 3×3 along the moving direction of the droplet is sufficient enough to find the droplet in the next frame. Consequently, the LSM becomes much easier and more time-saving.

In this work, the LSM can be divided into 2 steps.

Step1: Defining a template droplet.

Since the droplet varies a little bit every frame due to the force it is bearing and some unknown factors, a template is defined to approximate the average of the same droplet shown in a series of frames. One example is shown in Fig.2-6. The template droplet is obtained by averaging the droplet in Frame 8, Frame 9, Frame 10 and Frame 11.

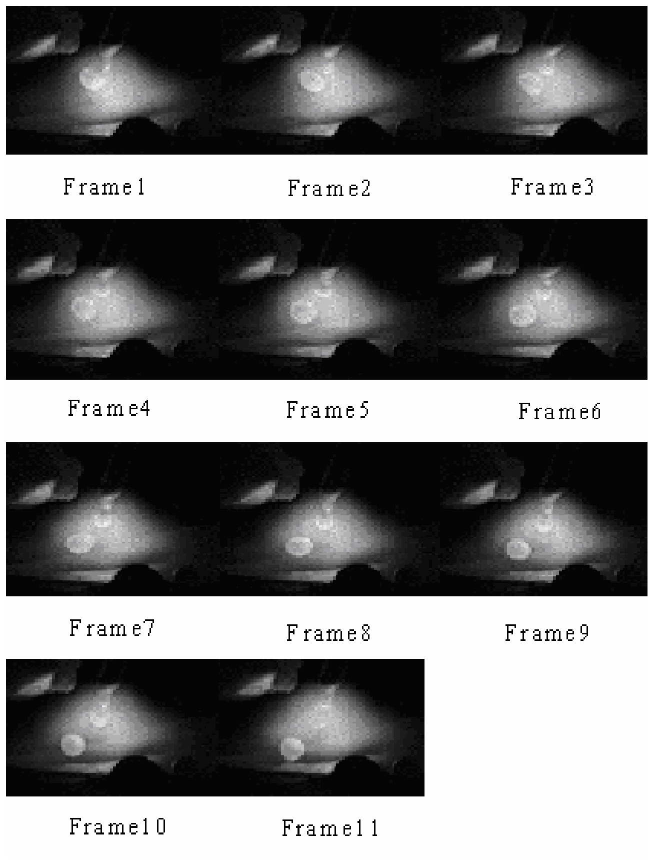


Fig. 2-5 Eleven adjacent images

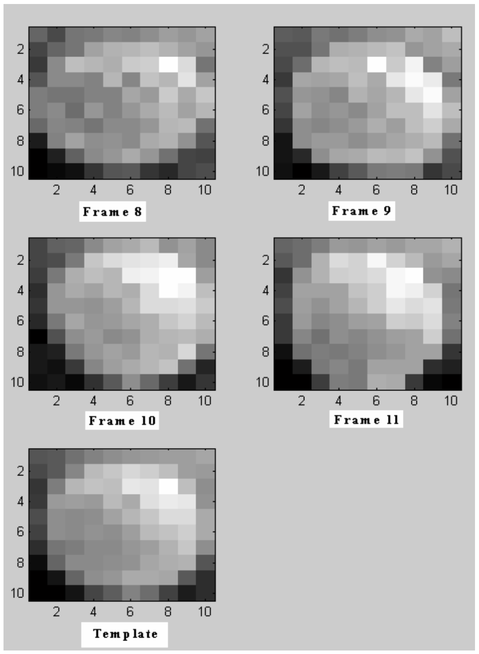


Fig. 2-6 The droplet in four adjacent frames and its template

Step2: Track the droplet based on the least square error

Search along the bottom left direction within a region of 3×3 and calculate the mean square error of each position using the following equation.

$$MSE = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}\left(h(i,j)-g(i,j)\right)^2 \tag{2-5}$$

Where h is the template droplet and g is the 10×10 area on the arc that may contain the droplet. Among the nine positions, the one with the least square error is the position of the droplet.

The four frames shown in Fig. 2-8 are used to test the LSM tracking algorithm. Every time the exact position of the droplet will be found without any error. Furthermore, LSM is really fast. It takes less than 1 second to track the four frames in MATLAB, which means that it can meet the real time requirement definitely. But in reality, the brightness distribution within the droplet is varying and the shape and size of the droplet are also varying. Hence, it is impossible to define a universal template or finite number of templates for all the droplets produced during the metal transfer process. As a result, cumulative errors will increase for ever. So LSM can not be used in tracking this low quality and varying shape/size object.

2.3 Kalman Filtering Tracking[29]–[34]

The Kalman Filtering is an efficient recursive filter which estimates the state of a dynamic system from a series of incomplete and noisy measurements. It has two distinct phases: Predict and Update. The predict phase uses the estimate from the previous time-step to produce an estimate of the current state. In the update phase, measurement information from the current time-step is used to refine the prediction to arrive at a new and more accurate estimate.

The movement of the droplet can be treated as a dynamic system and its state is the position and the speed of the droplet. The droplet is affected by a number of forces including its gravity, arc pressure, electromagnetic force pointing from the anode toward the cathode, the surface tension and the friction. The joint force gives the droplet an acceleration in both x and y directions. This acceleration is assumed as normally distributed with mean m_a and standard deviation σ_z. From Newton's laws of motion, it is concluded the relationship of the x coordinate x_k and y coordinate y_k in the kth frame with those in the $(k-1)$th frame can be represented by the following equation:

$$\begin{bmatrix} x_k \\ y_k \\ \dot{x}_k \\ \dot{y}_k \end{bmatrix} = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{k-1} \\ y_{k-1} \\ \dot{x}_{k-1} \\ \dot{y}_{k-1} \end{bmatrix} + \begin{bmatrix} \Delta t^2/2 & 0 & 0 & 0 \\ 0 & \Delta t^2/2 & 0 & 0 \\ 0 & 0 & \Delta t & 0 \\ 0 & 0 & 0 & \Delta t \end{bmatrix} \begin{bmatrix} a_x \\ a_y \\ a_x \\ a_y \end{bmatrix} \tag{2-6}$$

Where $\dot{x}_k$ is the speed of the droplet in x direction in the k*th* frame and $\dot{y}_k$ is the speed of the droplet in y direction in the k*th* frame. x_{k-1} is the x coordinate of the droplet in the (k-1)*th* frame and y_{k-1} is the y coordinate of the droplet in the (k-1)*th* frame. $\dot{x}_{k-1}$ is the speed of the droplet in x direction in the (k-1)*th* frame and $\dot{y}_{k-1}$ is the speed of the droplet in y direction in the (k-1)*th* frame. Δt is the time interval between the k*th* frame and the (k-1)*th* frame. a_x is the acceleration along the x direction and a_y is the acceleration along the y direction.

The predicted estimate covariance for the k*th* frame is written as follows:

$$P_k = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} P_{k-1} \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}^T + Q_k \tag{2-7}$$

Where

$$\begin{aligned} Q &= \operatorname{cov}\left(\begin{bmatrix} \Delta t^2/2 & 0 & 0 & 0 \\ 0 & \Delta t^2/2 & 0 & 0 \\ 0 & 0 & \Delta t & 0 \\ 0 & 0 & 0 & \Delta t \end{bmatrix} \begin{bmatrix} a_x \\ a_y \\ a_x \\ a_y \end{bmatrix} \right) \\ &= E\left\{ \left(\begin{bmatrix} \Delta t^2/2 & 0 & 0 & 0 \\ 0 & \Delta t^2/2 & 0 & 0 \\ 0 & 0 & \Delta t & 0 \\ 0 & 0 & 0 & \Delta t \end{bmatrix} \begin{bmatrix} a_x \\ a_y \\ a_x \\ a_y \end{bmatrix} \right) \left(\begin{bmatrix} \Delta t^2/2 & 0 & 0 & 0 \\ 0 & \Delta t^2/2 & 0 & 0 \\ 0 & 0 & \Delta t & 0 \\ 0 & 0 & 0 & \Delta t \end{bmatrix} \begin{bmatrix} a_x \\ a_y \\ a_x \\ a_y \end{bmatrix} \right)^T \right\} \\ &= \begin{bmatrix} \Delta t^2/2 & 0 & 0 & 0 \\ 0 & \Delta t^2/2 & 0 & 0 \\ 0 & 0 & \Delta t & 0 \\ 0 & 0 & 0 & \Delta t \end{bmatrix} E\left\{ \left(\begin{bmatrix} a_x \\ a_y \\ a_x \\ a_y \end{bmatrix} \right) \left(\begin{bmatrix} a_x \\ a_y \\ a_x \\ a_y \end{bmatrix} \right)^T \right\} \begin{bmatrix} \Delta t^2/2 & 0 & 0 & 0 \\ 0 & \Delta t^2/2 & 0 & 0 \\ 0 & 0 & \Delta t & 0 \\ 0 & 0 & 0 & \Delta t \end{bmatrix} \end{aligned} \tag{2-8}$$

It is assumed that the initial position is perfectly precise. Then

$$P_0 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

At each frame, a noisy measurement of the droplet's position is made. The noise is also assumed to be normally distributed with mean 0 and standard deviation σ_z.

$$z_k = Hx_k + v_k \qquad (2\text{-}9)$$

Where z_k is the noise measurement for the droplet in the kth frame and v_k is normally distributed noise with standard deviation σ_z.

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}.$$

The Optimal Kalman Gain is the one that yields the minimum mean square errors and it is acquired using the following equation.

$$K_k = P_k H^T S_k^{-1} \qquad (2\text{-}10)$$

Where

$$S_k = HP_k H^T + \sigma_z^2 \qquad (2\text{-}11)$$

After the Optimal Kalman Gain is obtained, the state and predicted estimate covariance need to be updated using the following equations.

$$\begin{bmatrix} \hat{x}_k \\ \hat{y}_k \\ \hat{\dot{x}}_k \\ \hat{\dot{y}}_k \end{bmatrix} = \begin{bmatrix} x_k \\ y_k \\ \dot{x}_k \\ \dot{y}_k \end{bmatrix} + K_k v_k \tag{2-12}$$

$$\hat{P}_k = (I - K_k H) P_k \tag{2-13}$$

The eleven frames shown in Fig. 2-6 is used to test the Kalman filter tracking algorithm. Fig. 2-7 shows the tracking result. From this figure, it is seen that the Kalman Filter tracking algorithm causes unacceptable accuracy. Although it is very fast (it takes less than 2 seconds in MATLAB for the Kalman Filter to track these eleven frames), it is too unreliable for practical use in this research work.

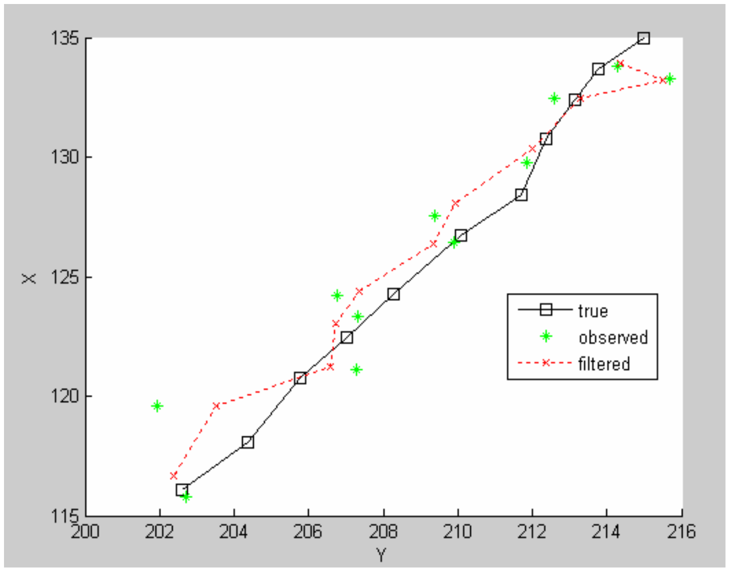


Fig. 2-7 The result of Kalman Filter tracking

In conclusion, tracking techniques are not suitable in locating the variable droplets in poor quality images. Other effective ways must be found to compute the droplet's size and position robustly and automatically.

CHAPTER 3

BRIGHTNESS BASED SEPARATION ALGORITHM

Based on the facts that droplets are brighter than most part of the welding arc in the image, a novel image processing algorithm named Brightness Based Separation Algorithm is proposed to compute the position and size of the droplet.

3.1 The Histogram-based-threshold Method

Before evaluating the proposed methods, the idea of a histogram-based thresh-holding method[35][36] (minimum error) is introduced first.

Suppose that an image contains only two principal gray-level regions. Let z denote gray-level values. These values can be viewed as random quantities, and their histogram can be considered as an estimate of their probability density function (PDF) $p(z)$. This overall density function is the sum or mixture of two densities, one for the light and the other for the dark regions in the image. Furthermore, the mixture parameters are proportional to the relative areas of the dark and light regions. If the form of the densities is known or assumed, it is possible to determine an optimal threshold T (in terms of minimum error) for segmenting the image into two distinct regions.

Fig.3-1 shows two probability density functions. The mixture probability density function describing the overall gray-level variation in the image is

$$p(z) = P_1 p_1(z) + P_2 p_2(z) \tag{3-1}$$

Where P_1 and P_2 are the probabilities of occurrence of the two classes of pixels. They have the following relationship.

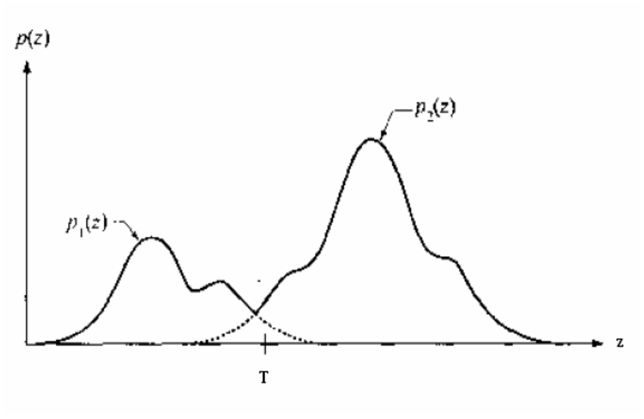


Fig. 3-1The mixture density function[37]

$$P_1 + P_2 = 1 \tag{3-2}$$

The probability of erroneously classifying a point belong to $p_2(z)$ as the point belong to $p_1(z)$ is

$$E_1(T) = \int_{-\infty}^{T} p_2(z)dz \tag{3-3}$$

Similarly, the probability of erroneously classifying a point belong to $p_1(z)$ as the point belong to $p_2(z)$ is

$$E_2(T) = \int_{T}^{\infty} p_1(z)dz \tag{3-4}$$

Then the overall probability of error is

$$E(T) = P_2 E_1(T) + P_1 E_2(T) \tag{3-5}$$

To find the threshold value for which this error is minimal requires differentiating $E(T)$ with respect to T (using Leibniz's rule) and equating the result to 0. The result is

$$P_1 p_1(T) = P_2 p_2(T) \tag{3-6}$$

This equation is solved for T to find the optimum threshold. The density function is treated as Gaussian density which is completely characterized by two parameters: the mean and the variance. In this case,

$$p(z) = \frac{P_1}{\sqrt{2\pi}\sigma_1} e^{-\frac{(z-\mu_1)^2}{2\sigma_1^2}} + \frac{P_2}{\sqrt{2\pi}\sigma_2} e^{-\frac{(z-\mu_2)^2}{2\sigma_2^2}} \tag{3-7}$$

Where μ_1 and σ_1^2 are the mean and variance of the Gaussian density of $p_1(z)$ and μ_2 and σ_2^2 are the mean and variance of $p_2(z)$. To simplify the computation, the variances are assumed to be equal

$$\sigma^2 = \sigma_1^2 = \sigma_2^2 \tag{3-8}$$

Combining Eq.3-6, Eq.3-7 and Eq.3-8, the optimal threshold is computed as

$$T = \frac{\mu_1 + \mu_2}{2} + \frac{\sigma^2}{\mu_1 - \mu_2} \ln\left(\frac{P_2}{P_1}\right) \tag{3-9}$$

This histogram-based thresh-holding technique is used in the proposed brightness based separation algorithm with P_1 and P_2 obtained in advance.

3.2 Brightness Based Separation Algorithm

This proposed algorithm which makes use of the high brightness of droplets can be divided into six steps. Fig. 3-2 is the example image that is used to illustrate the step-by-step results.

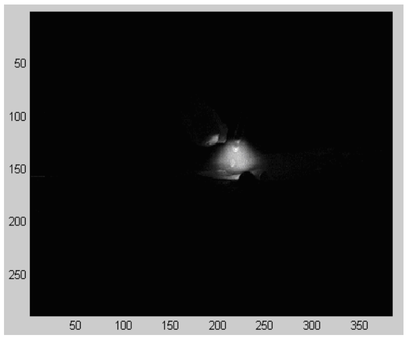


Fig.3-2 The example image used by the brightness based separation algorithm

Step1: Preprocessing

Step 2: Identification of the wire tip

Step 1 and Step 2 are completely the same as the first two steps of Least Square Matching Tracking Algorithm illustrated in Chapter 2.

Step 3: Binarization for droplet identification

In this step, the original image is re-binarized toward the identification of the droplet. To this end, Eq.2-2 is used again but with a higher new threshold which is determined based on the histogram of the metal arc area. Fig.3-3 shows the resultant binarized image.

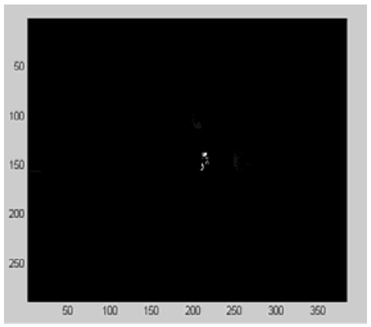


Fig. 3-3 Binarization toward droplet identification

Step 4: Image Filtering

Because multiple droplets may exist and the number of the droplets is unknown, a pre-specified threshold is used to determine if a particular bright area in the binarized image is a droplet or noise. The union of all bright areas which are larger than the threshold is defined as *S* and Eq.2-3 is then used to filer the image. The filtering result is demonstrated in Fig.3-4.

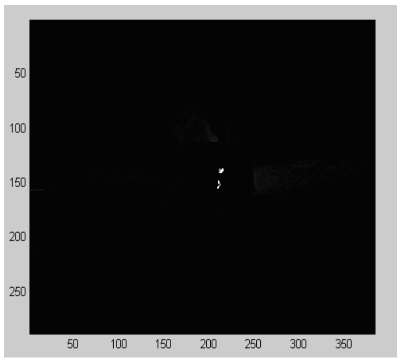


Fig. 3-4 Filtered binarized image

Step 5: Image dilating[37]–[39]

At this point, the droplet has been identified approximately. Next step is to approximate the computed droplet as much as possible to the real droplet through dilation. Before dilation is discussed, three basic definitions are introduced.

The reflection of set B, denoted $\hat{B}$, is defined as

$$\hat{B}=\{w \mid w=-b, \quad for \quad b\in B\} \tag{3-10}$$

The translation of set B by point $z=(z_1,z_2)$, denoted $(B)_z$, is defined as

$$(B)_z=\{c \mid c=b+z, \quad for \quad b\in B\} \tag{3-11}$$

The complement of a set B is the set of elements not contained in B

$$B^c=\{w \mid w\notin B\} \tag{3-12}$$

The dilation of A by B, denoted $A\oplus B$, is defined as

$$A \oplus B = \left\{ z \middle| \left(\hat{B} \right)_z \cap A \neq \varnothing \right\} \tag{3-13}$$

where A is the separated droplet in the filtered and binarized image and B is commonly referred to as structuring element[40]–[42] in dilation. This equation is based on obtaining the reflection of B about its origin and shifting this reflection by z. The dilation of A by B then is the set of all displacements, z, such that $\hat{B}$ and A overlap by at least one element. Based on this interpretation, Eq.3-13 can be rewritten as

$$A \oplus B = \left\{ z \middle| \left(\hat{B} \right)_z \cap A \subseteq A \right\} \tag{3-14}$$

Since the object to be dilated is a disk-like droplet, the following B is used

$$B = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix} \tag{3-15}$$

The result of binarization and dilation is shown in Fig.3-5 and Fig.3-6. Fig.3-5 is an example for one droplet case and Fig.3-6 is an example for two-droplet case.

Step 6: Size and position computation

The following equation is used to compute the droplet position (x_c, y_c) from the dilated image:

$$\begin{cases} x_c = \dfrac{x_{\max} + x_{\min}}{2} \\ y_c = \dfrac{y_{\max} + y_{\min}}{2} \end{cases} \tag{3-16}$$

where $x_{\min}$ and $x_{\max}$ are the minimum and maximum value of x coordinate of one droplet, and $y_{\min}$ and $y_{\max}$ are the minimum and maximum value of the y coordinate of the same droplet in the dilated image.

The equation used for computing the size, denoted S_{area}, is given below

$$\begin{cases} S_{area} = 0 \\ S_{area} = S_{area} + 1, \quad if \quad (x, y) \in S \end{cases} \tag{3-17}$$

where S represents the separated droplet in the dilated image.

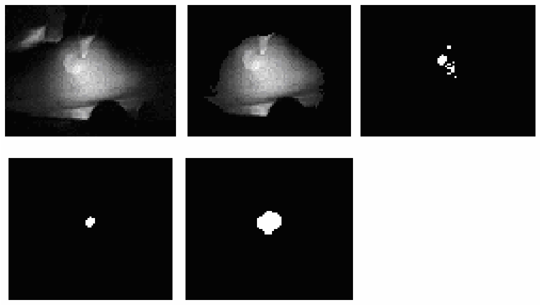


Fig. 3-5 Binarizing and saturation example for one droplet image

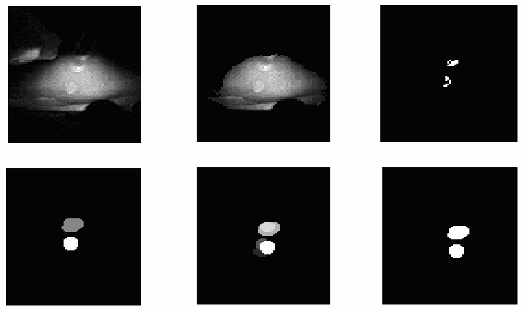


Fig. 3-6 Binarizing and saturation example for two droplets image

The computed position of the droplet from the last image in Fig.3-5 is (137, 216) and the observed position is (135, 215). The computed size is 88 and the observed size is 90. The computed position for the top droplet shown in the last image in Fig. 3-6 is (131, 219) and the computed position for the bottom droplet is (144, 218). The observed positions for the top and bottom droplets are (131, 218) and (144, 215) respectively. The computed sizes for the top and bottom droplets are 86 and 51 respectively, while the observed sizes of the corresponding droplets are 50 and 47.

CHAPTER 4

BRIGHTNESS AND SUBTRACTION BASED SEPARATION ALGORITHM

4.1 Analysis of the drawback of Brightness Based Separation Algorithm

The brightness based separation algorithm has a major drawback that is the brightness of the droplet is not evenly distributed, i.e. part of the droplet is very bright while other parts are kind of dark. This makes it difficult to compute the droplet size with a high accuracy. To resolve this issue, the author has tried to enhance the difference between droplets and other parts in the image. To this end, several popular image enhancement algorithms [43]–[57] , e.g. histogram equalization [49]–[52] , Laplacian [53]–[56] , homomorphic filtering [57] have been tested. Unfortunately, all these algorithms failed due to the poor resolution of the captured image. Hence, the Brightness and Subtraction Based Separation Algorithm is proposed.

4.2 Brightness and Subtraction Based Separation Algorithm

The Brightness and Subtraction Based Separation Algorithm is based on the Brightness Based Separation Algorithm but it makes full use of the high correlation relationship between adjacent frames. It is composed of eight steps. The first two steps are completely the same with those for the Brightness Based Separation Algorithm. The rest six steps are as follows.

Step 3: Subtracting

This step is to subtract every frame by a reference frame which can be chosen as the first frame of a life cycle series of the metal transfer.

It is obvious that the adjacent frames are very similar and the development of the droplets is the main cause for this difference. As a result, if the difference can be obtained, the development of the droplet would be able to be acquired. It is straightforward that subtraction can make the difference become obvious. Hence, the following equation has been defined to perform subtraction:

$$Diff_i(x,y) = f_i(x,y) - f_R(x,y) \tag{4-1}$$

where f_i denotes the image of frame i and f_R denotes the image of reference frame. One example of the image $Diff_i$ is shown in Fig.4-1. *Please be noticed that the reference frame should be chosen with the least droplet on it, e.g. the first frame in the life span of the droplet to guarantee that the value of the droplet part is positive after subtraction.*

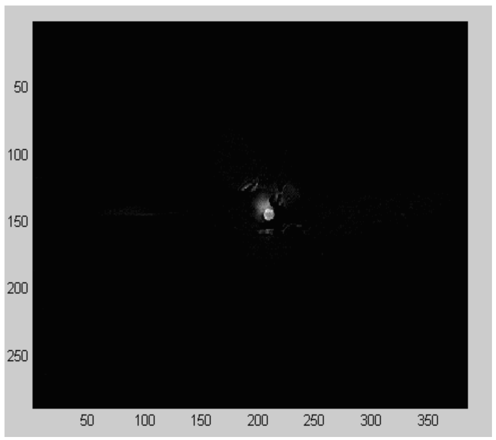


Fig. 4-1 Example of image $Diff_i$

Step 4: Binarizing

In this step, the image $Diff_i$ is binarized. Eq.3-10 is used again but with a higher new threshold which is determined based on the histogram of the part with value greater than

0 in the $Diff_i$ image and Fig. 4-2 shows its histogram. Eq.3-9 is used to get the threshold which is marked in Fig. 4-2. Fig. 4-3 shows the resultant binarized image.

As can be seen, after binarization, the droplet becomes much more obvious and the noise in $Diff_i$ is reduced a lot.

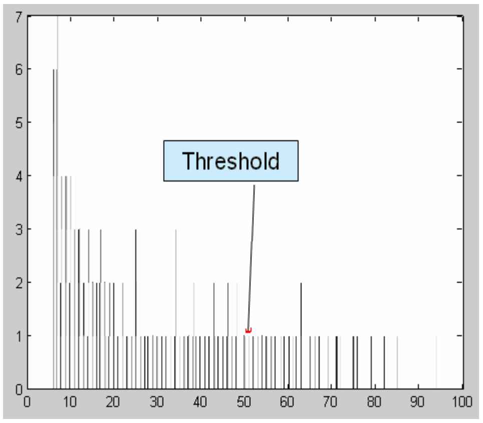


Fig. 4-2 Histogram of the image after subtraction

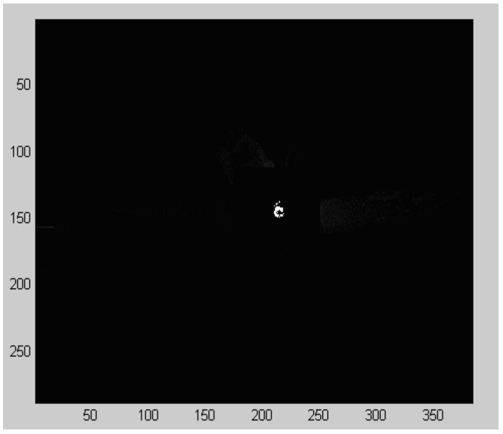


Fig. 4-3 Binarized $Diff_i$

Step 5: Binarized image filtering

As can be seen, the droplet can be now clearly observed in the binarized image. In this step, the binarized $Diff_i$ is filtered using Eq.3-11. The result is shown in Fig.4-4.

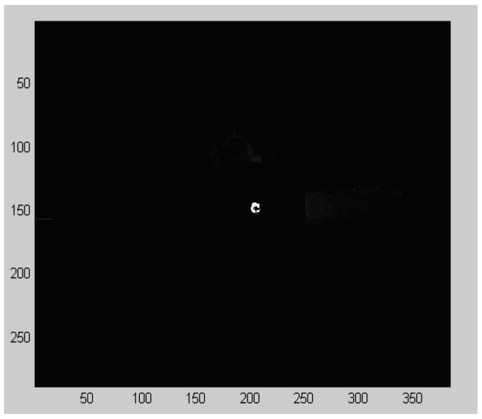


Fig.4-4 Filtering result of binarized $Diff_i$

Step 6: Region filling and bilinear interpolating

There might be holes or cracks inside the filtered droplet. At this time, region filling algorithm can be used to fill the holes and cracks. To this end, the part of the droplet with value 1 is defined as the boundary and the part with value 0 is defined as the region to be filled. Beginning with a point p inside the boundary, the objective of filling is to fill the entire region with 1's.

If the convention that all non-boundary (background) points are labeled 0 is adopted, value 1 can be assigned to p to begin. The following procedure then fills the region with 1's:

$$X_k = (X_{k-1} \oplus E) \bigcap D^c \qquad k = 1,2,3... \tag{4-2}$$

Where $X_0 = p$, D denotes the filtered image and E is the symmetric structuring element. In our case, E is chosen as

$$E = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

This algorithm terminates at iteration step k if $X_k = X_{k-1}$, i.e., the set union of X_k and D contains the filled set and its boundary. Fig.4-6 shows the droplet after region filling.

The bilinear interpolation[58][59] method can also be used for the filling the holes and cracks, its equation is defined as

$$f(x, y) = ax + by + cxy + d \tag{4-3}$$

Where x and y denote the coordinates of the point on the droplet, and a, b, c and d are the coefficients. It is straightforward that using four neighbors of the point in question one can obtain the coefficients. Then after substituting the coordinate of the point in question into Eq.4-3, its pixel value $f(x, y)$ can be computed. After every step of bilinear interpolation, if the pixel value is greater than 0.5, it is set to 1.

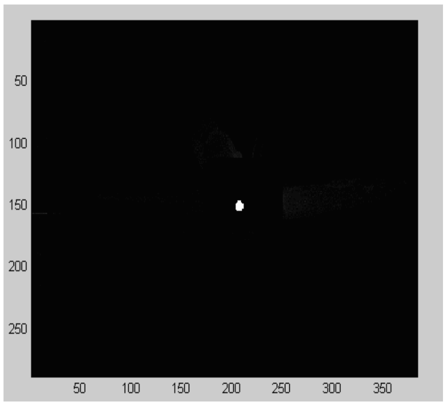


Fig. 4-5 Result after region filling

Step7: Dilating

In this step, the separated droplets are dilated as much as possible to the actual ones. Eq.3-17 is used again. In addition to adopting the previous disk structuring element

$$B_1 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

, another structuring element $B_2 = 1$ is introduced. The droplet is dilated separately using these two different structuring elements. Fig.4-6 shows the dilated droplet using structuring element B_1 and Fig.4-7 shows the dilated droplet using structuring element B_2.

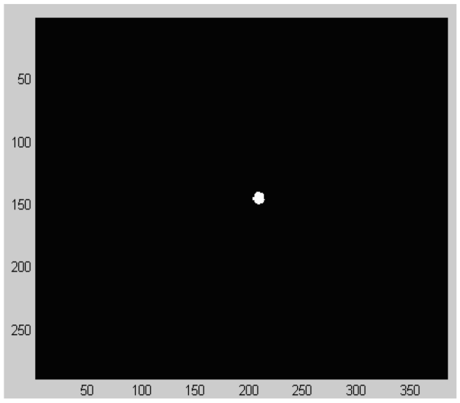


Fig. 4-6 Dilated droplet using structuring element B_1

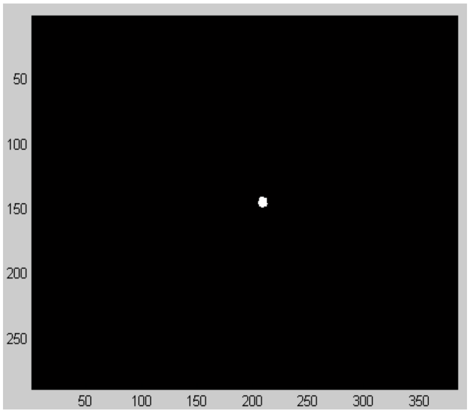


Fig. 4-7 Dilated droplet using structuring element B_2

Step8: Adaptive Computing

The computed dilated droplets' sizes based on the two structuring elements B_1 and B_2, denoted as S_1 and S_2 respectively, are compared with the pre-knowledge of the size of the droplet respectively. Here the authors choose the pre-knowledge of the size as 70.

If $S_2 < 70 < S_1$

$$S = (S_1 + S_2)/2 \tag{4-4}$$

If $S_1 < 70$

$$S = S_1 \tag{4-5}$$

If $S_2 > 70$

$$S = S_2 \tag{4-6}$$

If there is no pre-knowledge of the size of the droplet, Eq.4-4 is used directly to compute the droplet's size. A much more precise result can still be achieved.

Eq.3-19 is used again to compute the position of the droplet. Here only the dilated droplet based on structuring element B_1 is used to compute the position, since the result is the same no matter B_1 or B_2 is used. The computed position and size of the dilated droplet shown in Fig.4-1 are (146, 208) and 66 respectively, while the observed position is (146, 208.5) and the observed size is 67.

4.3 Results and Discussion

A series of adjacent frames of DE-GMAW metal transfer images have been used to test the Subtraction Based Droplet Separation algorithm. The used series are shown in Fig. 4-8. It contains 15 frames which describe the life span of a droplet. The first frame is the image when the droplet is just detached and the following frames record the travel of the detached droplet.

The image processing results for Frame 8 are shown in Fig.4-9. The separated droplet is shown in the last picture in Fig. 4-9. Since the droplet is enhanced and separated, it is natural that it retains the original position. The error in position computation using Eq.3-

19 can only be caused by the shape of the droplet. Even if the size is not the same as the original one, the position can be computed correctly as long as the algorithm is symmetric in recovering size. It is apparent that the separated droplet not only retains the original position but also retains the shape. As the experimental results verified, the size of the droplet can also be restored correctly.

The image processing results for Frame9, Frame10, Frame11, Frame12 and Frame13 in Fig.4-8 are shown in Fig.4-10, Fig.4-11, Fig.4-12, Fig.4-13 and Fig.4-14 respectively. It can also be seen that all the separated droplets preserve their actual shapes well. This indicates that the size of the droplet can be recovered accurately.

Tab.4-1 shows these frames' observed positions and sizes against their computations. It is seen that the positions can be recovered almost with no errors. Also the errors for the computed sizes can be ignored.

Since the interest in computing the size and position of the droplets is to calculate the detachment rate and average droplet size, computing part of the frames appears being adequate to fulfill the task.

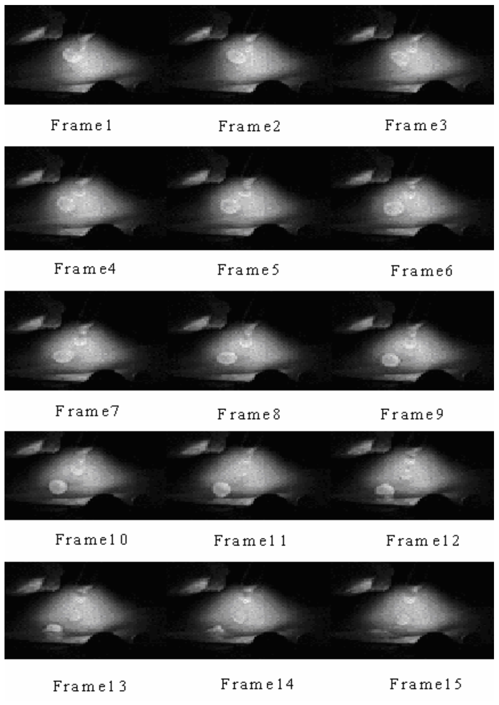


Fig. 4-8 Fifteen adjacent frames

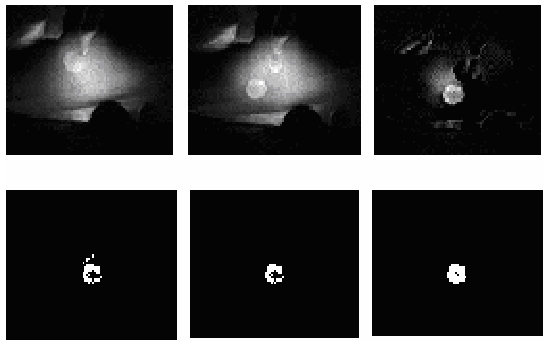


Fig.4-9.The result of processing the droplet shown in frame 8

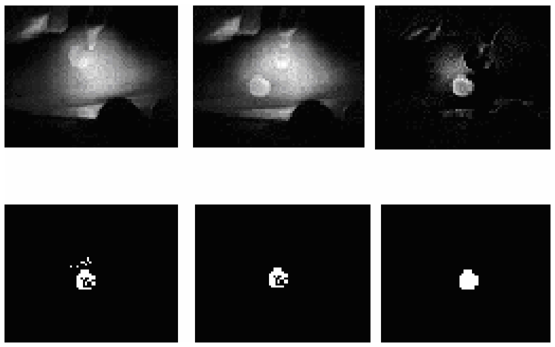


Fig.4-10. The result of processing the droplet shown in frame 9

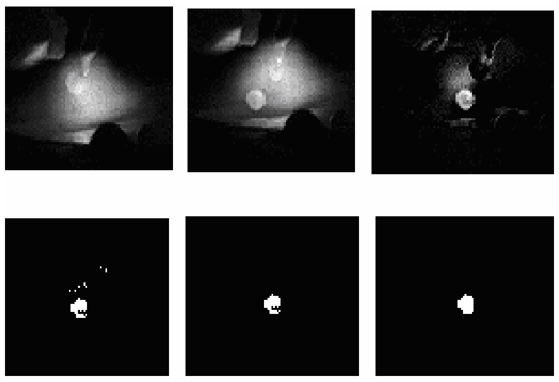


Fig.4-11. The result of processing the droplet shown in frame 10

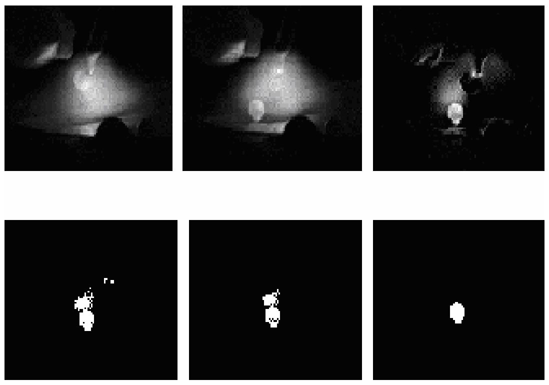


Fig.4-12. The result of processing the droplet shown in frame 11

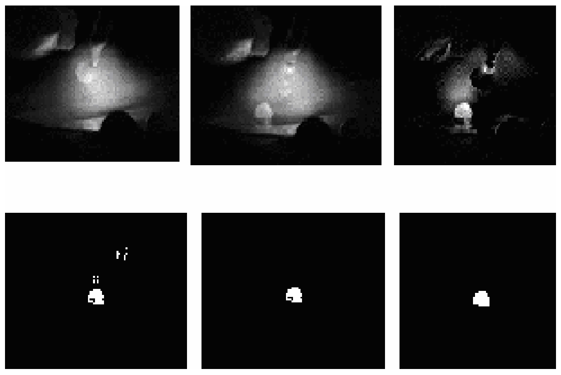


Fig.4-13.The result of processing the droplet shown in frame 12

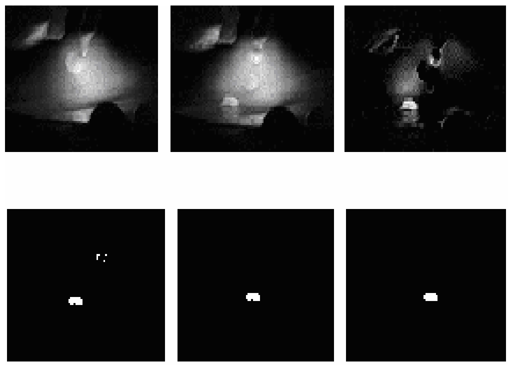


Fig.4-14.The result of processing the droplet shown in frame 13

Tab.4-1 Comparison of computed and actual droplets

Index of Frames	**Size of droplet**	**Coordinates Of droplet**	**Computed Size**	**Computed coordinates**	**Error of size**	**Error of Coordinates**
Frame8	67	(144.5, 209)	66	(144.5, 209)	1	(0, 0)
Frame9	67	(146, 208.5)	68	(146, 208)	1	(0, 0.5)
Frame10	69	(147.5, 208)	70	(147, 208)	1	(-0.5, 0)
Frame11	75	(149, 207)	76	(149, 207)	1	(0, 0)
Frame12	61	(149, 206.5)	61	(149,207)	0	(0, 0.5)
Frame13	36	(149.5, 205.5)	36	(149.5, 206)	0	(0, 0.5)

Testing shows that the proposed Brightness and Subtraction Based Separation Algorithm needs less than 3 seconds to process one frame in MatLab (The used computer is a notebook computer with Intel® Pentium® M 1.86GHz processor and 1 Gbytes Ram). Since the final real time processing system will be programmed in C, the processing time will be significantly reduced. Luo et al. [60] reported an over 100 times of speed improvement of compiled C over the interpreted MatLab. In reference [61], Tommiska et al. stated "This indicated complied (i.e., C source) code runs approximately 400 times faster than interpreted (i.e., MatLab) code" in their case. In addition, a multiple parallel processor system can be used to further reduce the processing time if necessary. Hence, the proposed Brightness and Subtraction Based algorithm appears to have the potential to be used in on-line control of metal transfer process, which will be described in the following session.

4.4 On-line Control Ability Analysis

To be used in on-line control, a sequence of images is acquired *first*. After the image acquisition is completed, *then* the sequential images are processed by the proposed Brightness and Subtraction Based algorithm and the resultant positions and sizes of the droplets in sequential images are used to determine the trajectory, detach rate and average droplet size as the process feedback. *Finally*, the control algorithm uses the process feedback to determine how the welding parameters should be adjusted to achieve the desired trajectory and droplet size. Please notice (1) it is not desirable that the welding parameters be adjusted rapidly and (2) it is a *wrong impression* that the images must be processed at the same speed as they are acquired in order to be used in on-line control. The needed image processing speed is typically much lower than that of the image acquisition.

To demonstrate how to determine the required image processing speed, let's consider a typical example where: (1) the transfer rate is 100 Hz; (2) 5 frames are used to track the life span of a droplet in order to accurately calculate the trajectory and size of the droplet; (3) the control rate is 2 Hz. In this typical example, 20 ms of time period can be used to acquire 10 images to assure that at least one droplet's entire life span is completely recorded. The image acquisition rate should be 500 Hz. For the 2 Hz control rate, the allowed image processing time for the 10 frames of image can be up to 470 ms because the control algorithm computation time can be negligible. Hence, the image processing speed needed would be approximately 47 ms per frame.

Based on the above analysis, the authors identified the problem to be solved in this study as the development of the image processing algorithms which have the potential to be implemented to extract the droplets from the DE-GAMW metal transfer image and perform related computations at a speed of 47 ms or less per frame.

CHAPTER 5

BRIGHTNESS BASED SELECTION AND EDGE DETECTION BASED ENHANCEMENT SEPARATION ALGORITHM

5.1 **Resolution Improvement by Bilinear Interpolation**

Edge detection is an important method for image processing. However, it may have difficulties to be effectively applied if the resolution of the image is low. Because of the need for high frame rate, the resolution of the metal transfer images is typically low. There are two possible ways to increase the resolution for metal transfer images. One is to change the CCD camera to a higher resolution through reducing the frame rate. Another is to increase the image resolution by bilinear interpolation. Since high frame rate is a must in recording DE-GMAW metal transfer images, the hardware becomes the constraint. As a result, the bilinear interpolation method is adopted to increase the resolution in order to effectively use the edge detection technique.

The bilinear interpolation for point (x , y) to be interpolated is performed using

$$f(x,y) = ax + by + cxy + d \tag{5-1}$$

Where the coefficients a , b , c and d are obtained using the four neighbors as:

$$\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} x-1 & y-1 & (x-1)(y-1) & 1 \\ x+1 & y-1 & (x+1)(y-1) & 1 \\ x-1 & y+1 & (x-1)(y+1) & 1 \\ x+1 & y+1 & (x+1)(y+1) & 1 \end{bmatrix}^{-1} \begin{bmatrix} f(x-1,y-1) \\ f(x+1,y-1) \\ f(x-1,y+1) \\ f(x+1,y+1) \end{bmatrix} \tag{5-2}$$

Once the bilinear interpolation is done for all possible points in the area of interest, the resolution will be doubled. The bilinear interpolation can then be used again on the image which has been interpolated to further increase the resolution. In this research work, the author chose to increase the resolution five times. Fig. 5-1 shows part of an original metal transfer image and Fig. 5-2 shows the resultant image after the five-time bilinear interpolation. Fig. 5-3 and Fig. 5-4 are the edges detected from Fig. 5-1 and Fig. 5-2 respectively. It is apparent that much more detailed information about the edge is present in the higher resolution images.

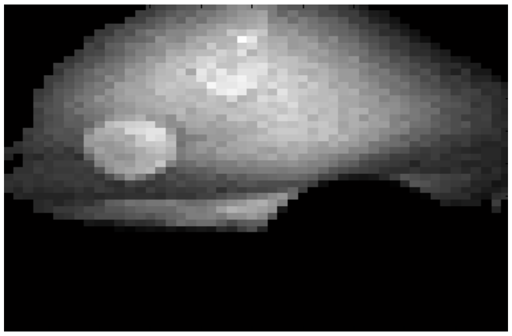


Fig. 5-1 Original image

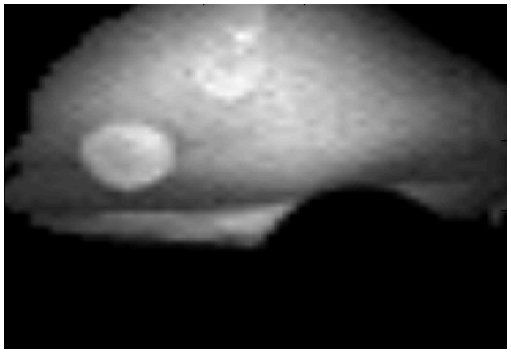


Fig. 5-2 Image after bilinear interpolation

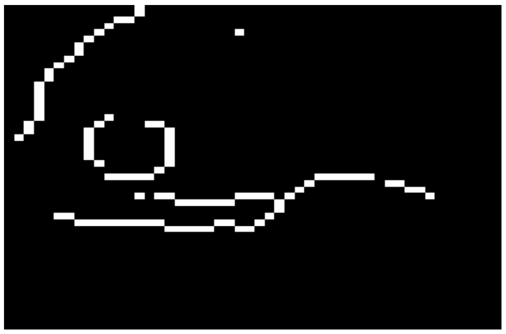


Fig. 5-3 Edge detected from the original image in Fig. 5-1

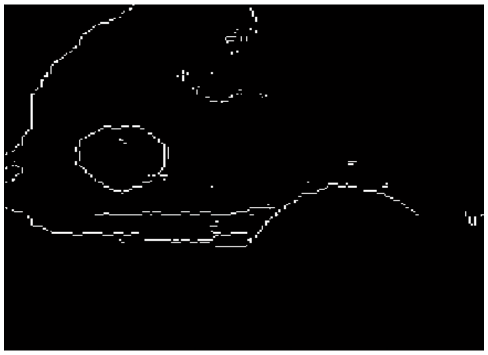


Fig. 5-4 Edge detected from the bilinear interpolated image in Fig. 5-2

Fig. 5-5and Fig. 5-6 can demonstrate the effect of bilinear interpolation further. In Fig. 5-5, (a) is part of the metal transfer image and (b) is the edge detected from it; (c) is the image whose resolution has been increased by using the bilinear interpolation and (d) shows the edge detected from interpolated image. By comparing (b) and (d), it is seen that the droplet's edge has been lost in (b) but (d) provides necessary information to locate the droplet and compute its size. Fig.5-6 also demonstrates the effect of bilinear interpolation in which one droplet is lost in (b) but it is detectable from (d).

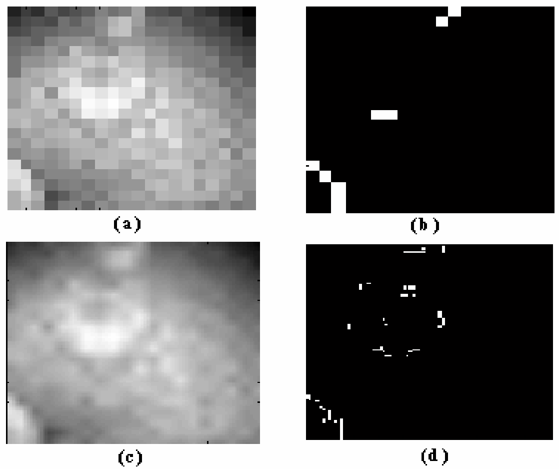


Fig.5-5 Demonstration of the effect of bilinear interpolation: example 1

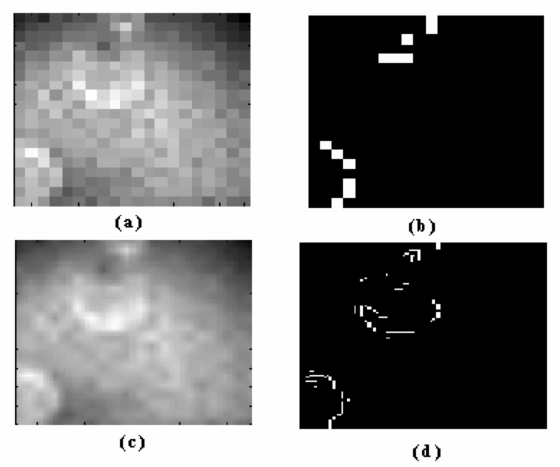


Fig.5-6 Demonstration of the effect of bilinear interpolation: example 2

5.2 Brightness Based Selection and Edge Detection Based Enhancement Separation Algorithm

The proposed brightness based selection and edge based separation algorithm inherits the merit of the previously proposed algorithm and still makes use of the reality that part of the droplet's brightness is higher than other regions. Based on the brightness, the image of the arc area can be divided into several areas, depending on how many droplets the image contains. Only the areas that contain a droplet are processed in order to reduce the processing time. These specific areas are defined as ROIs (regions of interest) in this paper. However, to overcome the difficulty caused by the fact that the droplet is not uniformly brighter than other regions, the proposed algorithm will make use of the droplet edge. It can be described as ten steps.

Step1: Preprocessing.

Step 2: Identification of the wire tip.

Step 3: Binarization for droplet identification.

Step 4: Image filtering.

Step 1, Step 2 Step 3 and Step 4 are completely the same as the first four steps of the Algorithm proposed in Chapter 3.

Step 5: Selection of ROI based on the result in Step 4.

Once the binarized image is filtered, the areas binarized as 1 represent where the droplets are located. For example, if there are two droplets in the metal transfer image, then there should be two main areas with value 1 after binarization. Each ROI is defined based on the center of each area with value 1. The following equation is used to compute the center of these areas.

$$\begin{cases} x_c = \dfrac{x_{\max} + x_{\min}}{2} \\ y_c = \dfrac{y_{\max} + y_{\min}}{2} \end{cases} \tag{5-3}$$

After the centers are found, the ROIs are defined as 20×20 squares around corresponding centers, where the window 20×20 is selected based on the observation of typical diameter of droplet which is not greater than 10 pixels.

Step 6: Bilinear interpolation of the selected ROI.

After all the ROIs are defined, each of them is interpolated by Eq.5-1 for a five-time increase of resolution.

Step 7: Edge detection of interpolated high resolution ROI.

The following four SOBEL operators are used to find the edges:

$$S_x = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix} \quad S_y = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix} \quad S_d = \begin{pmatrix} -2 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 2 \end{pmatrix} \quad S_{id} = \begin{pmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{pmatrix}$$

Where S_x is used to find the edge in x direction, S_y in y direction, S_d in diagonal direction and S_{id} in inverse diagonal direction.

Fig.5-4, Fig. 5-5 (d) and Fig.5-6 (d) illustrate the results of SOBEL edge detection from corresponding bilinear-interpolated images.

Step 8: K-means clustering

Since all edges in the ROI will be detected, this ROI contains some edges that may not belong to the droplet and should be eliminated. An example ROI is shown in Fig. 5-7, it is seen that the bottom-left regions contain edges which do not belong to droplet this ROI corresponds to.

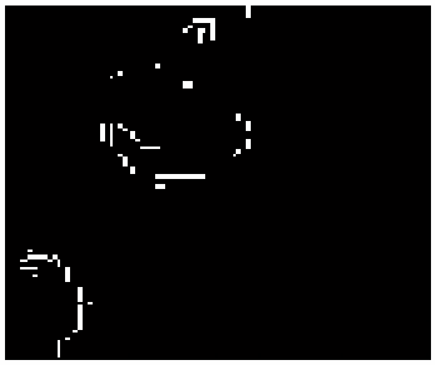


Fig. 5-7 ROI after edge detection

Every droplet's edge is defined as a pattern class [62]–[69] and the prototype of each pattern class is the mean vector of the patterns of that class:

$$m_j = \frac{1}{N_j} \sum_{x \in w_j}^{n} x \qquad j = 1,2 \tag{5-4}$$

where N_j is the number of pattern vectors from class w_j. W is the number of pattern classes. One way to determine the class membership of an unknown pattern vector x is to assign it to the class of its closest prototype. Using the Euclidean distance to determine closeness reduces the problem to computing the distance measures:

$$D_j(x) = \|x - m_j\| \qquad j = 1,2 \tag{5-5}$$

where $\| \ \|$ is the Euclidean norm. x is assigned to class w_i if $D_i(x)$ is the smallest distance. That is, the smallest distance implies the best match in this formulation.

The decision boundary for separating class w_i from w_j is given by values of x for which

$$d_i(x) = d_j(x) \tag{5-6}$$

where $d_i(x)$ and $d_j(x)$ are the decision functions.

Eq. 5-6 is equivalent to evaluate the following decision function $d_i(x)$

$$d_i(x) = x^T m_i - \frac{1}{2} m_i^T m_i \qquad i = 1,2 \tag{5-7}$$

x is assigned to class w_i if $d_i(x)$ yields the largest value.

Then the decision boundary between class w_i and w_j is represented as:

$$\begin{aligned} d_{ij}(x) &= d_i(x) - d_j(x) \\ &= x^T(m_i - m_j) - \frac{1}{2}(m_i - m_j)^T(m_i + m_j) = 0 \end{aligned} \tag{5-8}$$

After the decision boundary is computed, each droplet's edge can be separated based on the boundary. The class with the largest N_j which contains most edge points is maintained. Fig.5-11 shows the result after K-means clustering.

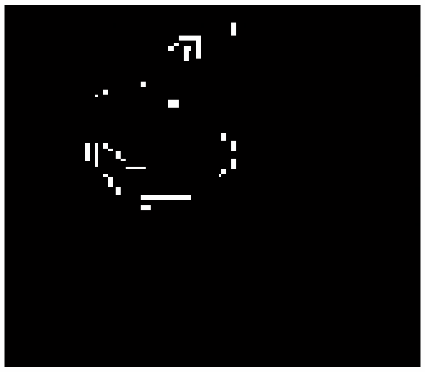


Fig. 5-8 Result of K-means clustering

Step 9: Computing and Validating

The following equation is used to compute the position of the droplet:

$$x_m = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{5-9}$$

$$y_m = \frac{1}{n}\sum_{i=1}^{n} y_i \tag{5-10}$$

The area surrounded by the detected edge points is the size of the droplet. To compute the area of the droplet, three different models are proposed.

The first proposed model is based on the reality that most droplets are like circles, the area of the droplet is computed using the following equation.

$$S = \pi r^2 \tag{5-11}$$

where r is the radius of the circular edge detected for the droplet and it is computed as follows

$$r = \frac{1}{n}\sum_{i=1}^{n}\sqrt{\left(x_i - x_m\right)^2 + \left(y_i - y_m\right)^2} \tag{5-12}$$

where (x_i, y_i) is the point on the edge of the droplet. (x_m, y_m) is the mean value of all the points on the detected edge.

The second proposed model is to fit an ellipse[70][71] to the detected edge using the following equation.

$$\frac{(x-x_m)^2}{a^2}+\frac{(y-y_m)^2}{b^2}=1 \tag{5-13}$$

Where a and b are the short and long radius of the fitted ellipse which can be estimated by the Least Squares algorithm using the detected edge points (x_i, y_i) $(i=1,\ldots,N)$.

$$\theta=(\varphi^T\varphi)^{-1}\varphi^T X \tag{5-14}$$

$$X=\left[1,1\cdots,1\right]^T \tag{5-15}$$

$$\varphi=\begin{pmatrix} (x_1-x_m)^2 & (y_1-y_m)^2 \\ (x_2-x_m)^2 & (y_2-y_m)^2 \\ \vdots & \vdots \\ (x_N-x_m)^2 & (y_N-y_m)^2 \end{pmatrix} \tag{5-16}$$

$$\theta=\left[a^2 \quad b^2\right]^T \tag{5-17}$$

After θ is computed, the size of the droplet can be calculated as the following area:

$$S=\pi ab \tag{5-18}$$

Since the assumed center of the ellipse may not be (x_m, y_m), this ellipse fitting model is improved by moving the center around a 3×3 window and fit nine times and find the one with the minimum Least Square. Figs.5-9-5-14 demonstrate the effectiveness of the minimum Least Square fitting ellipse model. From these figures, it is seen that the fitted ellipse with minimum Least Square Error (MLSE) is more accurate.

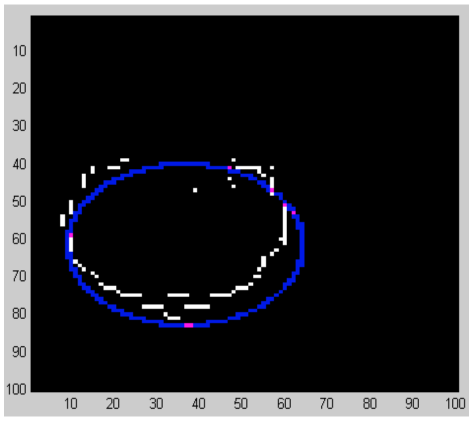


Fig.5-9 Fit an ellipse based on the center (x_m, y_m)

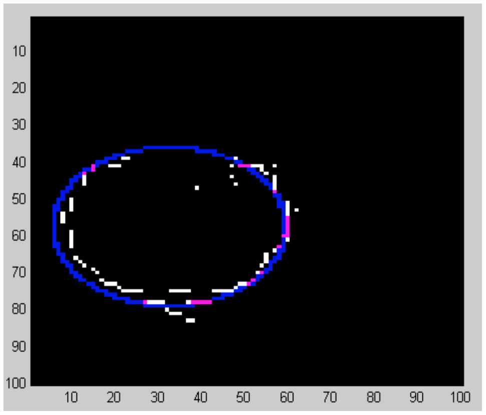


Fig.5-10 Fit an ellipse with MLSE for the same droplet edge shown in Fig.5-9

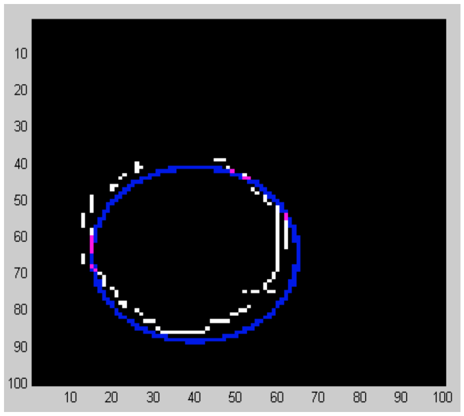


Fig.5-11 Fit an ellipse based on the center (x_m, y_m)

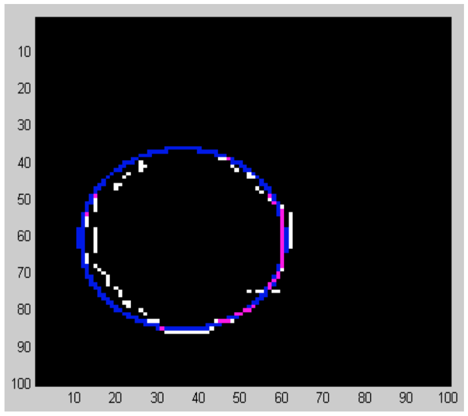


Fig.5-12 Fit an ellipse with MLSE for the same droplet edge shown in Fig.5-11

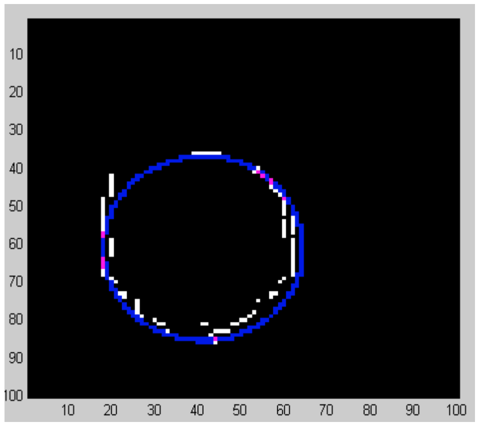


Fig.5-13 Fit an ellipse based on the center (x_m, y_m)

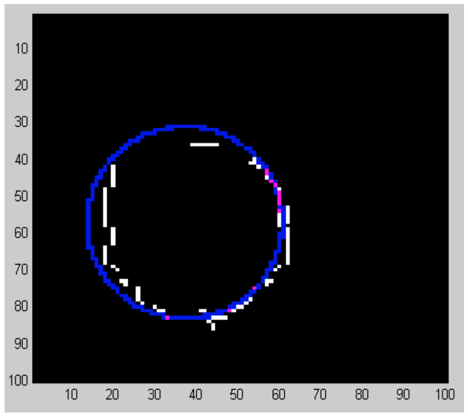


Fig.5-14 Fit an ellipse with MLSE for the same droplet edge shown in Fig.5-13

Since there are discontinuities in the detected edge points, to robustly estimate the droplet size, the third second order curve fitting[72]-[76] model is proposed:

$$r = a_0 + a_1\theta + a_2\theta^2 \tag{5-19}$$

where r and θ are the amplitude and angle of the vector which points to a point on the droplet edge from the computed center, and a_0, a_1 and a_2 are the coefficients which can be estimated by the Least Square algorithm using the detected edge points (x_i, y_i) $(i = 1,..., N)$ or the corresponding (r_i, θ_i) $(i = 1,..., N)$ as

$$\phi = (\psi^T \psi)^{-1} \psi^T Y \tag{5-20}$$

where

$$Y = \left[r_1, r_2 \cdots, r_N\right]^T \tag{5-21}$$

$$\psi = \begin{bmatrix} 1 & \theta_1 & \theta_1^{\,2} \\ 1 & \theta_2 & \theta_2^{\,2} \\ \vdots & & \vdots \\ 1 & \theta_N & \theta_N^{\,2} \end{bmatrix} \tag{5-22}$$

$$\phi = \left[a_0, \quad a_1, \quad a_2\right]^T \tag{5-23}$$

After ϕ is computed, the size of the droplet can be calculated as the following area:

$$A = \frac{1}{2}\int_{-\pi}^{\pi} r^2 d\theta = \frac{1}{2}\int_{-\pi}^{\pi} \left(a_0 + a_1\theta + a_2\theta^2\right)^2 d\theta \tag{5-24}$$

It should be pointed out that the proposed model for the droplet edge appears to be more appropriate than a circle or ellipse because it simply makes use of the fact that the actually droplet edge is smooth. In fact, the shape of the droplet is determined by a number of forces including the gravity, arc pressure and the surface tension. Because of the surface tension, the droplet maintains to be smooth but the effect of other forces make it impossible for the droplet to be an ideal circle or ellipse in the image. In addition, if an ellipse model is used, the coordinates of its center must be re-estimated in order to avoid large fitting error. However, as can be proved, the estimation of the center coordinates requires non-linear optimization and does not have the analytical solution like the one in (5-20). Although the estimation of the ellipse and the estimation of the center coordinates can be separated as demonstrated in the second proposed model, it becomes more time-consuming. For model (5-19), the estimate can be obtained as an analytic solution and the center used to calculate (r_i, θ_i) $(i = 1,..., N)$ from (x_i, y_i) $(i = 1,..., N)$ should not affect the fitting accuracy. Hence, model (5-19) gives a robust yet computation effective method to estimate the size of the droplet.

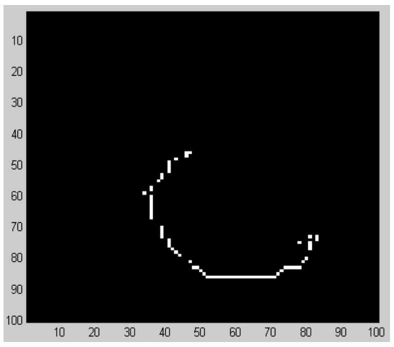


Fig.5-15 An example of the detected edge with a large missing angle range

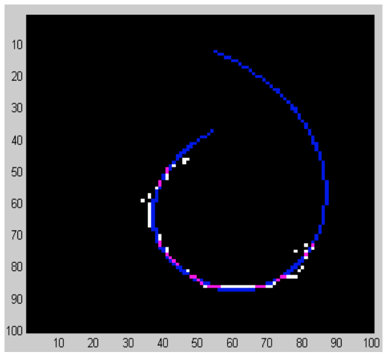


Fig.5-16 Model fitting using data set with a large missing range shown in Fig.5-15

In case a significant range of angle misses in the measurement data set (r_i, θ_i) $(i = 1,...,N)$, a set of coefficients can still be estimated for model (5-19) to fit the given data set well. However, using the model to predict the edge in the missing angle range is to extrapolate and the accuracy can not be assured. Hence, to robustly estimate the size of the droplet which requires computation in the whole angle range $[-\pi, \pi]$ as can be seen in Eq.5-24, a model obtained using a data set with a large missing angle range should not be trusted and used. In this paper, $\pi / 2 = 1.5708$ is taken as the maximally allowed missing angle range. If the missing angle range in the data set is larger than $\pi / 2$, the model would fail the validation. Fig.5-15 shows an example where the missing angle range is large. Fig.5-16 plots the fitted edge. As can be seen, the model can fit the measurement data with high accuracy but its extrapolation appears not acceptable. Hence, the edge data set should be validated or the fitted model needs to be validated to assure that the estimate of the droplet size is robust. Because a droplet would appear in several images and its size should be calculated as an average, the proposed validation would not affect the availability of the computed size but improve the robustness of the estimation.

Step 10: Transform the computed results into their original coordinates

Since the size and the position are computed in the region of interest after bilinear interpolation, they need to be transformed back into their original coordinate system by using following equations:

$$A_0 = A / 25 \tag{5-25}$$

$$x_o = x_c - 10 + x_m / 5 \tag{5-26}$$

$$y_o = y_c - 10 + y_m / 5 \tag{5-27}$$

5.3 Results and Discussion

The proposed Brightness Based Selection and Edge Detection Based Enhancement Separation Algorithm is tested using a series of adjacent frames of metal transfer images. The used series of images are shown in Fig. 5-17. It contains 10 frames which describe the life span of a droplet. The first frame is the image when the droplet is just detached and the following frames record the travel of the detached droplet. In each frame, there are two droplets. The upper one is the new droplet which is just produced and being developed to detach from the wire tip. The lower one is the droplet which has been detached and is traveling.

To demonstrate the effectiveness of the proposed Brightness Based Selection and Edge detection Based Enhancement Separation Algorithm, a few intermediate results of the image processing are demonstrated. Figs 5-18-5-21 show the image processing results for Frame 7 through frame 10. In all these figures, the original image, bilinear-interpolated image, all detected edges, and edges after matching are shown. From these figures, it is seen that the detected edge of the droplet is sufficient to compute the position of the droplet with a high accuracy. Also the detected edge is enough to compute the size of the droplet. Figs 5-22-5-25 plot the modeled edges for the data sets in Figs. 5-18-5-21 using the second order curve fitting model. Comparing Figs 5-22-5-25 with Figs 5-9-5-14, it is seen that the second order curve fitting model can approximate the droplet's contour more accurately. So in theory, to compute size of the droplet using the second order curve fitting model should achieve the most accuracy.

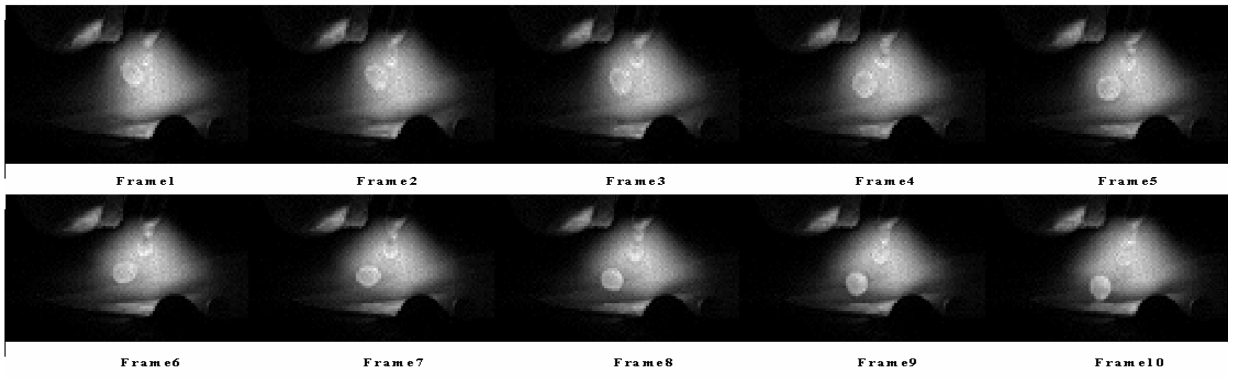


Fig.5-17 Ten adjacent frames

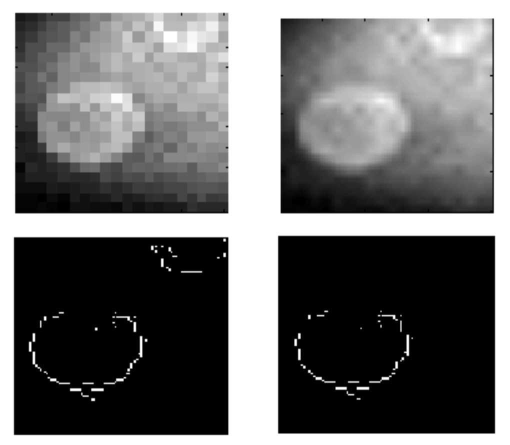


Fig.5-18 Image processing results for frame 7

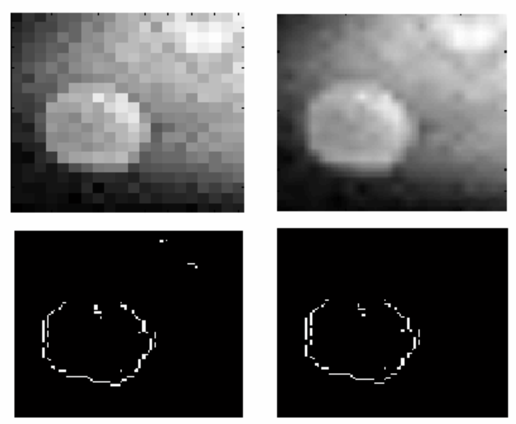


Fig.5-19 Image processing results for frame 8

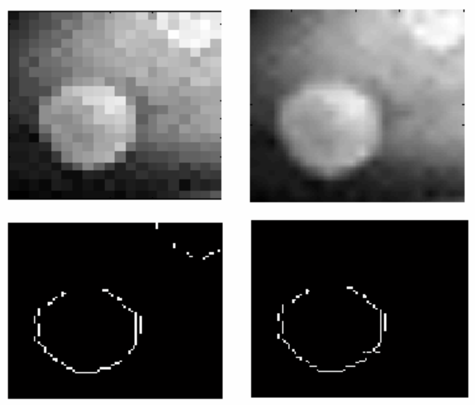


Fig.5-20 Image processing results for frame 9

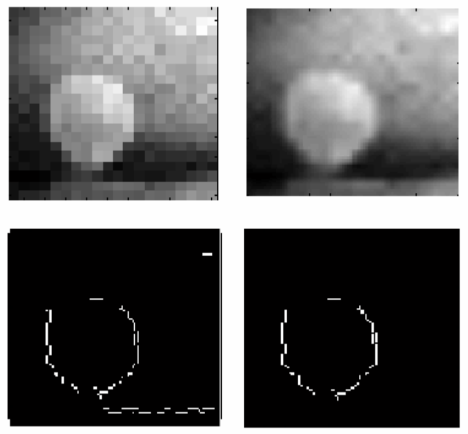


Fig.5-21 Image processing results for frame 10

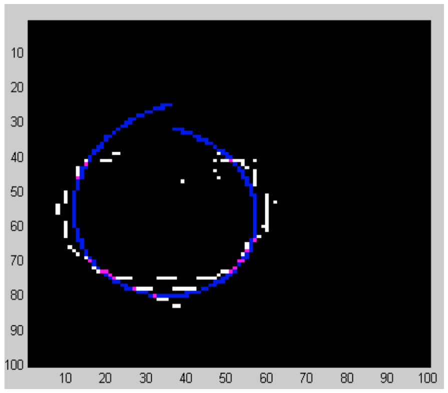


Fig. 5-22 Model fit for detected edge shown in Fig. 5-21

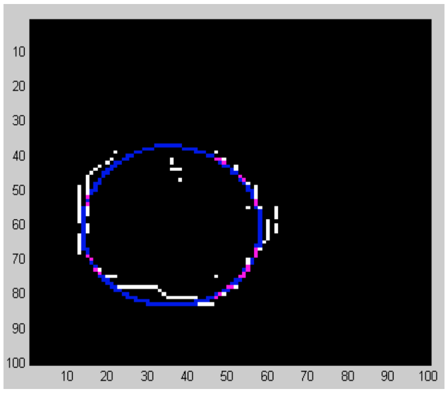


Fig.5-23 Model fit for detected edge shown in Fig.5-22

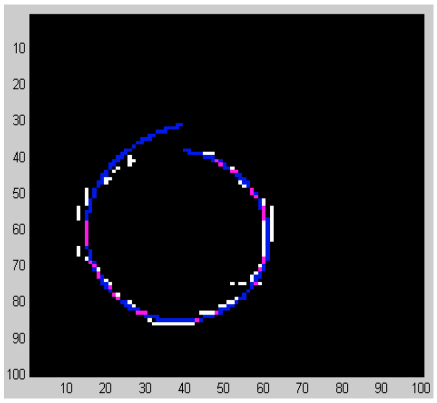


Fig.5-24 Model fit for detected edge shown in Fig.5-23

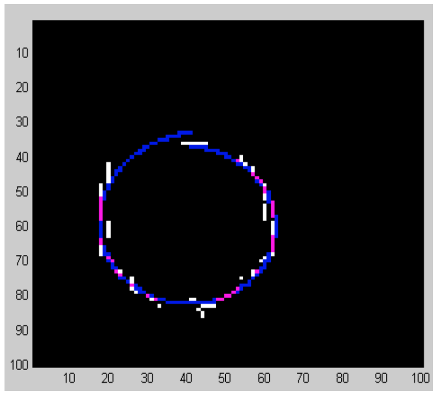


Fig.5-25 Model fit for detected edge shown in Fig.5-24

Table 5-1 shows the observed sizes of the lower droplet in these frames against their computations using the first proposed area computation model. Table 5-2 lists those for the upper droplet.

Tab.5-1. Comparison of the sizes computed using the first area computation model with the observed sizes for the bottom droplets

Frame Index	Droplet' size	Computed size	Error of size
Frame1	81	85	**4**
Frame2	68	61	**-7**
Frame3	76	74	**-2**
Frame4	78	77	**-1**
Frame5	77	49	**-28**
Frame6	67	41	**-26**
Frame7	69	62	**-7**
Frame8	67	63	**-4**
Frame9	66	66	**0**
Frame10	67	64	**-3**

Tab.5-2. Comparison of the sizes computed using the first area computation model with the observed sizes for the top droplets

Frame Index	Droplet' size	Computed size	Error of size
Frame1	31	27	**4**
Frame2	48	46	**-2**
Frame3	46	34	**-12**
Frame4	47	59	**12**
Frame5	50	53	**3**
Frame6	48	47	**-1**
Frame7	40	21	**-19**
Frame8	49	42	**-7**
Frame9	56	31	**-25**
Frame10	70	71	**1**

Table 5-3 shows the observed sizes of the lower droplet in these frames against their computations using the second proposed ellipse fitting area computation model. Table 5-4 lists those for the upper droplet.

Tab.5-3. Comparison of the sizes computed using the second ellipse fitting area computation model with the observed sizes for the bottom droplets

Frame Index	Droplet' size	Computed size	Error of size
Frame1	81	87	**6**
Frame2	68	63	**-5**
Frame3	76	75	**-1**
Frame4	78	77	**-1**
Frame5	77	47	**-30**
Frame6	67	42	**-25**
Frame7	69	64	**-5**
Frame8	67	65	**-2**
Frame9	66	66	**0**
Frame10	67	65	**-2**

Tab.5-4. Comparison of the sizes computed using the second ellipse fitting area computation model with the observed sizes for the top droplets

Frame Index	Droplet' size	Computed size	Error of size
Frame1	31	26	**5**
Frame2	48	46	**-2**
Frame3	46	36	**-10**
Frame4	47	62	**15**
Frame5	50	52	**2**
Frame6	48	47	**-1**
Frame7	40	27	**-13**
Frame8	49	45	**-4**
Frame9	56	37	**-19**
Frame10	70	72	**2**

Table 5-5 shows the observed positions and sizes of the lower droplet in these frames against their computations using the third proposed model. Table 5-6 lists those for the upper droplet. The computed results which pass the validation are used to calculate the averages. As can be seen, the error for the lower droplet size is 0.44 pixel^2 or 0.6%. For the size of the upper droplet, the error is 3.5 pixel^2 or 7.7%. Those accuracies should be sufficient for the control of the droplet size. For the position, the accuracy appears also to be sufficient to estimate the trajectory of the flying droplet (lower droplet). Hence, the proposed algorithm appears to have provided a robust estimate with sufficiency accuracy for possible applications in droplet size and trajectory control.

Tab.5-5. Comparison of computed sizes and positions with observed values for the lower droplet

Frame Index	**Droplet' size**	**Droplet' position**	**Computed size**	**Computed position**	**Size error**	**Position error**	**Max-Missing angle**	**Validity Passed?**
Frame1	81	(135 , 213)	93	(136 , 213)	**-12**	**(1 , 0)**	**1.5644**	**Yes**
Frame2	68	(137 , 213)	66	(135 , 211)	**2**	**(-2 , -2)**	**0.9226**	**Yes**
Frame3	76	(138 , 212)	76	(138 , 211)	**0**	**(0 , -1)**	**0.7655**	**Yes**
Frame4	78	(140 , 211)	76	(141 , 210)	**2**	**(1 , -1)**	**0.6084**	**Yes**
Frame5	81	(142 , 210)	97	(144 , 209)	**-16**	**(2 , -1)**	**1.5895**	**No**
Frame6	67	(144 , 210)	57	(145 , 209)	**10**	**(1 , -1)**	**0.8141**	**Yes**
Frame7	72	(145 , 209)	72	(145 , 209)	**0**	**(0 , 0)**	**0.5672**	**Yes**
Frame8	64	(146 , 208)	64	(146 , 208)	**0**	**(0 , 0)**	**0.5407**	**Yes**
Frame9	73	(148 , 208)	72	(148 , 208)	**1**	**(0 , 0)**	**0.5118**	**Yes**
Frame10	67	(149 , 207)	66	(149 , 207)	**1**	**(0 , 0)**	**0.7489**	**Yes**
Average	71.8		71.333		**0.44**			

Tab.5-6. Comparison of computed sizes and positions with observed values for the upper droplet

Frame Index	**Droplet' size**	**Droplet' position**	**Computed size**	**Computed position**	**Size error**	**Position error**	**Max-Missing angle**	**Validity Passed?**
Frame1	31	(132 , 219)	26	(134 , 217)	**5**	**(2 , -2)**	**1.1092**	**Yes**
Frame2	48	(132 , 218)	52	(135 , 218)	**-4**	**(3 , 0)**	**1.6715**	**No**
Frame3	46	(132 , 219)	35	(135 , 216)	**11**	**(3 , -3)**	**1.8812**	**No**
Frame4	47	(133 , 219)	59	(135 , 217)	**-12**	**(2 , -2)**	**1.0775**	**Yes**
Frame5	33	(134 , 219)	34	(134 , 219)	**-1**	**(0 , 0)**	**1.3189**	**Yes**
Frame6	48	(135 , 218)	49	(131 , 219)	**-1**	**(-4 , 1)**	**0.9813**	**Yes**
Frame7	34	(135 , 218)	23	(135 , 217)	**11**	**(0 , -1)**	**0.7556**	**Yes**
Frame8	49	(135 , 218)	38	(135 , 218)	**11**	**(0 , 0)**	**0.9450**	**Yes**
Frame9	51	(136 , 218)	33	(137 , 216)	**18**	**(1 , -2)**	**1.1508**	**Yes**
Frame10	70	(135 , 218)	73	(134 , 217)	**-3**	**(-1, -1)**	**1.4146**	**Yes**
Average	45.38		41.8750		**3.5**			

Testing shows that the Algorithm proposed in this chapter is a little bit faster than the proposed Subtraction Based Separation Algorithm. Since the Subtraction Based Separation Algorithm has the potential to be implemented in real time processing which is evaluated in Chapter 4, the proposed Edge Detection Based Enhancement Algorithm can also meet the requirement. Furthermore, since the droplet is separated based on its edge instead of depending on a pre-specified or computed threshold, it is more robust for automatic processing.

CHAPTER 6

AUTOMATED PROCESSING TOOL FOR THE DROPLET ANALYZATION

Due to the strong arc light limitation of DE-GMAW metal transfer process, the frame rate of the camera must be up to 3000 frames per second or more to get a clear view of the metal transfer process. The used camera is OLYMPUS i-SPEED which supports frame rate up to 33000 frames per second. However, it is difficult to find a frame grabber which supports real time grabbing and processing rate up to 1000 frames per second. As a result, the real time processing system can not be implemented due to the hardware constraint. But it is still desirable that an automated processing tool can analyze the droplet and compute the related parameters off-line. So an automated processing tool programmed in C++ is developed and its analyzing results are evaluated in this chapter.

The automated processing tool processes the DE-GMAW metal transfer video frame by frame till its maximum frame number is reached. Both subtraction based separation algorithm and edge based separation algorithm are integrated into the C++ automated processing tool. Since the edge based separation algorithm is more robust and faster, only its result after processing is shown in this chapter. The following parameters of the droplets are computed by the developed automated processing tool.

1. The position of the droplet

The position of droplet is critical for the computation of the droplet detaching rate. So the requirement for its accuracy is relatively high. The computation results of the first two hundred and ten frames of the DE-GMAW metal transfer video are plotted in Figs 6-1. In Fig. 6-1(a), it is seen that there are seven wide periods which describe seven droplets' horizontal coordinates respectively. These wide periods are suitable to compute the speeds of the droplets (detach rate). There are also some narrow periods which also represent the existence of the droplets, but they are too narrow for the detach rate computation. As a result, a filter is implemented after the computation to eliminate them. As can be seen in Fig.6-2, the horizontal coordinates will decrease and the vertical coordinates will increase with time passing by in each period. That reflects the reality that the droplet's moving direction is left downward and the coordinate original is the leftmost and topmost point.

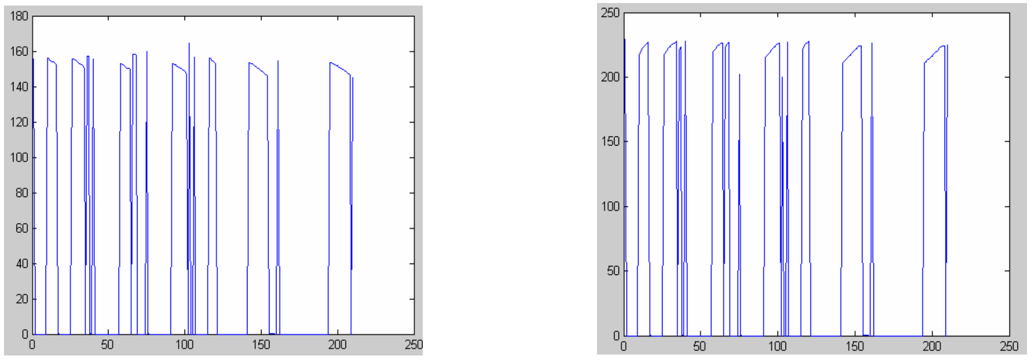

(a) (b)

Fig.6-1 Plot of horizontal (a) and vertical (b) coordinates of the droplets in pixels

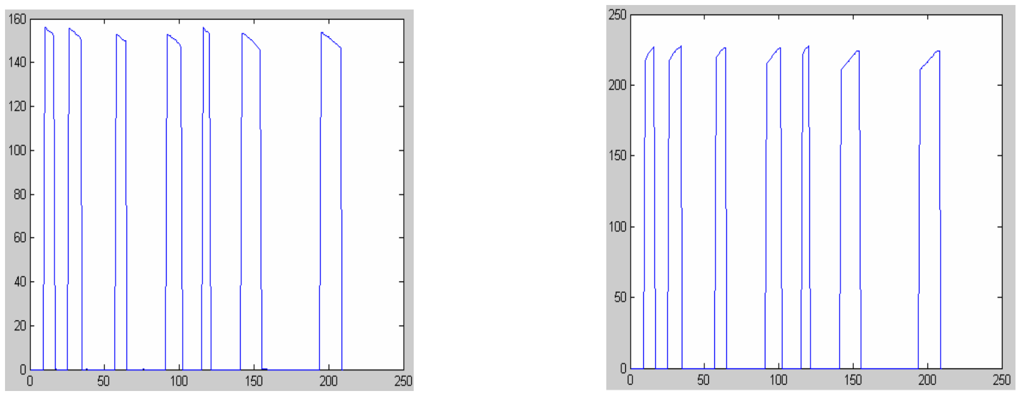

(a) (b)

Fig.6-2 Plot of the horizontal (a) and vertical (b) coordinates of the droplet (after filtering)

2. The detach rate of the droplets

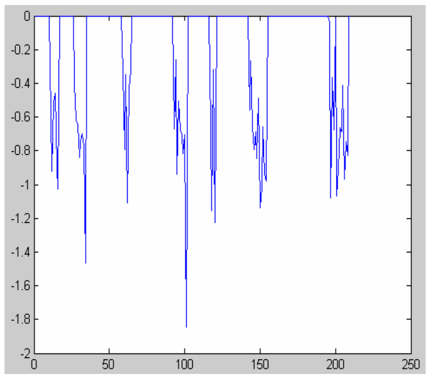
The detach rates of the droplets are computed directly from the computed horizontal and vertical coordinates of the droplets using the following equations:

$$d_h = (h_i - h_{i-1})/t \qquad (6\text{-}1)$$

$$v_h = (v_i - v_{i-1})/t \qquad (6\text{-}2)$$

Where d_h represents the detach rate in horizontal direction while d_v represents the detach rate in vertical direction. h_i is the computed horizontal coordinate in the ith frame and h_{i-1} is the computed horizontal coordinate in the $(i-1)$th frame. v_i is the computed horizontal coordinate in the ith frame and v_{i-1} is the computed horizontal coordinate in the $(i-1)$th frame. t is the time interval between adjacent frames.

Fig.6-3 shows the plot of the computed detach rates of the droplets. As can be seen, the detach rate of one droplet varies during its life period. However the average detach rate will be used for the welding control, this variation will not affect the later control application.


(a)

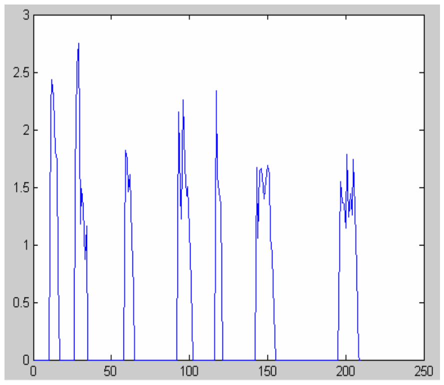

(b)

Fig.6-3 Plot of the detach rate of the droplet in horizontal (a) and vertical (b) direction

3. The angle displacement corresponding to vertical direction

Because of the added by-pass electrode, there is a horizontal electromagnetic force applied to the droplet. To measure the contribution of this force, it is necessary to compute the displaced angle of the droplet corresponding to the vertical direction. Fig.6-4 shows the computed angle displacements of the droplets. It is seen that there is an increase tendency in each period which reflects that the detach rate in horizontal direction increases faster than the detach rate in the vertical direction. Also the average angle displacement will be used as the control feed back.

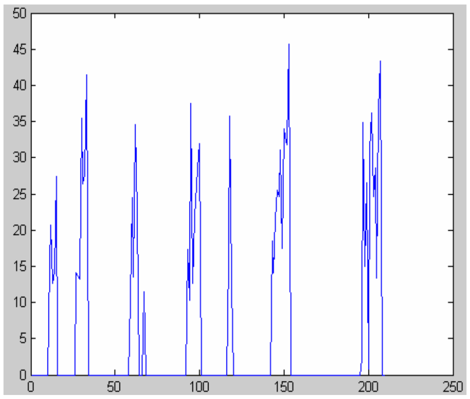


Fig.6-4 Plot of angle displacement corresponding to vertical direction (degree)

4. The areas of the droplets

The area of the droplet is computed by counting the number of pixels it contains. Fig.6-5 shows the plot of the computed areas of seven droplets. It is seen that the area of the same droplet varies in different frames due to the changing shape and position of the droplet. So the average of the areas in each period will be used for the feed back control.

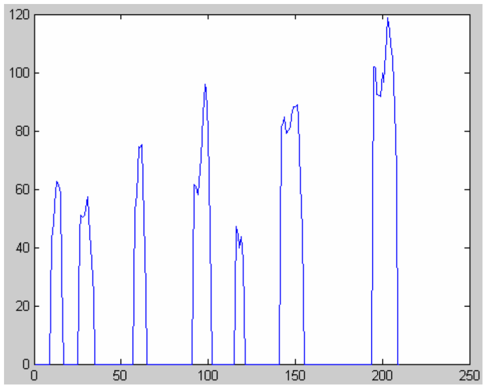


Fig. 6-5 Plot of the sizes of the droplets in pixels

5. The volume of the droplet

Since the droplet is three dimensional, the best feed back information should be the volume of the droplet instead of its area. Although stereo vision can be used to acquire the three dimensional information, it is not available currently due to the hardware constraint. Hence, the droplet is assumed as a sphere computed as follows.

$$V = {4\pi r^3}/{3} \tag{6-3}$$

Where r is the radius of the droplet computed by Eq.5-15. Since the volume of the droplet in the real world coordinate is different from that in the camera coordinate. Hence, Calibration is used to convert the droplet from camera coordinate to real world coordinate. The calibration is based on the pre-knowledge of the width of the wire tip in the real world coordinate. By comparing the wire tip's real width and the number of pixels it contains in the camera coordinate, a universal scalar can be achieved and then applied to the computation of the real world coordinate. Fig.6-6 shows the plot of the computed volumes of the droplets. As can be seen, it is similar to the plot of the areas due to the limitation of the equation adopted.

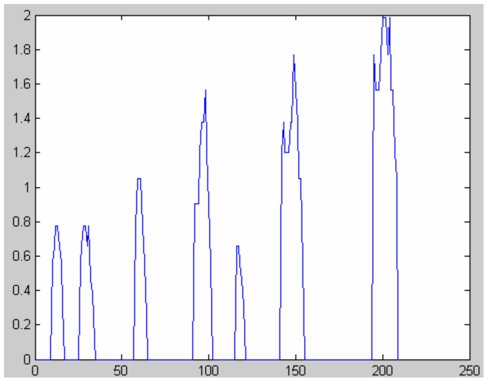


Fig. 6-6 Plot of the volume of the droplet in cubic millimeter

6. The mass of the droplet

After the volume of the droplet is computed, its mass can be computed as follows.

$$M = 7.8 * V \tag{6-4}$$

Where 7.8 is the density of the steel. Fig.6-7 shows the plot of the masses of the droplets.

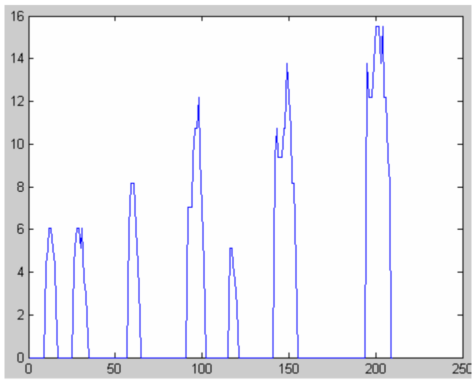


Fig. 6-7 Plot of the mass of the droplet in gram (after filtering)

7. The difference of the horizontal and vertical diameters

It is desirable for the shape of the droplet being monitored to guarantee that the droplet is indeed disk-like. As a result, the difference of the horizontal and vertical diameters is computed correspondingly. It can be determined based on this difference if the treatment of the droplet as ball or disk is wise or not. If the difference in a specific frame is too large, the computation result of this droplet in this frame will be ignored. Fig.6-8 shows the computed differences of the droplets. As can be seen, the average maximum difference is below 2 pixels, which is acceptable to treat the droplet as a disk or ball.

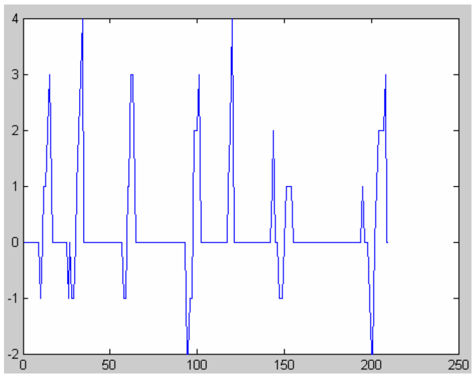


Fig. 6-8 Plot of the difference of the x diameter and y diameter in pixels

8. The reliability (MMA) of the computation (radian).

Because the detected edge of the droplet might not be continuous and if the discontinuity of the edge is too long, the reliability of the computation becomes very low. As a result, the maximum missing angle (MMA) is computed to judge if the computation is reliable. In this research work, $\pi/2$ is chosen as the threshold, i.e. if the maximum missing angle is less than $\pi/2$, the computation is treated reliable. Fig.6-9 shows the plot of the maximum missing angles of the droplets. It is seen that the maximal MMA is less than 0.7, so the computation result is reliable.

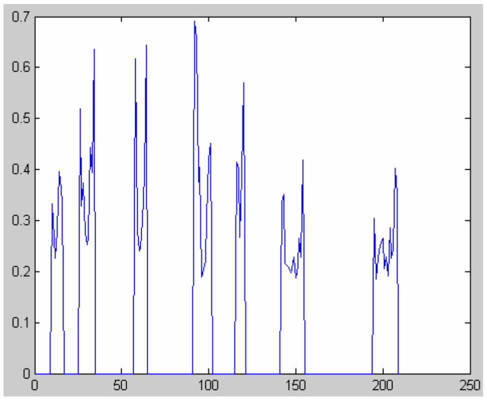


Fig.6-9 Plot of the reliability (maximum missing angle) of the computation in radian

Since the automated processing tool is used to process the metal transfer video off line, the requirement for the processing speed is not very strict. A metal transfer video with 930 frames is processed using the automated processing tool. The cost time for processing 930 frames is 35 seconds. The processing speed is 26.6 frames per second. This slow processing speed is caused by the complexity of the metal transfer video and the variety of parameters computed.

The subtraction based separation algorithm is also integrated into C++ program. The processing speed is very similar to that of the edge detection based C++ program. The cost time for processing 930 frames is about 36 seconds. The processing speed is 25.8 frames per second.

By comparing the automated processing with the manual analysis, it is apparent that the automated processing is more robust and faster. Although the on-line analysis can not be achieved at present due to the hardware constraint, the automated processing is still helpful for welding researchers to analyze and solve welding problems to improve the welding quality. Once the hardware constraint is overcome, the automated processing tool is ready for on-line analysis and computation to provide real time feed back control.

CHAPTER 7

SIMPLIFIED OPTICAL SYSTEM AND AUTOMATIC IMAGE PROCESSING ALGORITHMS

Although the clear metal transfer images may also be acquired without laser back-lighting by using a relatively high frame rate, 3,000 frames per second or higher, it is difficult to find frame grabbers which can support real time grabbing and processing with a frame rate up to 1,000 frames per second at present. Although the technology will develop to provide faster frame grabbers, there is no question that the dependence of application on frame rate is not desirable. Hence, laser back-lighting or similar techniques which allow free selection of frame rate may still be preferred and the authors thus decide to simplify the laser back-lighting method technique. As mentioned in chapter 1, the laser back-light technique had been widely used for imaging the GMAW metal transfer process. Although a high contrast is critical for manual off-line analysis of metal transfer images and the laser back-lighting technique can assure a high contrast, the complexity and the need for careful set-up of the optical system do reduce its desirability. On the other hand, the development of high speed video cameras has made it possible for metal transfer images to be on-line processed to feedback control the metal transfer process. However, the complex optical system needed by the laser back-lighting technique is difficult to maintain the needed precision during travel and its application to real-time control would be unrealistic. As a result, a much easier laser imaging system is proposed in this research work. Also the consequent image processing algorithms are developed for robust and automatic computation of the characteristic parameters of the metal transfer process.

7.1 **Simplified Laser Back-Lighting Technique and System**

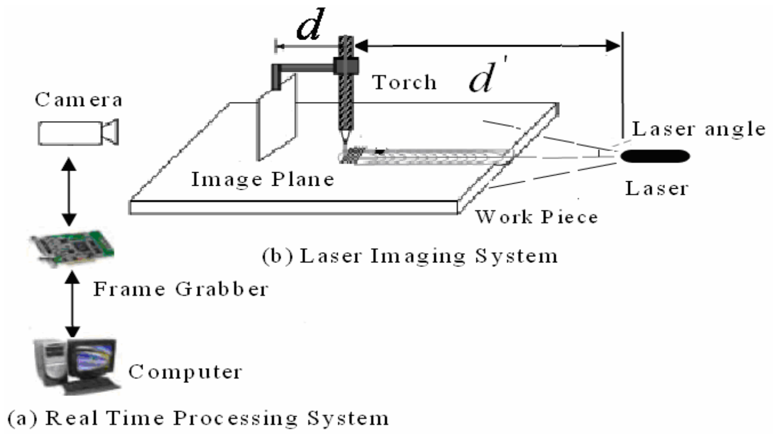


Fig.7-1 Simplified laser back-lighting system

It is suggested that a slightly divergent laser is directly projected without a lens through the metal transfer process. Also, no lens is used between the image plane and torch (Fig.7-1). The sophisticated optical system in Fig.1-2 is thus eliminated and the system is greatly simplified. It is apparent that this method is similar to the laser back-lighting technique except for the elimination of the optical system. To distinguish from the laser back-lighting technique which uses a sophisticated optical system, this method is referred to as the "simplified laser back-lighting technique" in this research work. This less sophisticated technique is illustrated in Fig.7-1.

There are a few parameters which need to be appropriately selected for the simplified laser back-lighting technique. As can be seen in Fig.7-1(b), d is the distance between the image plane and the axis of the torch. It is known that the amount of radiation passing through a specific area is inversely proportional to the square of the distance of that area from the radiation source. This phenomenon is called the inverse square law. It is obvious that the intensity of the radiation diminishes with distances in a non-linear fashion. If d is too large, the intensity of the laser on the image plane would not be sufficient. If d is too small, the intensity of the arc radiation on the image plane would be comparable with or stronger than that of the laser. Hence, the nearest position where the intensity of the arc radiation has been sufficiently reduced should be selected to place the image plane. In our experiments, the distance was chosen as 300 millimeters for the laser beam used (20 mW, 685 nm, 10 degree divergent angle).

Another parameter is d', the distance from the laser projector to the torch. This distance should be chosen based on the divergence of the laser to make sure that the scene (wire and droplet) to be imaged be fully illuminated by the laser beam. In our experiment, this distance was chosen as 250 millimeters for the used laser whose divergent angle is 10 degree.

It is expected that the image quality obtained from the simplified laser back-lighting technique should be reduced from that obtained using the standard laser back-light technique with a sophisticated optical system. Figs. 7-2-7-4 show typical images acquired from a typical spray metal transfer whose peak current is significantly higher than the critical current. In Fig.7-2, a detached droplet is traveling and the tip of the wire is melted but the degree of the tapering is not significant. Fig.7-3 shows a similar case except for the absence of a detached droplet. In Fig.7-4, the tapering is significant. All of these metal transfer images are captured from a pulsed GMAW process whose average current is 176 A powered by an adaptive pulse GMAW machine which automatically set the pulse parameters. From the significance of the tapering observed in Fig.7-4, it can be judged that the peak current used by this adaptive pulse GAMW may have far exceeded the transition current which is considered the minimal current level needed to produce droplets similar to or smaller than that of the wire's diameter.

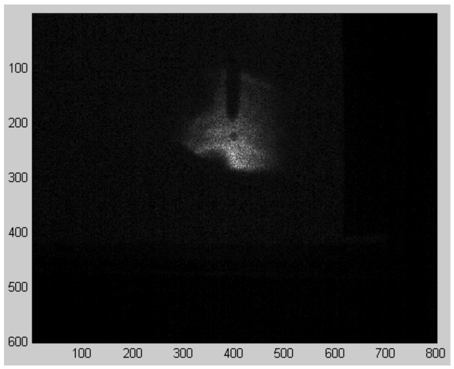


Fig.7-2 Image of insignificant tapering with a traveling droplet

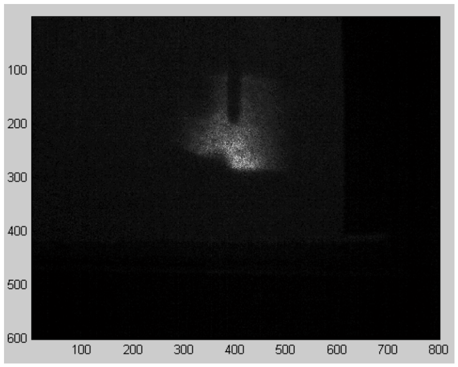


Fig.7-3 Image of insignificant tapering without a traveling droplet

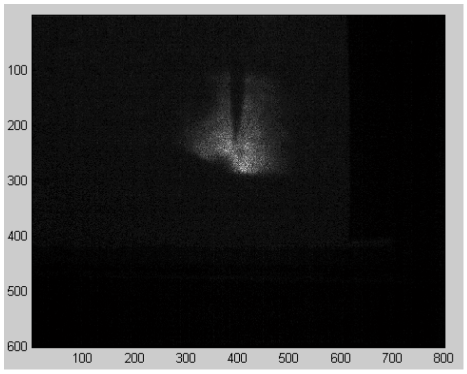


Fig.7-4 Image of significant tapering

In the system used, a high speed camera is connected with a frame grabber that controls the camera to grab images. After grabbing an image, the frame grabber transmits it to the host computer and the computer processes the image in real-time. The camera, frame grabber and computer form the real time processing system shown in Fig.7-1 (a).

Since it is real-time processing, the requirement on the CPU is relatively high. Furthermore, part of RAM needs to be allocated to the frame grabber as the non-paged memory. Hence, the requirement for the RAM is also high. The configuration of the used computer is as follows:

- Windows XP operating system
- Intel® Pentium® D processor 2.99GHz
- 2Gbytes RAM and 120Gbytes hard disk

The used frame grabber is Matrox Meteor-II/Digital which supports a maximum real time grabbing rate of 420 frames per second. The used camera is DALSA CA-D6 which supports capturing rate at 955 and 262 frames per second corresponding to resolution 260×260 and 532×516 respectively.

If a real-time processing is not required and a stand-alone high speed camera is used to record the images, the real-time image processing system shown in Fig.7-1 (a) can be replaced by the camera. The recorded images can then be processed using the image process algorithms to be developed in this research work.

7.2 Analysis of Factors Affecting the Pixel Intensity in the Obtained Metal Transfer Images

The pixel intensity depends on surface reflectivity parameters, surface orientation, type and position of illuminants, and the position of the viewer. Hence, it is important for the study of the factors that affect the pixel intensity of the obtained image during welding, which will for sure sub-serve the development of image processing algorithms.

It is well known that any point source which spreads its influence equally in all directions without a limit to its range will obey the inverse square law. This comes from strictly geometrical considerations. The intensity of the influence at any given radius r is the source strength divided by the area of the sphere. Being strictly geometric in its origin, the inverse square law [77] applies to diverse phenomena. Point sources of gravitational force, electric field, light, sound or radiation obey the inverse square law. Fig. 7-5 shows the demonstration of the inverse square law.

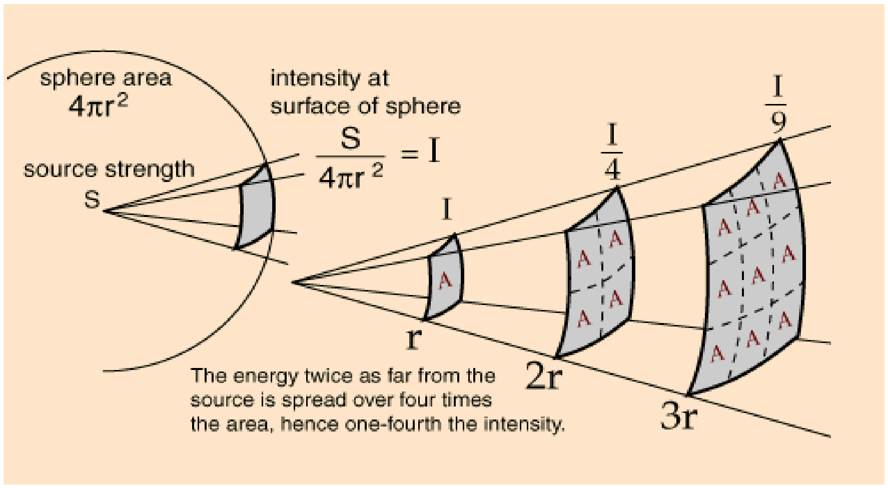


Fig.7-5 Demonstration of Inverse Square Law
(This figure is adapted from internet [77])

For Laser imaging system, the used imaging screen is a white paper. It is assumed that radiation is reflected in all angles equally or near-equally. As a result, the reflectance should be Lambertian reflectance. In computer graphics, Lambertian reflection [78] is often used as a model for diffuse reflection, and is calculated by taking the dot product of the surface's normalized normal vector *N* and a normalized vector *L* pointing from the surface to the light source. This number is then multiplied by the color of the surface and the intensity of the light hitting the surface:

$$I_D = |N| \times |L| \times \cos\alpha \times C \times I_L \tag{7-1}$$

Where I_D is the intensity of the diffusely reflected light (surface brightness), C is the color and I_L is the intensity of the incoming light. α is the angle between the direction of the two vectors, the intensity will be the highest if the normal vector points in the same direction as the light vector ($\cos(0) = 1$).

Based on the inverse square law, the intensity of the incoming light can be computed as follows.

$$I_L = \frac{I_0}{r^2} \tag{7-2}$$

Where I_0 is the intensity of the light at its source and r is the distance between the point on the imaging screen and the center of the light source.

The intensity I_0 of the arc light at its source can be described by planck's law[79] :

$$I_0 = \frac{2hv^3}{c^2} \times \frac{1}{e^{hv/kT} - 1} \tag{7-3}$$

Where v is the spectral frequency in Hz, T is the electron temperature of the source, h is the planck's constant, c is the speed of the light and k is the Boltzmann constant. Since the electron temperature will vary with the current during the welding processing, the intensity of the arc light varies from frame to frame.

7.3 Image Enhancement and Analysis

7.3.1 Image enhancement

It is noticed that the noise in the images acquired by the simplified system is uniformly distributed. A moving averaging filter[80][81] thus can be used to effectively enhance the images. The authors choose the window size of the filter as 5 and apply it line by line first and then column by column. The equations of the moving averaging filter are given below:

$$
\begin{aligned}
&\overline{x}(i) = \big(x(i-1) + x(i-2) + x(i) + x(i+1) + x(i+2)\big)/5 \\
&\qquad i = 3,\ldots M-2 \\
&\overline{x}(2) = (x(1) + x(2) + x(3))/3 \\
&\overline{x}(1) = x(1) \\
&\overline{x}(M-1) = (x(M) + x(M-1) + x(M-2))/3 \\
&\overline{x}(M) = x(M)
\end{aligned}
\tag{7-4}
$$

where $\overline{x}$ is the grayness value of the pixels after filtering in one line or column. Figs. 7-6-7-8 show resultant images after filtering. It is apparent that the filtered images are sufficiently clear and can be used for off-line analysis. Hence, the simplified laser back-lighting technique suggested by the authors appears acceptable for being used in monitoring of metal transfer process.

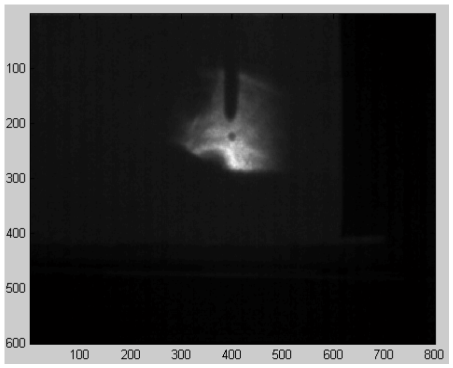


Fig.7-6 Enhancement result for the image in Fig.7-2

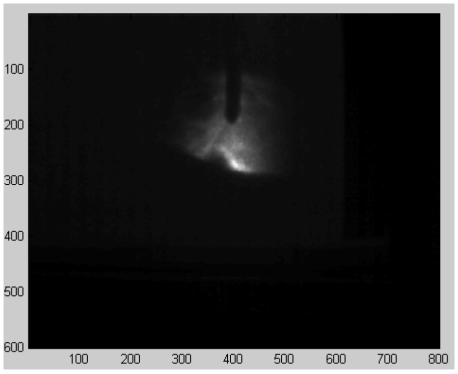


Fig.7-7 Enhancement result for the image in Fig.7-3

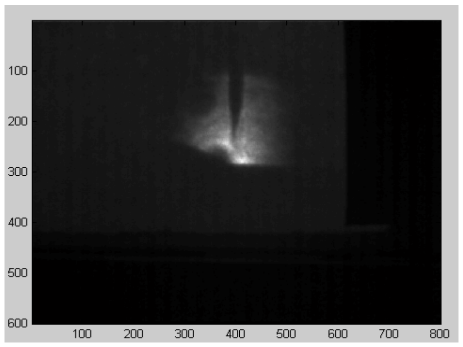


Fig.7-8 Enhancement result for the image in Fig.7-4

7.3.2 Image analysis

By observing the enhanced images, it is seen that the brightness of laser illuminated area appears to vary from frame to frame. This might have been caused by the radiation variation of the arc light. In fact, although the intensity of the arc radiation on the image plane has been reduced to an acceptable level by appropriately choosing the distance d in Fig.7-1, the effect of the arc radiation on the image plane is still not completely eliminated and is thus affected by the welding current. This of course also implies that the intensity of arc radiation on different parts of the imaging screen might be also different because of the different distances between the arc light and different locations on the image plane. Hence, hypotheses are proposed that the arc radiation affects the grayness on the image plane and that the effect of the arc radiation on the image plane varies with the location.

To confirm the first hypothesis, a specific ROI (region of interest) which must cover the arc light radiation area but not illuminated by the laser is cut, and its minimum grayness is computed. Fig.7-9 shows the minimum grayness of the selected ROI in 15 adjacent frames. It is apparent that the minimum grayness varies from frame to frame. As a result, the effect of the arc radiation on the image plane and its variation are confirmed.

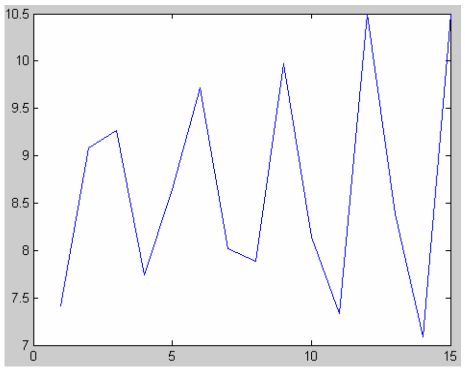


Fig.7-9 Minimum grayness in an arc radiation affected region in 15 adjacent frames

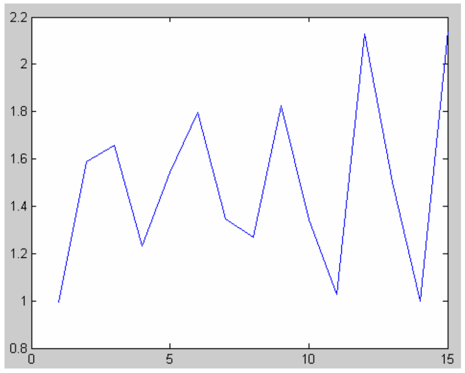


Fig.7-10 Difference of minimum grayness in two arc radiation affected regions in 15 adjacent frames

To verify the second hypothesis that the effect of the arc radiation varies with the location on the image plane, two ROIs are selected: one from the top and the other from the bottom part of the image. Both ROIs are subject to the arc radiation but are not illuminated by the laser. Then the average grayness is computed for these two ROIs to find their difference. Fig.7-10 shows the difference in 15 adjacent frames. Although the difference varies from frame to frame, it is always greater than 1. Further, the variation of the difference is understandable because the intensity of the arc radiation on the image plane varies from frame to frame due to the variation of the welding current. In fact, the pattern of the variation of the difference shown in Fig.7-10 is almost identical to that in Fig.7-9. Hence, the effect of the arc radiation does vary with the location on the image plane.

Another important phenomenon which needs to be noticed is that the grayness of the pixels at the wire tip or droplet's edge appears to be brighter than the center part of the wire tip or droplet but much darker than the area directly illuminated by the laser. This phenomenon might have been caused by the arc plasma which could have caused the laser to refract. When the laser refracts, the sharpness of the shadow will be affected and the gradient of the grayness at the edge of the droplet and wire would be reduced. In addition, the arc plasma in the pulsed GMAW is dynamic and changing, its effect on the laser refraction (thus on the gradient of the grayness at the edge) and the intensity of the laser on the image plane would vary. Hence, the third hypothesis is proposed that the arc plasma affects the propagation of the backing laser.

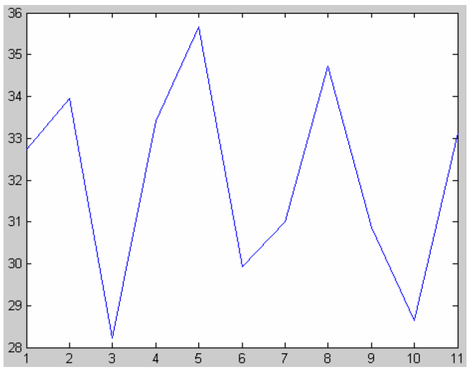


Fig.7-11 Average grayness in a laser illuminated region

To test the third hypothesis, first a ROI in the laser illuminated area is chosen. Then its average grayness in 15 adjacent frames is plotted in Fig.7-11. In comparison with that in Fig.7-9, it is apparent that the pattern of the variation in Fig.7-11 is significantly different. Hence, additional source/sources other than the arc radiation must exist and be responsible for the variation in Fig.7-11. However, laser and arc radiation are the only two sources which produce measurable effect on the grayness on the image plane. Because the arc radiation has been excluded and the laser output intensity is constant, the only cause which may be responsible for this variation appears to be the arc plasma which affects the effect of the laser on the image plane through refracting the laser. Because the effect is realized through refraction and the refraction is not determined by the intensity of the arc radiation or the welding current, the variation of the intensity of laser illumination thus may not necessarily synchronize with the intensity of the arc. Hence, it is possible that the intensity of the laser on the image plane is affected by the arc plasma through laser refraction. Because of the refraction, the travel of the laser lights is not ideal. Hence, the gradient of grayness at the edge is reduced. As a result, the third hypothesis appears to be reasonable. In addition, it appears that the arc plasma affects the laser refraction through the flow and distribution of the gaseous plasma body. Hence, it is also assumed that the laser refraction effect should be Gaussian distributed.

Up to now, the analysis has been focused on factors which strongly/obviously affect the image such as arc radiation and laser refraction. However, there are relatively weak factors which also need to be understood. For example, at the first glance, it appears that the grayness distribution in an area without laser illumination should be ideally uniform, but a careful analysis shows that this is not exactly accurate. As can be seen in Fig.7-12, for the grayness in a relatively small region without laser illumination in the same frame, the grayness does change although the change is relatively small. It appears that although the area under examination is relatively small, the plasma arc still functions as a distributed illumination body and the image on a small region is determined by the integral of its elements in the distribution from a relatively large plasma body. Another example is that the laser illumination intensity is also not uniform as one would expect. Instead, it not only varies but also changes non-linearly.

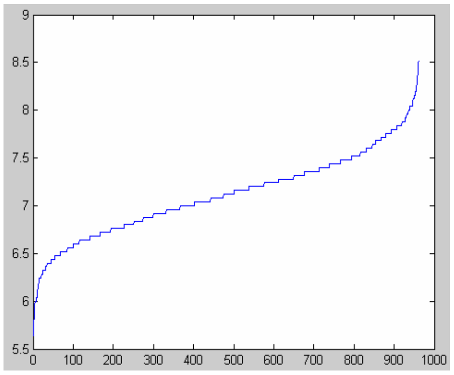


Fig.7-12 Grayness in an arc radiation illumination area

7.4 Evaluation of Automatic Thresholding Methods for the Fuzzy Image

7.4.1 Review of background of image thresholding

In many application of image processing, the gray levels of pixels belonging to the object are substantially different from the gray levels of the pixels belonging to the background. Thresh-holding then becomes a simple but effective tool to separate objects from background. For the captured metal transfer image, it is hard to distinguish the gray levels of the droplet from the background because of the fuzziness and the uncertainty of the pixel. Up to now, many thresh-holding techniques [82]–[89] have been proposed and evaluated. Almost all of them are based on the histogram information of the image. Some of them are suitable for automatic thresh-holding applications, while others are not. In this research work, some of the automatic thresh-holding techniques are evaluated.

7.4.2 Automatic thresholding methods

The automatic thresh-holding methods are categorized into four groups based on the study. 1. Histogram shape based method [84] . 2. Histogram approximation based method [85] . 3. Clustering Based method [86] . 4. Entropy-based method [87]–[89] .

1. Histogram shape based method

This method is based on analyzing the shape of the histogram. In [84], the convex hull of the histogram is calculated and the deepest concavity points become candidates for a threshold. Fig. 7-13 (a) shows a region of interest (ROI) of the metal transfer image and (b) shows its histogram. As can be seen the deepest concavity point occurs at 45, so it becomes a good candidate for thresh-holding. Fig. 7-14 shows the binarized result using the threshold 45.

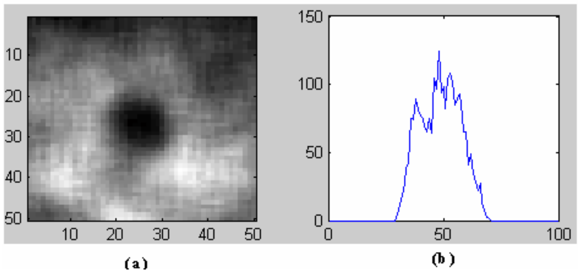


Fig.7-13 ROI and its histogram

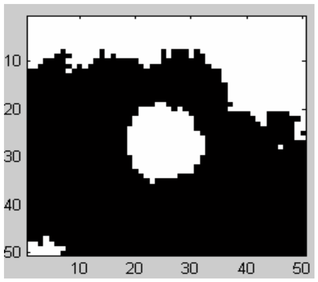


Fig.7-14 Binarized result with threshold=45

2. Histogram approximation based method[85]

This method approximates the histogram of the image with a bi-level function. Minimal sum of square errors or minimal variance is used to decide which point is the optimal threshold. Fig. 7-15 (a) shows the computed minimal sum of square errors from the histogram shown in Fig. 7-13 (b). As can be seen, the local minimal error occurs at point 42, so it is treated as the optimal threshold by this method.

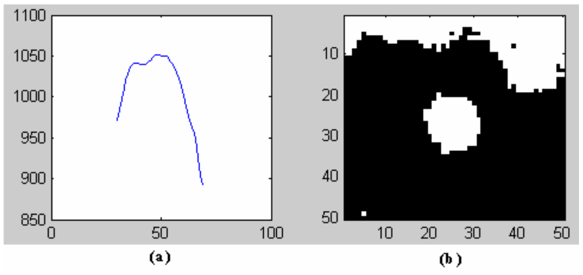


Fig. 7-15 Distribution of MSSE with different thresholds and the binarized result with threshold=42

3. Clustering based method[86]

This method makes use of clustering analysis of the histogram. First, the histogram is divided into two classes, background and foreground. The principle is to find the point which minimizes the weighted within-class variance (WWCV) or maximizing the between-class variance (BCV). Fig. 7-16 (a) shows the computed weighted within-class variance for the histogram shown in Fig. 7-13 (b). As can be seen, the minimal point occurs at 48. Fig. 7-16 (b) shows the binarized result using the threshold 48.

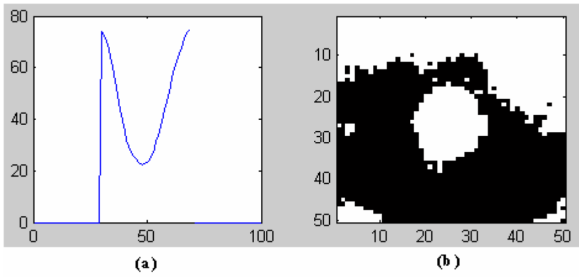


Fig. 7-16 Distribution of WWCV with different thresholds and the binarized result with threshold 48

4. Entropy based method[87]–[89]

This method exploits the entropy of the histogram of the image. The maximization of the entropy of the thresholded image is interpreted as indicative of maximum information transfer. Fig. 7-17 (a) shows the computed entropy distribution with different thresholds and the maximal point occurs at 49. Fig. 7-17 (b) shows the binarized result using the threshold 49.

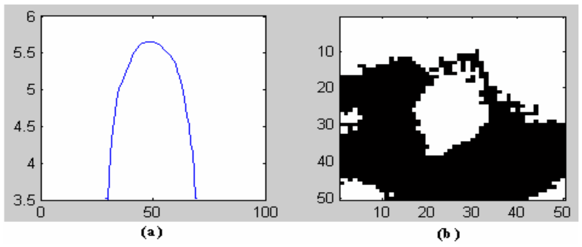


Fig. 7-17 Distribution of entropy with different thresholds and the binarizing result with threshold 49

To improve the accuracy of image thresh-holding, fuzzy set theory has been applied to partition the image space into meaningful regions by minimizing the measure of fuzziness of the image. The key point for fuzzy set is to define a membership function. In [88], the membership function is defined as follows.

$$\begin{aligned}\mu_X(x_{mn}) &= \frac{1}{1+|x_{mn}-\mu_0|/C} \qquad if \qquad x_{mn} \le t \\ &= \frac{1}{1+|x_{mn}-\mu_1|/C} \qquad if \qquad x_{mn} > t\end{aligned} \tag{7-5}$$

Where x_{mn} is a specific pixel at (m, n), C is a constant to make $1/2 \le \mu_X(x_{mn}) \le 1$, μ_0 is the average gray levels of the background and μ_1 is the average gray levels of the foreground. t is a given threshold.

The entropy of a fuzzy image is defined by multiplying the membership function into the entropy function. Fig. 7-18 (a) shows the computed distribution of entropy. As can be seen, the maximal point occurs at 45. Fig. 7-18 (b) shows the binarized result with the threshold 45.

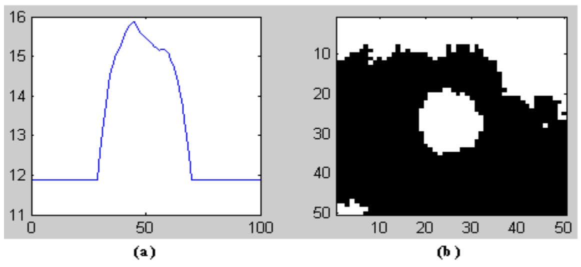


Fig. 7-18 Distribution of entropy and the binarized image with threshold 45

In [89], a different membership function is defined and two-dimensional histogram is used to compute the threshold. Fig. 7-19 (a) shows the computed distribution of entropy. As can be seen, the maximal point occurs at 44. Fig. 7-19 (b) shows the binarized result with the threshold 44.

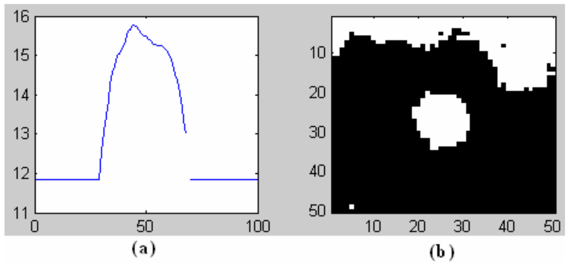


Fig. 7-19 Distribution of entropy and the binarized image with threshold 44

By comparing all of these evaluated thresh-holding algorithms, it is seen that the fuzzy entropy method can achieve the best result. In this research work, the application needs to compute the size of the droplet and the thresh-holding algorithm proposed in Chapter 7 seems more accurate for the computation. Hence, the proposed threshold-holding algorithm is used instead of the fuzzy entropy algorithm.

7.5 Image Processing

The objectives of the image processing are to (1) accurately identify the outlines or contours of the detached droplet and un-detached liquid metal, and (2) compute the characteristic parameters of the metal transfer image from the identified outlines. Two algorithms have been proposed: one for the detached droplet, and one for the un-detached liquid metal. Each algorithm includes two parts: identify the outline and then process the outline to compute the characteristic parameters.

7.5.1 Droplet processing algorithm

Step 1: Region of Interest

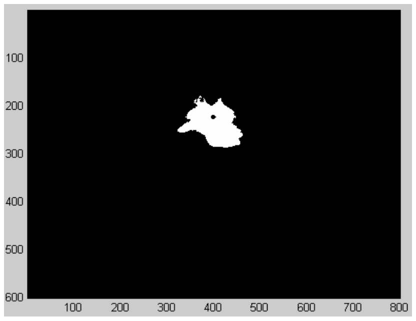


Fig.7-20 Binarized image after enhancement for the image shown in Fig.7-2

A ROI needs to be identified to speed the processing. To this end, the whole metal transfer image is enhanced first using the moving average filter as described earlier. Then a pre-specified threshold is used to binarize the enhanced image in order to obtain the bright area illuminated by the laser. Fig.7-20 shows the binarized result for image shown in Fig.7-2. Using the binarized image, the center, denoted as (x_c, y_c) , of the laser illuminated area is computed as

$$\begin{cases} x_c = \dfrac{x_{\max} + x_{\min}}{2} \\ y_c = \dfrac{y_{\max} + y_{\min}}{2} \end{cases} \tag{7-6}$$

where $x_{\min}$ and $x_{\max}$ are the minimum and maximum values of x coordinate of the bright area, and $y_{\min}$, $y_{\max}$ are the minimum and maximum values of the y coordinate. Once the coordinates of the center of the area are determined, the ROI can be defined. For droplet processing algorithm, the ROI is defined as a 200 pixel by 200 pixel square around (x_c, y_c) .

It is apparent that a different threshold may result in a different laser illuminated area. The center and resultant ROI thus would be different. However, as long as the detached droplet and un-detached liquid metal fall in the ROI, the processing results will not be affected at all. Because the ROI is large enough (200X200), it is guaranteed that detached droplet and un-detached liquid metal fall in the ROI. Hence, the requirement on the threshold for the bright area is low and a pre-specified threshold can be used.

Step 2: Shadow grayness estimate G_s

To detect if there exits a detached droplet and then locate the droplet if exists, the ROI in the enhanced image needs to be binarized using another threshold which can distinguish the droplet, formed as a shadow, from its surrounding area illuminated by the laser. Although the droplet is supposed to be dark, its grayness is not zero because the shadow area it forms at the image plane is also illuminated by the arc radiation and affected by the refracted laser. As proved in Section 7.3, the intensity of the arc radiation on the image plane fluctuates from frame to frame with the current. Use of a constant threshold may thus not be appropriate and the threshold should change from frame to frame.

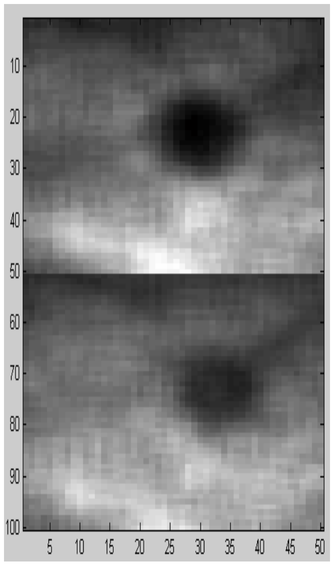


Fig.7-21 Visually comparison of grayness of droplet formed shadow in two adjacent frames

To obtain an adaptive threshold, a region around (x_c, y_c), referred to as the center region which must include at least part of the shadows can be selected to calculate its minimal grayness as an estimate of the shadow grayness G_s. To assure at least part of the shadows be included, the center region is selected as 50 pixels by 50 pixels initially. The minimum grayness in this region in the enhanced image is then calculated as G_s. Based on the study in Section 7.2, G_s should also vary from frame to frame. To demonstrate this, the 50X50 center regions used to calculate G_s in the first two images in a sequence are shown in Fig.7-21. It is apparent that the shadow of the droplet in the first image is darker than that in the second image.

In case the center region is not large enough to contain any part of the shadow, G_s will be similar to the grayness of the laser illuminated area. Hence, G_s is compared with the maximum grayness in the center region. If their difference is not greater than a pre-specified value, the center region is increased from its initial size 50X50 until it becomes greater enough. In this study, the pre-specified value is selected to be 20.

Step 3: Blurry region and shadow probability

Although the center region used to estimate the shadow grayness must include at least part of the shadows, it is not required to include the complete shadows formed by the un-detached liquid metal, low part of the solid wire, and detached droplet. Hence, the outline of the droplet must be more accurately determined. To this end, it is proposed to (1) find an underestimate for the shadow using a relatively low threshold; (2) find an overestimate for the shadow using a relatively high threshold; (3) assign a probability[90] to each pixel in the region defined by the outlines of the overestimate and underestimate of the shadow and then calculate the area of the droplet and droplet center accordingly.

An appropriate underestimate of the shadow should be the inner area where the laser refraction is not significant and an appropriate overestimate of the shadow should be defined by a minimal contour outside which the laser illumination is obvious. It is apparent that the grayness in the laser refraction significantly affected area, i.e., the edge of the droplet, should be closer to that of the shadow than to that of the laser illuminated area. Hence, it is possible that both the higher threshold T_1 and the lower threshold T_2 may be derived from G_s. Based on off-line analysis, $T_2 = G_s + 5$ and $T_1 = G_s + 13$ appear to be usable for the droplet edge if no other more plausible and scientific methods can be found.

After filtering, an inner contour and outer contour can be defined by T_2 and T_1 respectively. The region between the two contours is considered the area where the laser refraction's effect is significant and is referred to as the blurry region in this paper. While any pixel inside the inner contour or outside the outer contour can be considered shadow or not-shadow respectively for sure, it is not absolutely sure if a particular pixel in the blurry region is a shadow or not-shadow point.

The following simple linear equation is proposed to compute the probability of a particular pixel with grayness G_i in the blurry region for being a shadow point:

$$P_i \begin{cases} = (T_1 - G_i)/(T_1 - T_2) & T_1 > G_i > T_2 \\ = 1 & T_2 > G_i \quad i = 1, \ldots, N \\ = 0 & G_i > T_1 \end{cases} \tag{7-7}$$

where N denotes the number of pixels in the blurry region. The probability of any pixel in the underestimated shadow region is set to 1.

Step 4: Area and position computation for the droplet

The area of the droplet can be computed as

$$A_d = \sum_{j=1}^{M} P(O_j) \quad j = 1, \ldots, M \tag{7-8}$$

where $O_j's(j=1,...,M)$ is the set of all pixels/points in the overestimated shadow region defined by the outer contour and $P(O_j)(j=1,...,M)$ is the probability of pixel/point O_j for being a shadow point.

The position of the droplet is computed as follows:

$$x_d = \frac{1}{A_d}\sum_{j=1}^{M} P(O_j) * x_j \tag{7-9}$$

$$y_d = \frac{1}{A_d}\sum_{j=1}^{M} P(O_j) * y_j \tag{7-10}$$

where x_j and y_j are the vertical and horizontal coordinates respectively of point O_j.

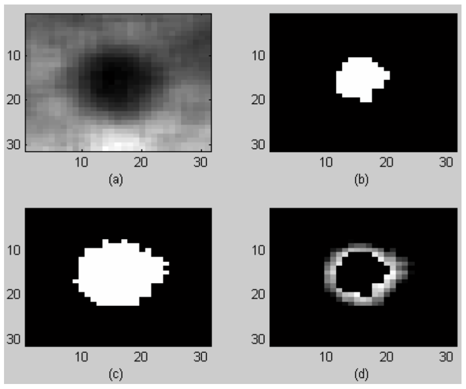


Fig.7-22 Example of droplet processing

Fig.7-22 (a) shows an example of a zoomed-in droplet. (b) and (c) are its underestimate and overestimate determined using T_1 and T_2 respectively, and the blurry region is given in (d). The computed position for this droplet shown is (222.45, 400.03) while the position estimated by observation is (222, 400). The computed area of droplet is 122.3.

Since the droplet location and area are identified and calculated on the image plane coordinate system and the laser diverges, the following equations are used to convert the computed results into the actual scale:

$$x_d' = \frac{d'}{d'+d} \times x_d \tag{7-11}$$

$$y_d' = \frac{d'}{d'+d} \times y_d \qquad (7\text{-}12)$$

$$A_d' = \frac{d'}{d'+d} \times \frac{d'}{d'+d} \times A_d \qquad (7\text{-}13)$$

Although the use of G_s in defining $T_2 = G_s + 5$ and $T_1 = G_s + 13$ appears to be reasonable, it may not be most plausible and scientific to use the two constants (5 and 13), obtained from off-line analysis, in the processing of all images. The resultant estimates could be conservative, i.e., the inner contour is smaller and the outer contour is larger than their possible optimal ones thus resulting in an increased blurry area. Although using probability may still give droplet area and center location very close to the possible optimal ones, it is preferred that a more plausible and scientific method is introduced. As a result, a method is proposed to determine T_1 and T_2 automatically for each frame based on the maximum slope difference principle proposed by the Dr Yuming Zhang in a previous study[91]. It is a method used to obtain a border point, which divides the deformed heat-affected zone (HAZ) and the deformed weld which is formed from the solidified molten metal, from a structured-light which has been noised because of the effect of arc radiation.

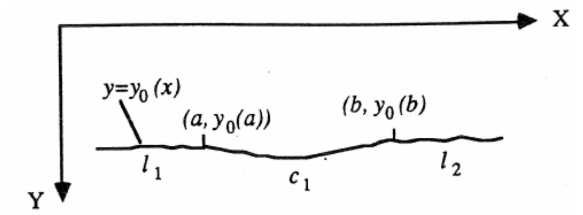


Fig.7-23 A noised structured-light used to determine the two weld border points which divide the weld profile c_1 from unformed HAZ profiles l_1 and l_2 [91]

Fig.7-23 shows an example of a noised structured- light. If it is not noised, it should reflect the accurate shapes of the surface being illuminated and the determination of the border points would be straightforward. However, the noises complicate the determination especially when the deformation of the weld profile is small. Dr Yu-ming Zhang proposed that for each point (x, y_0) on the structured-light, an appropriate number of points on its left and right neighborhoods are used to fit two straight lines respectively. It is apparent that when (x, y_0) is on l_1, some points on l_1 will be used to fit the right straight line so that the resultant slope reduces from its actual value while the slope of the left line is not affected. The difference is thus reduced. Similarly, if (x, y_0) is on c_1, the resultant slope for the fitted left straight line will be increased and their difference will be reduced. Hence, the actual border point $(a, y_0(a))$ should give the expectation of the slope difference a local maximum and the border point can be estimated as the point which gives the maximum slope difference. The detailed theoretical proof and assumptions can be seen in [91].

The maximum slope difference principle may be applied to determined optimal T_1 and T_2. To this end, the authors define a region which contains the center region, the blurry region and laser illuminated region as Region of Droplet (ROD). All points O_j's $(j = 1,\dots, M)$ in the ROD can be re-ordered based on the grayness. If the order index is considered as x coordinate and the grayness is considered as y_0, a $y = y_0(x)$ similarly as in Fig.7-23 would be resulted except for that this time the increase of $y_0(x)$ with x would never be negative because of the re-ordering. It is apparent that when $y_0(x)$ (grayness) increases with x (the order index) from the center region's shadow points, through the blurry area points, to the laser illuminated area points, $y_0(x)$ realizes its most increase at the two boundaries. Hence, it is appears to be plausible to define an optimal T_2^* at the outer boundary and define an optimal T_1^* at the inner boundary respectively. Hence T_1^* and T_2^* are obtained, the overestimate and underestimate of the droplet shadow can be computed to reduce the conservativeness. Eqs.7-7-7-10 can still use T_1^* and T_2^* to replace $T_1 = G_s + 13$ and $T_2 = G_s + 5$ respectively to re-compute the probability, area and center location. Fig.7-16 demonstrates the application of the maximum slope difference method in determining T_1^* and T_2^*.

In Fig.7-24, (a) is the same as in Fig.7-22 (a) which shows a droplet in enhanced image. The bright area in (b) shows the ROD obtained using $T_1 = G_s + 20$. (A larger T_1 is used to increase the number of points with higher grayness in order to calculate the slope of the fitted line on the right of the boundary points as can be seen in Fig.7-24 (b)). The re-ordered grayness distribution is shown in (c). As can be seen from Fig.7-24 (c), the grayness distribution is almost linear and the boundary points are difficult to identify visually. However, there are still variations and the maximum slope difference principle is thus applied to determine the boundary points statistically. To this end, the following model is proposed to find two slopes for point at $(i, g(i))$.

$$g(i) = a_1(i) + a_2(i) * i \tag{7-14}$$

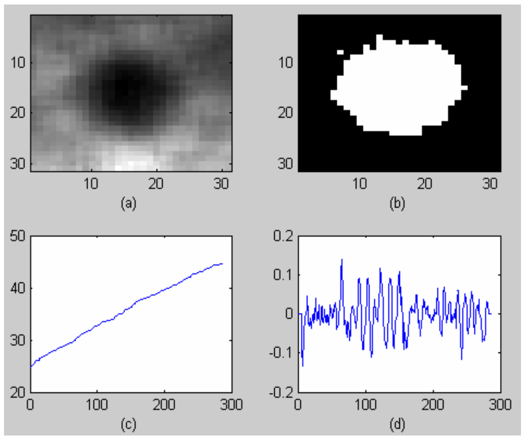


Fig.7-24 ROD and slope difference in grayness distribution

where i and $g(i)$ are the horizontal and vertical coordinate in the grayness distribution in Fig.7-24 (c). $a_1(i)$ and $a_2(i)$ are the coefficients which can be estimated using the Least Squares algorithm with a given specific number of points on its left or on its right. It is straightforward that the slope of fitted line is $a_2(i)$. Five points are chosen on the left and five points on the right respectively to compute its left slope $a_2(i, left)$ and right slope $a_2(i, right)$. Hence, the slope difference at point $(i, g(i))$ is computed as:

$$SlopeDiff(i) = a_2(i, right) - a_2(i, left) \tag{7-15}$$

Fig.7-24 (d) shows the plot of the computed slope differences. It is seen that the global peak occurs at horizontal coordinate 65 which corresponds to grayness value 29.76 in Fig.7-24 (c). This position is expected to be the boundary between the center region and the blurry region and T_2^* is thus chosen as 29.76. There is also a secondary peak which occurs at horizontal coordinate 122 which corresponds to the grayness 33.92 in Fig.7-24 (c). Hence, 33.92 is selected as T_1^* for the boundary between the blurry region and laser illuminated region.

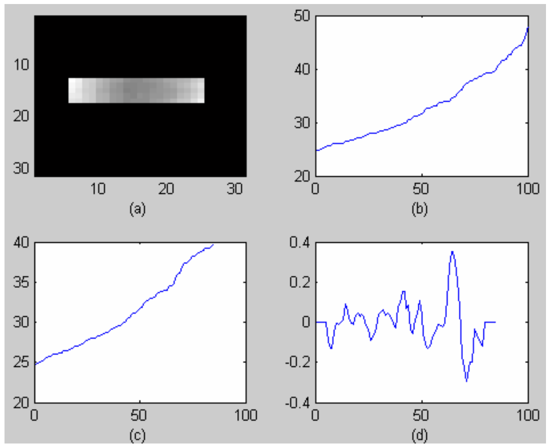


Fig.7-25 Proposed method to preserve the gradient characteristics and its application in droplet blurry region thresholds determination

In Fig.7-24 (c), the distribution appears too gradual and the characteristics of the grayness gradient in the three zones are not observable. The results obtained from such a gradual distribution appear not most trustable. To investigate further, the authors draw a horizontal stripe (5 pixel width and 20 pixel length) and generated the distribution from all pixels from it. Fig.7-25 (a) shows the chosen stripe with corresponding pixels from the droplet area assigned and (b) gives its re-sorted grayness distribution. In (c), all points with grayness greater than 40 are not plotted to exclude apparent laser illuminated points. Now it is apparent that the distribution contains three zones: approximately 1-42 which represents the center part of the droplet, approximately 43-62 which represents the blurry region of the droplet, and approximately 63-84 which represents the laser illuminated region around the droplet. Fig.7-25 (d) shows the plot of the slope differences computed from Fig.7-25 (c). As can be seen, two maximum peaks occur at horizontal coordinate 42 and 62 respectively. The corresponding T_2^* is 29.9 and the corresponding T_1^* is 34.04. Although they are similar with those computed earlier from the grayness distribution of the whole region, the confidence of the decision is improved because the decision is made based on observed characteristics of grayness gradient in the three zones. A careful thinking-through suggests that in the distribution generated from the whole region as shown in Fig.7-24 (c), the gradient change in a two-dimensional world has been re-arranged in one-dimensional world such that the more points in the laser illuminated area are included. As a result, the number of points used in the distribution in Fig.7-24 (c) increases as the distance from the center of the droplet toward the laser illuminated area. Hence, the gradient is reduced from its original visual impact as the distance increases and the confidence level is thus also reduced. By taking a stripe, the one-dimensional characteristics of the gradient change across the three zones are preserved in its grayness distribution shown in Fig.7-25 (c) and the decision confidence and accuracy are improved.

Using T_1^* and T_2^* determined from the maximum slope difference, Step 3 and Step 4 are performed again. Fig.7-26 shows the processing result of the same droplet shown in Fig.7-24. As can be seen, the blurry region in Fig.7-26 is narrower and the estimates are thus less conservative.

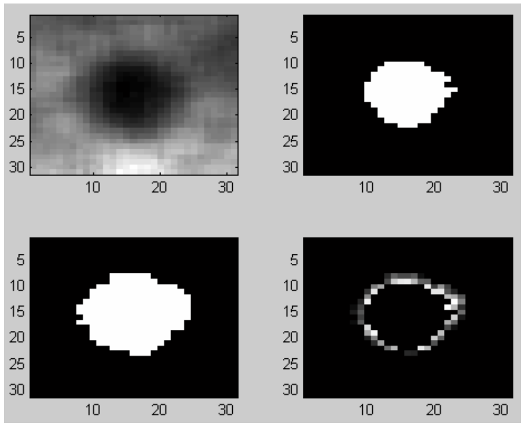


Fig.7-26 Example of droplet processing with the maximum slope difference determined boundaries

7.5.2 Undetached liquid metal processing algorithm

As shown in Figs.7-2-7-4, a rectangular region surrounding the tip of the wire is adequate to compute the length and average width of the undetached liquid metal. A 40X110 rectangular region is chosen, which not only contains all the required information, but also can expedite the processing speed greatly.

Step 1: ROI

Using (x_c, y_c) obtained in Step 1 in the droplet processing algorithm, a 40X110 rectangular region above is defined as the ROI for the identification of the undetached liquid metal. Fig.7-27 (a) shows an example of the chosen ROI.

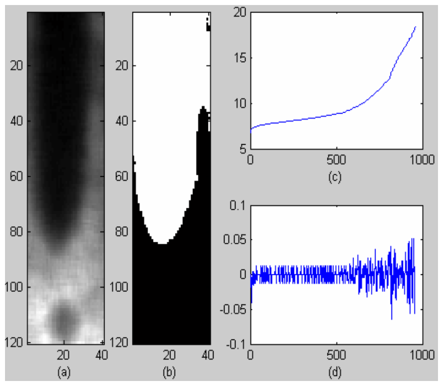


Fig.7-27 ROT and slope difference processing

Step 2: Blurry Area

The bright area in Fig.7-27 (b) shows the region of tapering (ROT) for the un-detached droplet or tapering of the liquid metal obtained by $G_s + 15$ and Fig.7-20 (c) shows the re-ordered grayness distribution. Fig.7-27 (d) shows the plot of the slope differences computed from the sorted grayness in ROT. Unlike the computation for ROD, the authors would use *a priori* knowledge of in what range the two boundaries are located here for ROT. In particular, off-line analysis shows that the underestimate threshold occurs within horizontal range 600-700 and the overestimate occurs within the horizontal range 700-800. Based on this *a priori* knowledge, T_1^* and T_2^* can be obtained based on the peak in range 600-700 and range 700-800 respectively. Fig.7-28 (b) shows the underestimate of the example liquid metal shown in Fig.7-28 (a) using T_2^* and Fig.7-28 (c) shows the overestimate computed using T_1^*. Eq.7-7 is used again to acquire the blurry region. Fig.7-28 (d) shows the computed blurry area.

Again, the use of the whole region for the ROT appears to have reduced the observed gradient characteristics in the inner droplet, blurry, laser illuminated region zones in the grayness distribution shown in Fig.7-27 (c). The visual impacts of the details at the edges have been reduced by adding many points in the laser illuminated areas in the grayness distribution. Hence, the authors draw a horizontal stripe (20 pixel width and 40 pixel length) from the ROT and re-generated a grayness distribution. Fig.7-29 (a) shows the chosen stripe with the corresponding pixels. Fig.7-29 (b) shows the re-sorted grayness distribution. As can be seen, it also contains the same three parts as in the droplet processing. The authors still eliminate the brightest laser illuminated part to exclude apparent laser illuminated points and the resultant distribution is shown in Fig.7-29 (c). Fig.7-29 (d) plots the computed slope differences. The maximum occurs at horizontal coordinate 559 and 612 respectively. The computed lower threshold T_2^* and higher threshold T_1^* are also both very close to those computed based on the slope differences plotted in Fig.7-20 (d). However, the decision is made this time based on the observed characteristics of the three zones without a reference to *a priori* knowledge on the ranges for T_1^* and T_2^*.

Step 3: Computation

First, the end position of the liquid metal is computed using the following equation:

$$endP = (\max(x), y) \ \forall (x, y) \in O \tag{7-16}$$

where O is the overestimate shadow for the un-detached liquid metal and solid wire, and x is the vertical coordinate and y is the horizontal coordinate as in the convention in image processing where a point is given by (row, column). .

Second, the width of the total liquid metal is computed using the following equation:

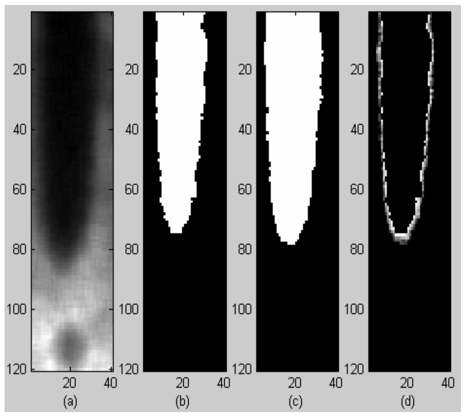


Fig.7-28 Example of processing results using thresholds determined with *a priori* knowledge

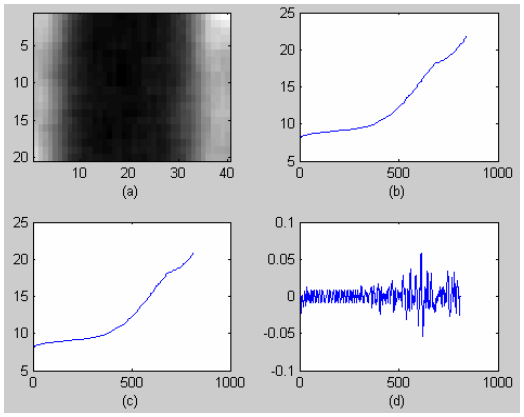


Fig.7-29 Proposed method to preserve the gradient characteristics and its application in un-detached liquid metal blurry region thresholds determination

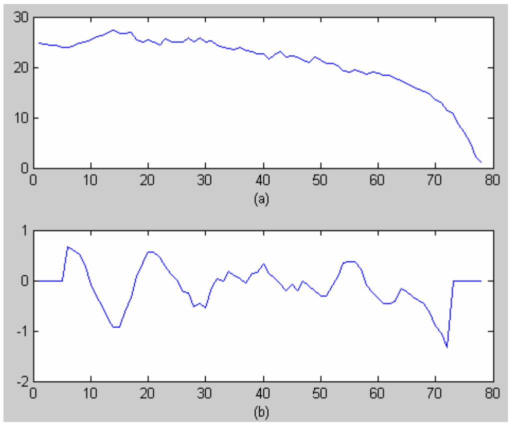


Fig.7-30 Illustration of computing the solid-liquid boundary

$$width(x) = \sum_{y=1}^{y_{\max}} P(x,y) \qquad x = 1,\ldots, x(endP) \tag{7-17}$$

where $P(x,y)$ is the probability of pixel (x,y) and the maximum of the *y* coordinate range for the ROT is 40. Fig.7-30 (a) shows the computed width versus the *x* coordinate for the image shown in Fig.7-28 (a).

Third, the solid-liquid boundary point ($slbP$) is computed based on the maximum slope difference from the width distribution shown in Fig.7-30 (a). Fig.7-30 (b) shows the plot of the computed slope differences in the range from 1 to $endp$. As can be seen, the global maximum occurs at horizontal position 6. Hence, the boundary point occurs at position 6. A careful observation of the image suggests that this estimate is similar as what is predicted.

Fourth, the length of the tapering including the un-detached droplet and the liquid is computed:

$$L = x(endP) - x(inP) \tag{7-18}$$

Finally, the average width of the tapering is computed using the following equation:

$$W_a = \sum_{x=x(inP)}^{x(endP)} width(x) / L \tag{7-19}$$

Step 4: Converting the position and size.

Eqs.7-11, 7-12 and 7-13 are used again to convert the computed results to the metal transfer coordinate system.

7.5.3 Edge based separation algorithm

The poor quality of the acquired metal transfer images plus the moving average filtering makes it impossible to detect the droplet's edge by applying the existing edge detectors. To overcome this problem, a Maximum Slope Difference based edge detecting algorithm is proposed to detect the edge of the droplet.

First, Bilinear interpolation is performed on the region around the droplet (ROD). A five time increase of resolution is chosen to achieve the sub-pixel accuracy. Fig.7-31 (a) shows an example ROD and (b) shows the interpolated result.

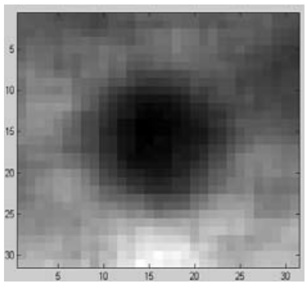
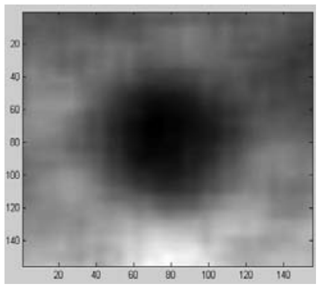



Fig.7-31 (a) An example ROD (b) the interpolated ROD

Second, compute the slope differences in the center toward edge direction of the interpolated droplet using Eq.7-14 and Eq.7-15. Choose the point with the maximal slope difference as the edge point. Fig.7-32 (b) shows an example of the chosen points based on the maximal slope difference. As can be seen, these points are Gaussian distributed around the center which proved the assumption in Sect. 7.3.2.

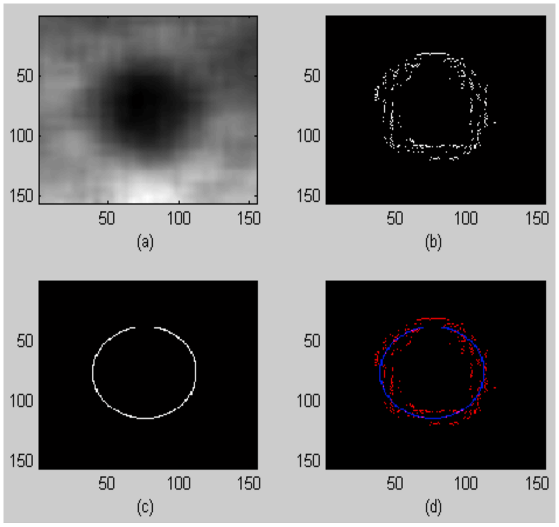


Fig.7-32 Demonstration of MLD based edge detection algorithm

Third, based on the detected edges, the model (5-22) proposed in Chapter 5 is used again to fit the detected edge into a second order curved line. Fig.7-32 (c) shows the fitted line and (d) shows the comparison of the fitted line with the detected edge. Eq.5-27 is used to compute the area of surrounded by the fitted line. The following equations are used to compute the x coordinate and y coordinate respectively.

$$x_{d0} = \frac{1}{M}\sum_{i=1}^{M} x_i \tag{7-20}$$

$$y_{d0} = \frac{1}{M}\sum_{i=1}^{M} y_i \tag{7-21}$$

Finally, the computed area and position are transformed to original scale without interpolation by Eqs.5-28, 5-29 and 5-30. Fig.7-33 and Fig.7-34 show the processing results for frame2 and frame4 to demonstrate the effectiveness of the proposed algorithm further.

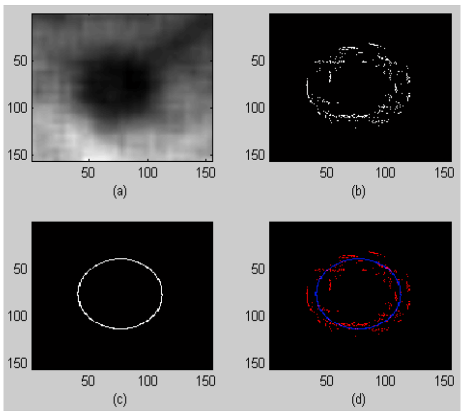


Fig.7-33 Frame 2 processing result

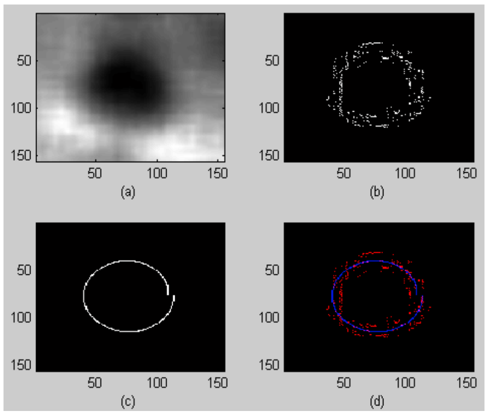


Fig.7-34 Frame 4 processing result

By comparing with this proposed algorithm with that proposed in Sect. 7.4.1, it is seen that the later proposed algorithm just calculate based on two dimensional information of the droplet image while the first proposed algorithm makes use of three dimensional information. Hence, the algorithm proposed in Sect. 7.4.1 is more accurate. As a result, the subsequent processing for tapering is omitted.

7.6 Results and Discussion

7.6.1 Droplet processing results

Ten adjacent images are processed to demonstrate the effectiveness of the proposed algorithm for the droplet. Figs. 7-35-7-40 show the processed results where (a), (b), (c), and (d) show the enhanced image, underestimate, overestimate, and the blurry regions respectively. Since some frames may not contain the droplet, the processing results just return none and the authors omit their figures of intermediate processing to make the demonstration concise. Fig.7-41 shows the plot of the computed position and size of the droplet.

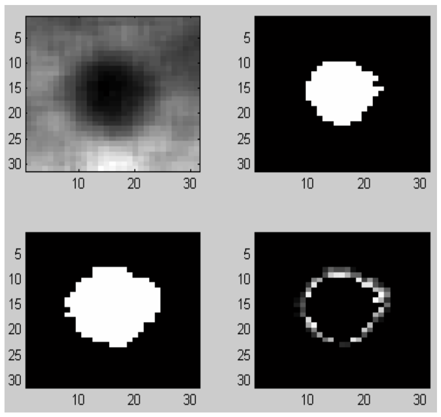


Fig.7-35 Processing result for frame 1

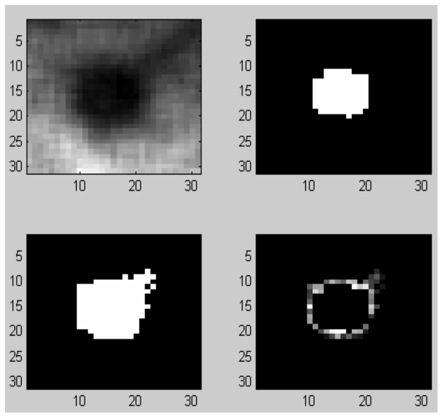


Fig.7-36 Processing result for frame 2

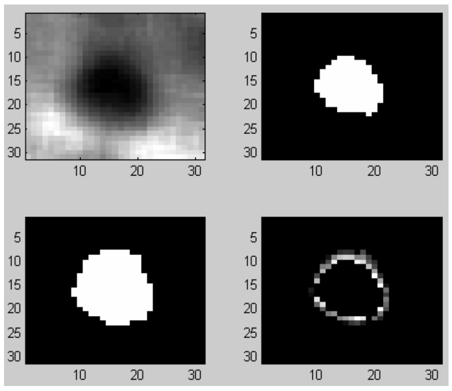


Fig.7-37 Processing result for frame 4

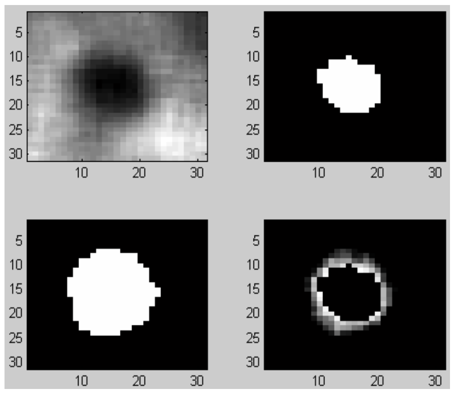


Fig.7-38 Processing result for frame 5

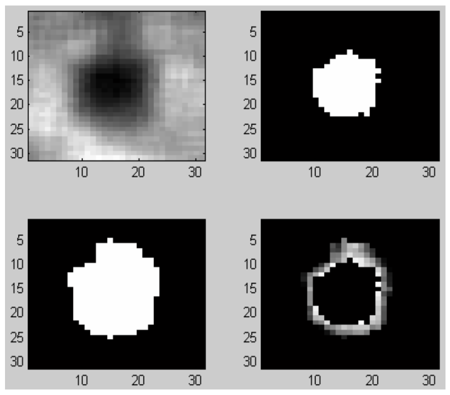


Fig.7-39 Processing result for frame 7

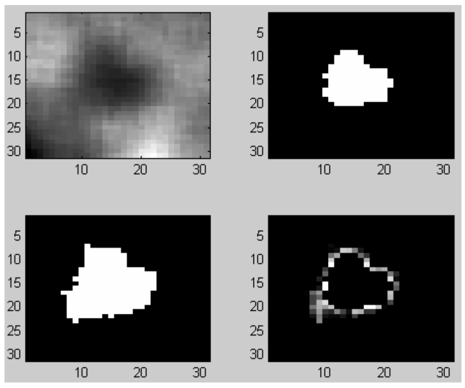


Fig.7-40 Processing result for frame 8

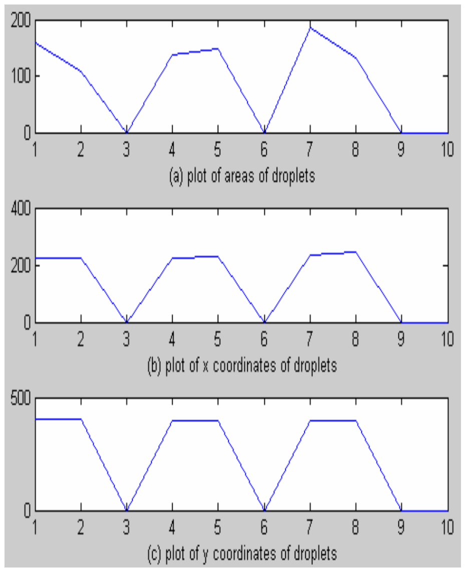


Fig.7-41 Plot of the computed area and position of droplet in 10 adjacent frames. Any horizontal coordinate (frame index) with a zero value in any of the three plots indicates that no droplet exists.

7.6.2 Undetached liquid processing results

The same 10 adjacent images are also used to demonstrate the effectiveness of the processing algorithm for the undetached liquid metal. Figs.7-42-7-51 show the processed results. Again, the enhanced image, underestimate, overestimate, and blurry regions in all the figures are shown in (a), (b), (c), and (d) respectively. The computation results are given in Fig.7-52.

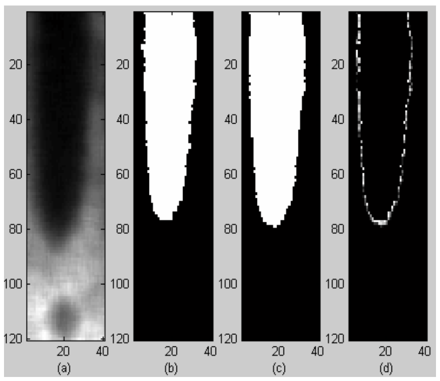


Fig.7-42 Processing results for frame 1

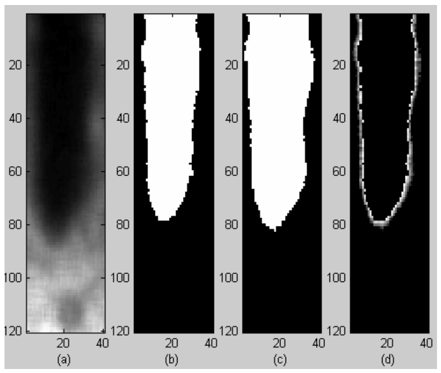


Fig.7-43 Processing results for frame 2

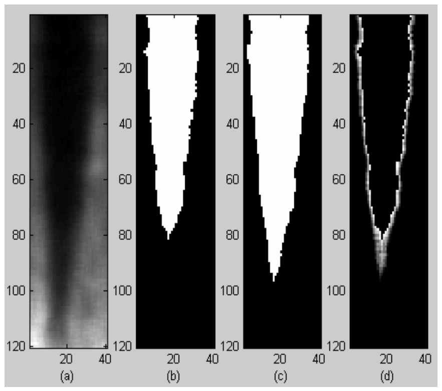


Fig.7-44 Processing results for frame 3

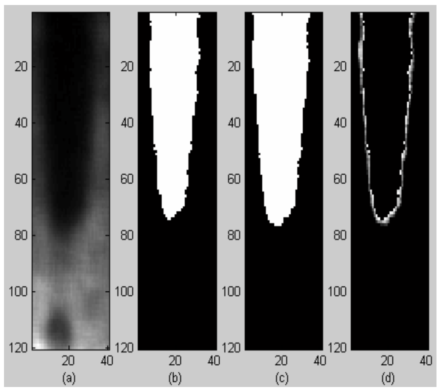


Fig.7-45 Processing results for frame 4

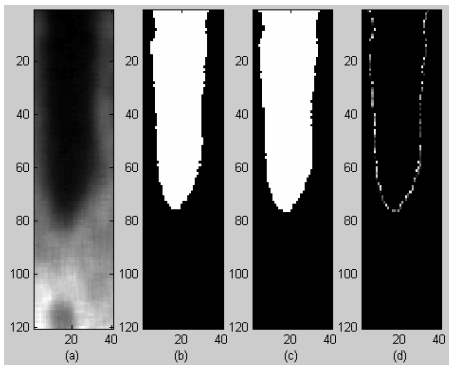


Fig.7-46 Processing results for frame 5

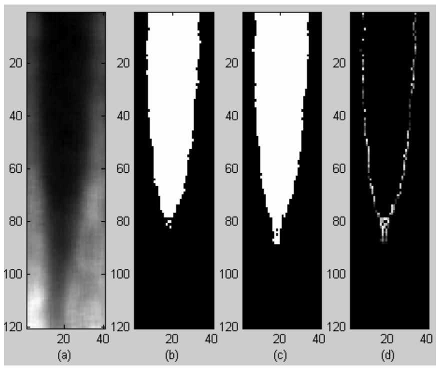


Fig.7-47 Processing results for frame 6

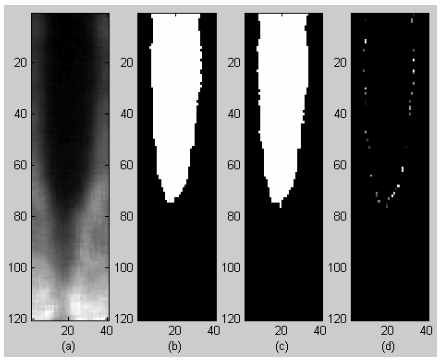


Fig.7-48 Processing results form frame 7

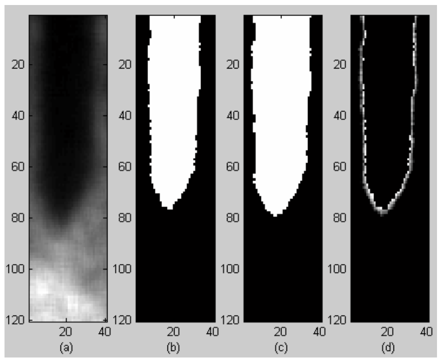


Fig.7-49 Processing results form frame 8

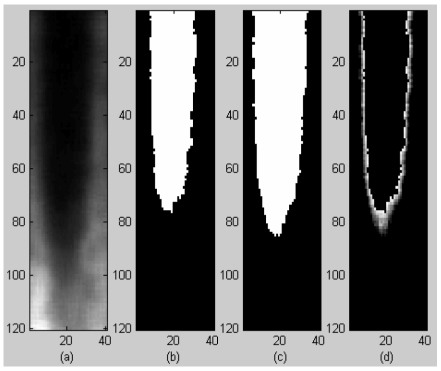


Fig.7-50 Processing results form frame 9

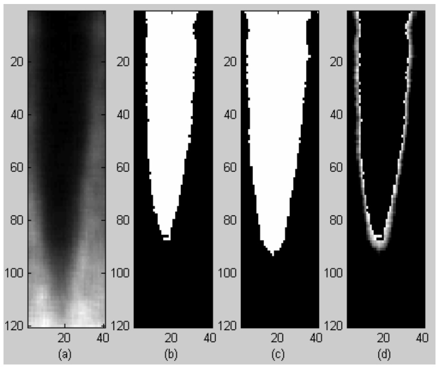


Fig.7-51 Processing results for frame 10

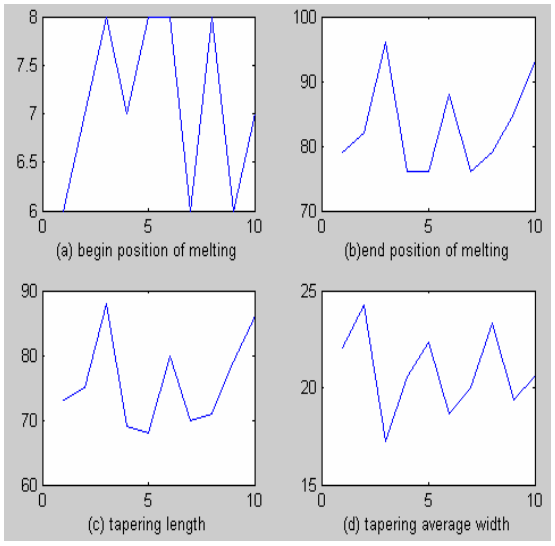


Fig.7-52 Plot of computed results for undetached liquid metal in 10 adjacent frames

7.6.3 Real time processing results

For the real time processing, the used frame grabber can achieve 420 frames per second grabbing speed. A Multiple buffering scheme is used to grab 40 frames with a frame rate of 420 frames per second into the buffers and then these images are processed one by one. After the images are processed, the frame grabber will grab another 40 frames of new images and then process them again. This loop continues until the maximum number frame which is specified by the user is reached or the program is closed by the user. The average grabbing and processing speed is about 100 frames per second.

7.6.4 Discussion

By comparing with the observed size and position of the droplet in the metal transfer coordinate system, the computed results appear to have adequate accuracies for the feedback control. Figs. 7-41-7-50 suggest that the computed tapering or significant tapering results are also sufficient. Fig.7-51 shows that, when there is a significant tapering, the peak value of the length is very obvious and easy to be detected and used as feedback for control. Hence, it appears that the proposed image processing algorithms have fulfilled the image processing tasks adequately. At the same time, the real time processing speed can achieve 420 frames per second grabbing rate for every 40 frames grabbed, which meets the frequency of bigger drop transfer in theory.

7.7 Conclusion

This chapter aims at establishing a capability needed to monitor the metal transfer process for real-time control. To this end, the complex optical system used in the laser back-lighting technique is simplified. The simplified technique reduces the image contrast but improves the applicability to be used in manufacturing environments. It is found that the reduced contrast can be easily re-gained through image enhancement to a level which is comparable with that achieved using the complex optical system. Hence, it appears that the simplified laser back-lighting technique provides a reasonable solution to acquire acceptable metal transfer images for real-time control.

To automatically process the images to obtain the feedback information needed in real-time control, the characteristics of the images are analyzed and have been used to design the algorithms to extract the detached droplet and un-detached liquid metal. In particular, because of the existence of the three zones for the grayness gradient, the two threshold method is proposed to define the laser refraction region and probability is introduced to fuzzily segment the image in the region. To improve the decision accuracy, the maximum slope difference principle is used to select optimal thresholds for the laser refraction regions both for the droplet and un-detached liquid metal. In addition, methods have been proposed to preserve the visual impacts for the observed gradient change characteristics from the inner shadow, across the laser refraction region, to the laser illuminated area.

Experimental results show that the proposed image processing algorithms can achieve the required processing speed and computation accuracy. In particular, for the hardware used, the proposed system and algorithms can accomplish the real time processing task with a grabbing speed up to 420 frames per second and grabbing with processing speed up to 100 frames per second.

CHAPTER 8

REAL TIME PROCESSING SYSTEM

8.1 Real Time Processing System

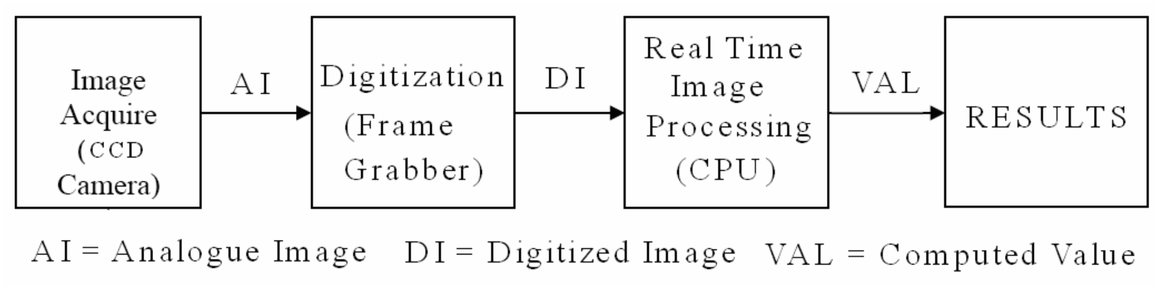


Image Acquire (CCD Camera)
AI
Digitization (Frame Grabber)
DI
Real Time Image Processing (CPU)
VAL
RESULTS
AI = Analogue Image
DI = Digitized Image
VAL = Computed Value

Fig. 8-1 The functional block diagram of the real time processing system

Fig. 8-1 shows the functional diagram of the real time processing system. It is composed of three parts: a computer, a frame grabber and a CCD camera.

Since it is for real time processing, the requirement for CPU is relatively high. Furthermore, part of RAM needs to be allocated to the frame grabber as the non-paged memory, so the requirement for the RAM is also high. The configuration of the used computer has been described in Chapter 7.

The used frame grabber in this research is Matrox Meteor-II/Digital which is part of the Matrox Meteor-II family of high performance frame grabbers for cost sensitive applications. Its maximum possible acquisition frame rate to host memory using a double buffer or multiple buffering function is 423 frames per second. Image data can be reformatted by Matrox VIA in real-time prior to transfer to host system or display.

The used camera is a CCD camera, CA-D6, and it has the following features:

- 260x260 or 532Hx516V pixels, 10μm square with 100% fill factor

• Frame transfer architecture and pixel reset—no shutter required

• 4 outputs at 25 MHz: frame rates to 955 or 262 frames/sec

• 8 bit digital data in EIA-644 (LVDS differential) format

• Separate connectors for power, control and data

• "Snapshot" operation

• Vertical antiblooming

8.2 Real time processing scheme

A digitizer is defined as the acquisition paths used to grab from one video source. Before grabbing from a digitizer, the Format property of its Digitizer control must be set to a digitizer configuration format (DCF) that corresponds to the camera. The DCF defines such settings as the input frequency and resolution, and will determine limits when grabbing an image. Once the resources of a Digitizer control are allocated, many of its properties are initialized to the settings specified in the DCF, unless you have set these properties to values other than the default.

To grab a sequence of frames in real-time, use successive, asynchronous calls to the Grab method, specifying a different image for each frame of the sequence. Alternatively, a single call can be made to the Multiple-Buffering-Process method of the Digitizer control. After a sequence of images are grabbed, the Write method of the AVIFileHandler Object can be used to export the sequence of images to an *.avi file.

Since the objective is to process in real time not just grab in real time, a real time processing scheme is needed. Fortunately, the Meteor II frame grabber supports a double buffering scheme which meets the requirement. Double buffering involves grabbing into one image while processing the previously grabbed image. It allows to grab and process either buffering at the same time. The destination of the grab between the two images is switched every time. In addition, a synchronization scheme is used for the grabbing and processing to make sure that:

1. An image will not be processed until an entire frame has been grabbed into its buffer.
2. The image should not be grabbed into the buffer until the previous frame in that buffer has been processed.

When a frame takes longer to process than the time required to grab the subsequent frame, a multiple buffering technique can be used to ensure that all processing is completed without losing any frames. To perform multiple buffering, the Multiple-Buffering-Process method of Digitizer control can be used. It allows a list of previously-allocated image controls to hold a series of sequentially grabbed images and process them as they are being grabbed. These grabs can either continue until stopped or continue until all image controls in the list are filled. This Multiple-Buffering-Process method grabs round-robin through the list of image controls; however, care must still be taken to assure that the average time it takes to process a frame is not greater than the frame rate of the camera, so that frames will not be missed.

Besides the round-robin grabbing and processing scheme, the Multiple-Buffering-Process also supports a more flexible scheme which allows grabbing a series of images into the buffer and then processing them one by one. The frame grabber will not grab another sequence of images into the buffer until all the images in the buffer have been processed. This processing scheme is quite suitable for the application in this research because the actual welding control frequency is much less than the frequency of grabbing. Suppose the control frequency is once every second. The processing result of one lifespan of a droplet may be enough to provide the necessary feedback information. In this situation, grabbing 20 frames and then processing them is adequate to compute the charateristic parameters as the feedback information. Using the proposed Brightness Based Selection and Edge Detection Enhancement Algorithm, it needs about 0.75 second to process these 20 frames. Since the used frame rate is 3000 frames/second for the metal transfer images obtained by direct view, the time needs to grab these 20 frames is 0.007 second. There will be about 0.24 second left for control data transmission which is sufficient. Please be noticed that all these simulation results are based on a general purpose PC. If special purpose computer system e.g. embedded system can be used, the processing speed will be much faster.

With the currently used general purpose computer, the control frequency can be decreased to make the control more robust. For example, if the control frequency is once every five seconds, then at least five groups of characteristic parameters can be computed as feedback information. In this situation, the average of these five groups of feedback information can be used for practical control. Because there might be some abnormal and instant conditions during the welding which will in turn affect the image processing result, the average feedback can increase the robustness of the control obviously.

After analyzing the feedback information obtained from the image processing system, the control system can judge if the current welding parameters are desired. If not, it can adjust the welding parameters based on the pre-knowledge accordingly. This pre-knowledge is acquired by experiments. For example, experiments can verify that with what current, voltage, diameter of welding wire and welding wire speed the welding process can produce the desired droplet size and detach rate. In turn, achieving the desired droplet size and detach rate will contribute to the desired welding quality.

CHAPTER 9

CONCLUTIONS AND CONTRIBUTIONS

9.1 Conclusions

Understanding, analysis, and control of GMAW process and its modifications reply on the availability of effective algorithms which can automatically process metal transfer images. This research aims at developing an automated algorithm to process the metal transfer image. It was found that traditional correlation method has difficulties to achieve a reasonable accuracy from metal transfer images. The proposed brightness based separation method can locate the droplets with adequate accuracy but its accuracy in droplet size computation is affected by the fact that the brightness of the droplets is typically not uniform and the droplets might not be brightest in the whole image. To improve the accuracy, the algorithm is modified by taking advantage of the correlation of adjacent images. Experimental results show that the revised algorithm which is referred to as the brightness and subtraction based separation algorithm can achieve adequate accuracy for both the droplet location and droplet size. In addition, it is found that the proposed algorithm has the potential to provide an adequate processing speed for being used in real-time metal transfer control.

The proposed brightness based selection and edge based separation algorithm can detect droplets from the image and try to detect adequate edge information from interpolated images. The proposed model for droplet edge gives an effective method to estimate the size of the droplet robustly and accurately and the proposed model validation assures that the model used meets a minimal accuracy requirement. Experimental results show that the proposed brightness based selection and edge detection based enhancement algorithm is effective in providing robust estimates with sufficient accuracy for possible applications in droplet size and detachment rate control as discussed in Chapter 5 and Chapter 8.

The proposed simplified laser back-lighting technique provides a reasonable solution to acquire acceptable metal transfer images for real-time control. The proposed image processing algorithms for the images acquired by the simplified laser back-lighting technique can also achieve the required processing speed and computation accuracy. Experimental results verified that the proposed system and algorithms can accomplish the real time processing task with a grabbing speed up to 420 frames per second and grabbing with processing speed up to 100 frames per second.

9.2 Contributions

This research work has the following contributions. First, some of the existing tracking algorithms are tested when trying to find an acceptable solution to this specific application. The failure of these algorithms makes it clear that tracking may not be suitable for such kind of applications. Second, robust and automatic algorithms are developed for metal transfer images captured by using a high speed camera solely. Although the proposed algorithms make use of some existing basic image processing techniques, they have novelty in the way how specific problems are solved effectively and in providing effective solutions to the challenging real-time metal transfer monitoring applications. These algorithms are also integrated into C++ programs. Once fast enough frame grabbers are available, they can be used for real time processing with first grabbing 40 frames and then processing scheme described in Chapter 7. Hence, the foundation for later real time control system is established. Third, robust and automatic algorithms are developed for metal transfer images captured by the simplified laser back-lighting technique and these algorithms are also implemented in C++ programs. In addition to the novelty as a solution, novel thresholding algorithms are proposed. Double thresholds and pixel probability are also novel for separating the fuzzy droplet and undetached liquid metal from the background.

APPENDICES

Appendix A

MImageSubtractAlgorithmDlg.cpp

```
///////////////////////////////////////////////////////////////////////////
///////
//  ActiveMIL Example:  MImageSubtractAlgorithm
//
//  Synopsis:   This program computes the position and size of the
droplets during metal //transfer using proposed subtraction based
separation algorithm
//
//  Note :      Look at the History logs of the ActiveMIL controls to
see
//              which properties were set at desing time.
//
///////////////////////////////////////////////////////////////////////////
///////

// MImageSubtractAlgorithmDlg.cpp : implementation file
//
#include "stdafx.h"
#include "MImageSubtractAlgorithm.h"
#include "MImageSubtractAlgorithmDlg.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#endif

//////////////////////
// ActiveMIL constants
//
// Location of images files
//
const char EXAMPLE_IMAGE[] = "recorded.mim";//"1.bmp";
const char EXAMPLE_AVI[] =
"D250_28_3_1uncomp.avi";//"metaltransfer1.avi";
```

```
// Standard display width
//
const int  DEFAULT_DISPLAY_WIDTH = 480;

// Image specifications
const int IMAGE_THRESHOLD_VALUE = 30;
const int IMAGE_SMALL_PARTICLE_RADIUS = 2;

// Zoom factor
//
const int ZOOM_FACTOR = 2;

// Number of procesing loops
//
const int NB_LOOP = 10;

// CAboutDlg dialog used for App About

class CAboutDlg : public CDialog
{
public:
    CAboutDlg();

// Dialog Data
    enum { IDD = IDD_ABOUTBOX };

    protected:
    virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV
support

// Implementation
protected:
    DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
```

```
{
    CDialog::DoDataExchange(pDX);
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
END_MESSAGE_MAP()

// CMImageSubtractAlgorithmDlg dialog

CMImageSubtractAlgorithmDlg::CMImageSubtractAlgorithmDlg(CWnd* pParent
/*=NULL*/)
    : CDialog(CMImageSubtractAlgorithmDlg::IDD, pParent)
    , m_CommentText("")
    , m_StepNumber(1)
{
    m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}

void CMImageSubtractAlgorithmDlg::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);

    DDX_Control(pDX, IDC_APPLICATION1, m_Application1);
    DDX_Control(pDX, IDC_SYSTEM1, m_System1);
    DDX_Control(pDX, IDC_IMAGE1, m_DisplayImage);
    DDX_Control(pDX, IDC_IMAGE2, m_SourceImage);
    DDX_Control(pDX, IDC_DISPLAY1, m_Display1);
    DDX_Control(pDX, IDC_GRAPHICCONTEXT1, m_GraphicContext1);
    DDX_Control(pDX, IDC_IMAGEPROCESSING1, m_ImageProcessing1);
    DDX_Control(pDX, IDC_SCROLLBAR_HORIZONTAL, m_HScrollBar);
    DDX_Control(pDX, IDC_SCROLLBAR_VERTICAL, m_VScrollBar);
    DDX_Text(pDX, IDC_COMMENTTEXT, m_CommentText);
}

BEGIN_MESSAGE_MAP(CMImageSubtractAlgorithmDlg, CDialog)
    ON_WM_SYSCOMMAND()
    ON_WM_PAINT()
    ON_WM_QUERYDRAGICON()
    //}}AFX_MSG_MAP
    ON_BN_CLICKED(IDC_NEXTBUTTON, OnBnClickedNextbutton)
    ON_WM_HSCROLL()
    ON_WM_VSCROLL()
END_MESSAGE_MAP()

// CMImageSubtractAlgorithmDlg message handlers

BOOL CMImageSubtractAlgorithmDlg::OnInitDialog()
{
    CDialog::OnInitDialog();

    // Add "About..." menu item to system menu.

    // IDM_ABOUTBOX must be in the system command range.
    ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
    ASSERT(IDM_ABOUTBOX < 0xF000);

    CMenu* pSysMenu = GetSystemMenu(FALSE);
    if (pSysMenu != NULL)
    {
        CString strAboutMenu;
        strAboutMenu.LoadString(IDS_ABOUTBOX);
        if (!strAboutMenu.IsEmpty())
        {
            pSysMenu->AppendMenu(MF_SEPARATOR);
            pSysMenu->AppendMenu(MF_STRING, IDM_ABOUTBOX, strAboutMenu);
        }
    }

    // Set the icon for this dialog.  The framework does this
automatically
    // when the application's main window is not a dialog
    SetIcon(m_hIcon, TRUE);         // Set big icon
    SetIcon(m_hIcon, FALSE);        // Set small icon

      CString ActiveMILImagesPath = static_cast<LPCTSTR>(m_Application1-
>ActiveMILImagesPath);
    CString AVIFilePath = ActiveMILImagesPath + EXAMPLE_AVI;
      //// Resize the dialog
      AdjustDialog(AVIFilePath);

    // Zoom the image to see the result of the image processing better
    //
    m_Display1->Zoom(ZOOM_FACTOR, ZOOM_FACTOR, FALSE);

    // Adjust the ScrollBar range
    //
    m_HScrollBar.SetScrollRange(0, (m_DisplayImage->SizeX / ZOOM_FACTOR)
- 1);
    m_VScrollBar.SetScrollRange(0, (m_DisplayImage->SizeY / ZOOM_FACTOR)
- 1);

    // Update the coment text
    //
    m_CommentText =    "This program computes the position and size of
the droplets during metal transfer using proposed subtraction based
separation algorithm\n"
                                         "Click <Next> or press <Enter>
to continue.\n";
    UpdateData(FALSE);

    return TRUE;  // return TRUE  unless you set the focus to a control
}

void CMImageSubtractAlgorithmDlg::OnSysCommand(UINT nID, LPARAM lParam)
{
    if ((nID & 0xFFF0) == IDM_ABOUTBOX)
    {
        CAboutDlg dlgAbout;
        dlgAbout.DoModal();
    }
    else
    {
        CDialog::OnSysCommand(nID, lParam);
    }
}

void CMImageSubtractAlgorithmDlg::OnPaint()
{
    if (IsIconic())
    {
        CPaintDC dc(this); // device context for painting
        SendMessage(WM_ICONERASEBKGND,
reinterpret_cast<WPARAM>(dc.GetSafeHdc()), 0);
        // Center icon in client rectangle
        int cxIcon = GetSystemMetrics(SM_CXICON);
        int cyIcon = GetSystemMetrics(SM_CYICON);
        CRect rect;
        GetClientRect(&rect);
        int x = (rect.Width() - cxIcon + 1) / 2;
        int y = (rect.Height() - cyIcon + 1) / 2;

        // Draw the icon
        dc.DrawIcon(x, y, m_hIcon);
    }
    else
    {
        CDialog::OnPaint();
    }
}

// The system calls this function to obtain the cursor to display while
the user drags
//  the minimized window.
HCURSOR CMImageSubtractAlgorithmDlg::OnQueryDragIcon()
{
    return static_cast<HCURSOR>(m_hIcon);
}

////////////////////////////////////////////////////////////////////////
/////
// ActiveMIL procedure: In response of clicking the "Next" button.
//                      Drives the example through each step,
//                      performing appropriate actions
void CMImageSubtractAlgorithmDlg::OnBnClickedNextbutton()
{
    // Increment the step counter and updates text in the dialog
    //
      CString strActiveMILImagesPath =
static_cast<LPCTSTR>(m_Application1->ActiveMILImagesPath);
    CString strActiveMILImageFile = strActiveMILImagesPath +
EXAMPLE_IMAGE;
```

```
      CString ActiveMILImagesPath = static_cast<LPCTSTR>(m_Application1-
>ActiveMILImagesPath);
    CString AVIFilePath = ActiveMILImagesPath + EXAMPLE_AVI;
      AVIHandler.CreateInstance("MIL.AVIFileHandler");

      AVIHandler->Open(_bstr_t(AVIFilePath), aviMJPEG,
aviRead);//aviDIB//aviMIL//aviMJPEG//aviFileFormatDefault
      AVIHandler->Read(_variant_t(m_SourceImage, true), 20,
1);//FrameNumber
      AVIHandler->Close();

   AdjustStepNumber();
    switch(m_StepNumber)
        {
        case 2:

      for (int i=1;i<=100;i++)
      {
              AVIHandler->Open(_bstr_t(AVIFilePath), aviMJPEG,
aviRead);//aviDIB//aviMIL//aviMJPEG//aviFileFormatDefault
              AVIHandler->Read(_variant_t(m_DisplayImage, true), i,
1);//FrameNumber
                  m_ImageProcessing1->Source1 = m_DisplayImage;
                  AVIHandler->Close();

                  m_ImageProcessing1->Source2 =  m_SourceImage;
                  m_ImageProcessing1->Subtract(TRUE,TRUE);

                  // Smooth the image to reduce noise
            m_ImageProcessing1->Smooth(impOverscanEnable);

            // Binarize the image so that particles are represented in
white
            // and the background in black
            m_ImageProcessing1->Binarize(impGreaterOrEqualTo,
IMAGE_THRESHOLD_VALUE, 0);//impLessOrEqualTo,IMAGE_THRESHOLD_VALUE,0

            // Remove the small particles
            //
            m_ImageProcessing1->Open(IMAGE_SMALL_PARTICLE_RADIUS,
impBinary);
```

```
			// Label the image
		m_ImageProcessing1->Label(imp8Connected, impGrayscale);

		// The largest label value corresponds to the number of 
particles in the image
		//
		m_ImageProcessing1->FindExtremes(FALSE, TRUE);
		VARIANT varMaxLabelNumber = m_ImageProcessing1->Results-
>Item(1);
		long NbParticles = (long) V_R4(&varMaxLabelNumber);

			double s=0;
			int a =0;
			double m=0;
			for (int i=1; i<=NbParticles; i++)
			{
			m_ImageProcessing1-
>CalculateStats(impNumberOfPixels,impInRange,i,i);
			m = m_ImageProcessing1->Results->Stats-
>GetNumberOfPixels();
			if (s<=m)
			{
				s=m;
				a =i;
			}
			}
			 m_ImageProcessing1->Free();
			 m_ImageProcessing1->ResultType = impAnyResultType;
			 //m_ImageProcessing1->ResultType = impExtremeList;
			 m_ImageProcessing1->ResultSize = 2000;
			 m_ImageProcessing1->Allocate();
			m_ImageProcessing1->LocateEvents(impInRange,a,a);

			double sumx=0;
			double sumy=0;
			double isize=0;
			if (s<=2000)
				isize=s;
			else
				isize=2000;
```

```
                  for (int j=1;j<=isize;j++)
                  {
            ILocateEventsResultPtr ResultItem =
            (ILocateEventsResultPtr)(V_DISPATCH(&m_ImageProcessing1-
>Results->Item(j)));
                  double PositionX =ResultItem->GetPositionX();
                  double PositionY =ResultItem->GetPositionY();
                  sumx+=PositionX;
                  sumy+=PositionY;
                  }
                  sumx/=isize;
                  sumy/=isize;

              FILE * fp;
              fp=fopen("finalresult.txt","a");
              fprintf(fp, "%ld\t %lf\t %lf\t %lf\n",a,s,sumx,sumy);
              fclose(fp);

            // Display the result in pseudo-color
            //
            m_Display1->SelectedLUT = dispPseudoColorLUT;
            m_Display1->LUTEnabled = TRUE;
      }
                  default:
                // Close dialog application
                //
                EndDialog(IDOK);
                break;

        }
}

/////////////////////////////////////////////////////////////////////////
/////
// ActiveMIL procedure: Increments the step counter and updates text in
//                      the dialog according to the new value
//
void CMImageSubtractAlgorithmDlg::AdjustStepNumber()
{
    // Increment the step count
    //
    m_StepNumber++;

    // Write the step count in the comment frame
    //
    CString str;
    str.Format("Step %d:", m_StepNumber);
    CWnd *pStepNumber = GetDlgItem(IDC_STEPNUMBER);
    pStepNumber->SetWindowText(str);
}

////////////////////////////////////////////////////////////////////////
///////
// ActiveMIL procedure :    Calculate how to resize the dialog in order
to
//                          view all element in all resolutions and font
sizes
//
void CMImageSubtractAlgorithmDlg::AdjustDialog(CString ImagePath)
{
    // Set the icon for this dialog.  The framework does this
automatically
    // when the application's main window is not a dialog
    //
    HICON hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
    SetIcon(hIcon, TRUE);           // Set big icon
    SetIcon(hIcon, FALSE);          // Set small icon

    // Retrieve the relevant dialog controls
    //
    CWnd *pNextButton = GetDlgItem(IDC_NEXTBUTTON);
    CWnd *pStepNumber = GetDlgItem(IDC_COMMENTTEXT);

    int MimImageSizeX = m_DisplayImage->FileInquire((LPCTSTR) ImagePath,
imSizeX);
    int MimImageSizeY = m_DisplayImage->FileInquire((LPCTSTR) ImagePath,
imSizeY);
    long nMaxDisplayWidth;
    long nMaxNextStepWidth;
    CRect rectNextButton;
    CRect rectCommentFrame;
    CRect rectDisplay;
    CRect rectDialog;
    CRect rectHScrollBar;
    CRect rectVScrollBar;

    // Adjust the size of the Display
    //
    m_Display1.SetWindowPos(NULL,                               //
pWndInsertAfter
                            0,                                  // x
                            0,                                  // y
                            MimImageSizeX,                      // cx
                            MimImageSizeY,                      // cy
                            SWP_NOMOVE | SWP_NOOWNERZORDER);    // flags

    // Get the sizes of relevant elements and calculate how to resize
the
    // dialog in order to view all elements, in all resolutions and font
sizes
    //
    pNextButton->GetWindowRect(&rectNextButton);
    pStepNumber->GetWindowRect(&rectCommentFrame);
    m_Display1.GetWindowRect(&rectDisplay);
    m_HScrollBar.GetWindowRect(&rectHScrollBar);
    m_VScrollBar.GetWindowRect(&rectVScrollBar);
    GetWindowRect(&rectDialog);

    // Convert the display's rectangle to Client coordinates
    //
    ScreenToClient(&rectDisplay);

    if(!(rectDisplay.Width() == 32 && rectDisplay.Height() == 32))
        {
        // Adjust the ScrollBar size and position
        //
        m_HScrollBar.SetWindowPos(  NULL,
                                    rectDisplay.left,
                                    rectDisplay.bottom,
                                    MimImageSizeX,
                                    rectHScrollBar.Height(),
                                    SWP_NOOWNERZORDER);
        m_VScrollBar.SetWindowPos(  NULL,
                                    rectDisplay.right,
                                    rectDisplay.top,
                                    rectVScrollBar.Width(),
                                    MimImageSizeY,
                                    SWP_NOOWNERZORDER);
        }
    else
        {
        m_HScrollBar.ShowWindow(SW_HIDE);
        m_VScrollBar.ShowWindow(SW_HIDE);
        }

    // Convert back to screen coordinates
    //
    ClientToScreen(&rectDisplay);

    // Check the width of the display, if it is big enough, we will
    // adjust the comment text frame and the next button
    // to fit the width of the diplay
    //
    if(rectDisplay.Width() > DEFAULT_DISPLAY_WIDTH)
        {
        // Calculate the size of the window displacements
        //
        int SizeChange = rectDisplay.right + rectVScrollBar.Width() -
rectNextButton.right;
        // Change the sizes of the Next button and the comment frame
        //
        rectNextButton.OffsetRect(SizeChange, 0);
        rectCommentFrame.InflateRect(0, 0, SizeChange, 0);
        // Move the windows
        //
        ScreenToClient(&rectNextButton);
        ScreenToClient(&rectCommentFrame);

        pNextButton->MoveWindow(&rectNextButton, FALSE);
        pStepNumber->MoveWindow(&rectCommentFrame, FALSE);

        // Reconvert the rectangles to screen coordinates
        //
        ClientToScreen(&rectCommentFrame);
        ClientToScreen(&rectNextButton);
        }

    nMaxDisplayWidth  = rectDisplay.Width() +
                        2 * abs(rectDialog.left - rectDisplay.left);

    nMaxNextStepWidth = rectNextButton.left - rectDialog.left +
                        rectNextButton.Width() +
                        rectCommentFrame.left - rectDialog.left;

    if (nMaxNextStepWidth > nMaxDisplayWidth)
        {
        SetWindowPos(NULL,                                      //
pWndInsertAfter
                    0,                                          // x
                    0,                                          // y
                    nMaxNextStepWidth,                          // cx
                    rectDisplay.Height() +
                    rectDisplay.top - rectDialog.top +
                    rectHScrollBar.Height() +
                    rectDisplay.left - rectDialog.left,         // cy
                    SWP_SHOWWINDOW);                            // flags
        }
    else
        {
        SetWindowPos(NULL,                                      //
pWndInsertAfter
                    0,                                          // x
                    0,                                          // y
                    nMaxDisplayWidth,  // cx
                    rectDisplay.Height() +
                    rectDisplay.top - rectDialog.top +
                    rectHScrollBar.Height() +
                    abs (rectDisplay.left - rectDialog.left),   // cy
                    SWP_SHOWWINDOW);                            // flags
        }
}
```

```
////////////////////////////////////////////////////////////////////////
///////
// ActiveMIL procedure :    Reposition the ScrollBar and Pan the display
when
//                          the user Scrolls the Horizontal ScrollBar
//
void CMImageSubtractAlgorithmDlg::OnHScroll(UINT nSBCode, UINT nPos,
CScrollBar* pScrollBar)
{
    // Calculate the new position of the scroll bar
    //
    int nCurPos = ScrollFunction(nSBCode, nPos, pScrollBar);

    // Pan the display accordingly
    //
    m_Display1->PanX = nCurPos;

    CDialog::OnHScroll(nSBCode, nPos, pScrollBar);
}

////////////////////////////////////////////////////////////////////////
///////
// ActiveMIL procedure :    Reposition the ScrollBar and Pan the display
when
//                          the user Scrolls the Vertical ScrollBar
//
void CMImageSubtractAlgorithmDlg::OnVScroll(UINT nSBCode, UINT nPos,
CScrollBar* pScrollBar)
{
    // Calculate the new position of the scroll bar
    //
    int nCurPos = ScrollFunction(nSBCode, nPos, pScrollBar);

    // Pan the display accordingly
    //
    m_Display1->PanY = nCurPos;

    CDialog::OnVScroll(nSBCode, nPos, pScrollBar);
}
```

```
////////////////////////////////////////////////////////////////////////
///////
// ActiveMIL procedure :    Calculates the new position of a scroll bar,
//                          avoiding to Scroll over the size of the
image
//
int CMImageSubtractAlgorithmDlg::ScrollFunction(UINT nSBCode, UINT nPos,
CScrollBar* pScrollBar)
{
    // Get the minimum and maximum scroll-bar positions.
    //
    int nMinPos;
    int nMaxPos;
    pScrollBar->GetScrollRange(&nMinPos, &nMaxPos);

    // Get the current position of scroll box.
    //
    int nCurPos = pScrollBar->GetScrollPos();

    // Determine the new position of scroll box.
    //
    switch (nSBCode)
        {
        case SB_LEFT:           // Scroll to far left.
            nCurPos = nMinPos;
            break;

        case SB_RIGHT:          // Scroll to far right.
            nCurPos = nMaxPos;
            break;

        case SB_LINELEFT:       // Scroll left.
            if (nCurPos > nMinPos)
                nCurPos--;
            break;

        case SB_LINERIGHT:      // Scroll right.
            if (nCurPos < nMaxPos)
                nCurPos++;
            break;
        case SB_PAGELEFT:      // Scroll one page left.
            if (nCurPos > nMinPos)
                {
                nCurPos = max(nMinPos, nCurPos - (int) nMaxPos / 4);
                }
            break;

        case SB_PAGERIGHT:     // Scroll one page right.
            if (nCurPos < nMaxPos)
                {
                nCurPos = min(nMaxPos, nCurPos + (int) nMaxPos / 4);
                }
            break;

        case SB_THUMBPOSITION:  // Scroll to absolute position. nPos is
the position
            nCurPos = nPos;     // of the scroll box at the end of the
drag operation.
            break;

        case SB_THUMBTRACK:     // Drag scroll box to specified
position. nPos is the
            nCurPos = nPos;     // position that the scroll box has been
dragged to.
            break;

        case SB_ENDSCROLL:      // End scroll.
            break;
        }

    // Set the new position of the thumb (scroll box).
    //
    pScrollBar->SetScrollPos(nCurPos);

    return nCurPos;
}
```

Appendix B

MImageEdgedetectionAlgorithmDlg.cpp

```
//////////////////////////////////////////////////////////////////////////
///////
//  ActiveMIL Example:  MImageEdgedetectionAlgorithm
//
//  Synopsis:   This program detects the edge in the image,the droplet's
edge is choosen //based on the maximum missing angle and fitted with a
second order curve. Finally its //position and size are computed.
//
//  Note :      Look at the History logs of the ActiveMIL controls to
see
//              which properties were set at desing time.
//
//////////////////////////////////////////////////////////////////////////
///////

// MImageEdgedetectionAlgorithmDlg.cpp : implementation file
//
#include "stdafx.h"
#include "MImageEdgedetectionAlgorithm.h"
#include "MImageEdgedetectionAlgorithmDlg.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#endif

#define pi 3.1416
//////////////////////
// ActiveMIL constants
//
// Location of images files
//
const char EXAMPLE_IMAGE[] = "transfer2.mim";//"Cell.mim";//"1.bmp";
const char EXAMPLE_AVI[] = "D250_28_3_1uncomp.avi";

// Standard display width
//
```

```
const int  DEFAULT_DISPLAY_WIDTH = 480;

// Image specifications
const int IMAGE_THRESHOLD_VALUE = 128;
const int IMAGE_SMALL_PARTICLE_RADIUS = 2;

// CAboutDlg dialog used for App About

class CAboutDlg : public CDialog
{
public:
    CAboutDlg();

// Dialog Data
    enum { IDD = IDD_ABOUTBOX };

    protected:
    virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV
support

// Implementation
protected:
    DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
END_MESSAGE_MAP()

// CMImageEdgedetectionAlgorithmDlg dialog
```

```
CMImageEdgedetectionAlgorithmDlg::CMImageEdgedetectionAlgorithmDlg(CWnd*
pParent /*=NULL*/)
    : CDialog(CMImageEdgedetectionAlgorithmDlg::IDD, pParent)
    , m_CommentText("")
    , m_StepNumber(1)
{
    m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}

void CMImageEdgedetectionAlgorithmDlg::DoDataExchange(CDataExchange*
pDX)
{
    CDialog::DoDataExchange(pDX);

    DDX_Control(pDX, IDC_APPLICATION1, m_Application1);
    DDX_Control(pDX, IDC_SYSTEM1, m_System1);
    DDX_Control(pDX, IDC_IMAGE1, m_Image1);
    DDX_Control(pDX, IDC_DISPLAY1, m_Display1);
    DDX_Text(pDX, IDC_COMMENTTEXT, m_CommentText);
    DDX_Control(pDX, IDC_IMAGEPROCESSING1, m_ImageProcessing1);
//    DDX_Control(pDX, IDC_LIST1, m_listResults);
}

BEGIN_MESSAGE_MAP(CMImageEdgedetectionAlgorithmDlg, CDialog)
    ON_WM_SYSCOMMAND()
    ON_WM_PAINT()
    ON_WM_QUERYDRAGICON()
    //}}AFX_MSG_MAP
    ON_BN_CLICKED(IDC_NEXTBUTTON, OnBnClickedNextbutton)
END_MESSAGE_MAP()

// CMImageEdgedetectionAlgorithmDlg message handlers

BOOL CMImageEdgedetectionAlgorithmDlg::OnInitDialog()
{
    CDialog::OnInitDialog();

    // Add "About..." menu item to system menu.

    // IDM_ABOUTBOX must be in the system command range.
    ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
    ASSERT(IDM_ABOUTBOX < 0xF000);

    CMenu* pSysMenu = GetSystemMenu(FALSE);
    if (pSysMenu != NULL)
    {
        CString strAboutMenu;
        strAboutMenu.LoadString(IDS_ABOUTBOX);
        if (!strAboutMenu.IsEmpty())
        {
            pSysMenu->AppendMenu(MF_SEPARATOR);
            pSysMenu->AppendMenu(MF_STRING, IDM_ABOUTBOX, strAboutMenu);
        }
    }

    // Set the icon for this dialog.  The framework does this
automatically
    //  when the application's main window is not a dialog
    SetIcon(m_hIcon, TRUE);            // Set big icon
    SetIcon(m_hIcon, FALSE);        // Set small icon

    /////////////////////////////////////
    // ActiveMIL example initialization
    //
    // Get the example images path
    //
    CString strActiveMILImagesPath =
static_cast<LPCTSTR>(m_Application1->ActiveMILImagesPath);
    CString strActiveMILImageFile = strActiveMILImagesPath +
EXAMPLE_IMAGE;

    // Resize the dialog
    //
    // AdjustDialog(strActiveMILImageFile);

      CString ActiveMILImagesPath = static_cast<LPCTSTR>(m_Application1-
>ActiveMILImagesPath);
    CString AVIFilePath = ActiveMILImagesPath + EXAMPLE_AVI;
      //// Resize the dialog
      AdjustDialog(AVIFilePath);

    // Update the coment text
    //
    m_CommentText =    " This program detects the edge in the image,the
droplet's edge is choosen based on the maximum missing angle and fitted
with a second order curve. Finally its position and size are computed\n"
              "Click <Next> or press <Enter> to continue.\n";
    UpdateData(FALSE);

    return TRUE;  // return TRUE  unless you set the focus to a control
}

void CMImageEdgedetectionAlgorithmDlg::OnSysCommand(UINT nID, LPARAM
lParam)
{
    if ((nID & 0xFFF0) == IDM_ABOUTBOX)
    {
        CAboutDlg dlgAbout;
        dlgAbout.DoModal();
    }
    else
    {
        CDialog::OnSysCommand(nID, lParam);
    }
}

// If you add a minimize button to your dialog, you will need the code
below
//  to draw the icon.  For MFC applications using the document/view
model,
//  this is automatically done for you by the framework.

void CMImageEdgedetectionAlgorithmDlg::OnPaint()
{
    if (IsIconic())
    {
        CPaintDC dc(this); // device context for painting

        SendMessage(WM_ICONERASEBKGND,
reinterpret_cast<WPARAM>(dc.GetSafeHdc()), 0);

        // Center icon in client rectangle
        int cxIcon = GetSystemMetrics(SM_CXICON);
        int cyIcon = GetSystemMetrics(SM_CYICON);
        CRect rect;
        GetClientRect(&rect);
        int x = (rect.Width() - cxIcon + 1) / 2;
        int y = (rect.Height() - cyIcon + 1) / 2;

        // Draw the icon
        dc.DrawIcon(x, y, m_hIcon);
    }
    else
    {
        CDialog::OnPaint();
    }
}

// The system calls this function to obtain the cursor to display while
the user drags
//  the minimized window.
HCURSOR CMImageEdgedetectionAlgorithmDlg::OnQueryDragIcon()
{
    return static_cast<HCURSOR>(m_hIcon);
}

////////////////////////////////////////////////////////////////////////
/////
// ActiveMIL procedure: In response of clicking the "Next" button.
//                      Drives the example through each step,
//                      performing appropriate actions
//

void CMImageEdgedetectionAlgorithmDlg::OnBnClickedNextbutton()
{
    // Increment the step counter and updates text in the dialog
    //
      CString strActiveMILImagesPath =
static_cast<LPCTSTR>(m_Application1->ActiveMILImagesPath);
    CString strActiveMILImageFile = strActiveMILImagesPath +
EXAMPLE_IMAGE;
    AdjustStepNumber();
```

```
      CString ActiveMILImagesPath = static_cast<LPCTSTR>(m_Application1-
>ActiveMILImagesPath);
    CString AVIFilePath = ActiveMILImagesPath + EXAMPLE_AVI;
      AVIHandler.CreateInstance("MIL.AVIFileHandler");

    switch(m_StepNumber)
        {
        case 2:
                  for (int i=1;i<=100;i++)
      {

              AVIHandler->Open(_bstr_t(AVIFilePath), aviDIB,
aviRead);//aviDIB//aviMIL//aviMJPEG//aviFileFormatDefault
              AVIHandler->Read(_variant_t(m_Image1, true), i,
1);//FrameNumber
                  m_ImageProcessing1->Source1 = m_Image1;
                  AVIHandler->Close();

            // Smooth the image to reduce noise
            m_ImageProcessing1->Smooth(impOverscanEnable);

                  // perform the edge detection
                  m_ImageProcessing1-
>EdgeDetect(FALSE,FALSE,0,impOverscanDisable);
                  m_ImageProcessing1->Binarize(impGreaterOrEqualTo,
IMAGE_THRESHOLD_VALUE+20, 0);

            // Label the image
            //
            m_ImageProcessing1->Label(imp8Connected, impGrayscale);

            // The largest label value corresponds to the number of
particles in the image
            //
            m_ImageProcessing1->FindExtremes(FALSE, TRUE);
            VARIANT varMaxLabelNumber = m_ImageProcessing1->Results-
>Item(1);
            long NbParticles = (long) V_R4(&varMaxLabelNumber);
```

```
                    double px=0;
                    double py=0;
                    double area=0;
                    double angc=pi/4;

                    for (int i=1; i<=NbParticles; i++)
                    {
                          double m=0;
                          m_ImageProcessing1-
>CalculateStats(impNumberOfPixels,impInRange,i,i);
                        m = m_ImageProcessing1->Results->Stats-
>GetNumberOfPixels();
                          if (m>10)
                          {
                     m_ImageProcessing1->Free();
                     m_ImageProcessing1->ResultType = impAnyResultType;
                     m_ImageProcessing1->ResultSize = 200;
                     m_ImageProcessing1->Allocate();
                     m_ImageProcessing1->LocateEvents(impInRange,i,i);

                    double sumx=0;
                    double sumy=0;
                    double isize=0;
                    if (m<=200)
                          isize = m;
                    else
                          isize=200;

                    double xarray[200];
                    double yarray[200];
                    double sumarray[200];
                    double minx = 1000;
                    double miny = 1000;
                    double maxx = 0;
                    double maxy = 0;

                    for (int j=1;j<=isize;j++)
                    {
              ILocateEventsResultPtr ResultItem =
              (ILocateEventsResultPtr)(V_DISPATCH(&m_ImageProcessing1-
>Results->Item(j)));
```

```
double PositionX =ResultItem->GetPositionX();
double PositionY =ResultItem->GetPositionY();
xarray[j] = PositionX;
yarray[j] = PositionY;
sumx+=PositionX;
sumy+=PositionY;
if (minx > PositionX)
      minx = PositionX;
if (miny > PositionY)
      miny = PositionY;
if (maxx < PositionX)
      maxx = PositionX;
if (maxy < PositionY)
      maxy = PositionY;
}

sumx/=isize;
sumy/=isize;

double xdis=maxx-minx;
double ydis=maxy-miny;
if ((xdis<(ydis+5))&&(ydis<(xdis+5))&&sumy>200)
{

Matrix xo(isize, 1);
Matrix yo(isize,1);
for (int i=1; i<=isize; i++)
{
      xo(i,1)=(double) (xarray[i]-sumx);
      yo(i,1)=(double) (yarray[i]-sumy);
}

Matrix ang(isize,1);
for (int j=1; j<=isize; j++)
{
      ang(j,1) = (double) atan(yo(j,1)/xo(j,1));
}

Matrix th(isize, 3);
for (int k=1; k<=isize; k++)
{
```

```
                th(k,1) = (double) 1;
                th(k,2) = (double) ang(k,  1);
                th(k,3) = (double) (ang(k,1)*ang(k,1));
            }

            Matrix dradius(isize,1);
            for (int h=1; h<=isize; h++)
            {
                dradius(h,1) = (double)
sqrt(xo(h,1)*xo(h,1)+yo(h,1)*yo(h,1));
            }

            Matrix inter1 = th.t();
            Matrix inter2 = inter1*th;
            Matrix inter3 = inter2.i();
            Matrix inter4 = inter3*inter1;

            Matrix amatrix = inter4*dradius;

            double dsize=0;
            double a1 = (double) amatrix(1,1);
            double a2 = (double) amatrix(2,1);
            double a3 = (double) amatrix(3,1);
            dsize =
a1*a1*pi+(1/3)*a2*a2*(pi*pi*pi)+(1/5)*a3*a3*pi*pi*pi*pi*pi+(2/3)*a1*a3*p
i*pi*pi;

            Matrix sortedang(isize,1);
            for (int i=1; i<=isize; i++)
            {
                if (xo(i,1)>0 && yo(i,1)>0)
                {
                    sortedang(i,1)=(double)
(atan(yo(i,1)/xo(i,1)));
                }
                if (xo(i,1)>0 && yo(i,1)<0)
                {
                    sortedang(i,1)=(double)
(2*pi+atan(yo(i,1)/xo(i,1)));
                }
                if (xo(i,1)<0 && yo(i,1)>0)
```

```
                                {
                                        sortedang(i,1)=(double)
(pi+atan(yo(i,1)/xo(i,1)));
                                }
                                if (xo(i,1)<0 && yo(i,1)<0)
                                {
                                        sortedang(i,1)=(double) (3*pi/2-
atan(yo(i,1)/xo(i,1)));
                                }
                        }
                        sort_ascending(sortedang);
                        Matrix diff(isize,1);
                        for (int i=1; i<=(isize-1);i++)
                        {
                                diff(i,1)=sortedang(i+1,1)-sortedang(i,1);
                        }
                        diff(isize,1)=2*pi-sortedang(isize,1)+sortedang(1,1);

                        double maxang = diff.maximum();

                        if (maxang < angc)
                        {
                                px=sumx;
                                py=sumy;
                                area=dsize;
                                angc=maxang;
                        }
                        }
                        }
                                }

                     FILE * fp2;
                     fp2=fopen("finalresult.txt","a");
                     fprintf(fp2, "%ld\t %lf\t %lf\t %lf\t
%lf\n",NbParticles,px,py,area, angc);//m_Digitizer1->MultipleBuffering-
>NumberOfFrameProcessed);
                     fclose(fp2);

                // Display the result in pseudo-color
                m_Display1->SelectedLUT = dispPseudoColorLUT;
                m_Display1->LUTEnabled = TRUE;
```

```
      }
                  default:
                // Close dialog application
                //
                EndDialog(IDOK);
                break;

        }
}

////////////////////////////////////////////////////////////////////////
/////
// ActiveMIL procedure: Increments the step counter and updates text in
//                      the dialog according to the new value
//
void CMImageEdgedetectionAlgorithmDlg::AdjustStepNumber()
{
    // Increment the step count
    //
    m_StepNumber++;

    // Write the step count in the comment frame
    //
    CString str;
    str.Format("Step %d:", m_StepNumber);
    CWnd *pStepNumber = GetDlgItem(IDC_STEPNUMBER);
    pStepNumber->SetWindowText(str);
}

////////////////////////////////////////////////////////////////////////
///////
// ActiveMIL procedure :    Calculate how to resize the dialog in order
to
//                          view all element in all resolutions and font
sizes
//
void CMImageEdgedetectionAlgorithmDlg::AdjustDialog(CString ImagePath)
{
    // Set the icon for this dialog.  The framework does this
automatically
    // when the application's main window is not a dialog
    //
    HICON hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
    SetIcon(hIcon, TRUE);           // Set big icon
    SetIcon(hIcon, FALSE);          // Set small icon

    // Retrieve the relevant dialog controls
    //
    CWnd *pNextButton = GetDlgItem(IDC_NEXTBUTTON);
    CWnd *pStepNumber = GetDlgItem(IDC_COMMENTTEXT);

    int MimImageSizeX = m_Image1->FileInquire((LPCTSTR) ImagePath,
imSizeX);
    int MimImageSizeY = m_Image1->FileInquire((LPCTSTR) ImagePath,
imSizeY);
    long nMaxDisplayWidth;
    long nMaxNextStepWidth;
    CRect rectNextButton;
    CRect rectCommentFrame;
    CRect rectDisplay;
    CRect rectDialog;

    // Adjust the size of the Display
    //
    m_Display1.SetWindowPos(NULL,                              //
pWndInsertAfter
                            0,                                 // x
                            0,                                 // y
                            MimImageSizeX,                     // cx
                            MimImageSizeY,                     // cy
                            SWP_NOMOVE | SWP_NOOWNERZORDER);    // flags

    // Get the sizes of relevant elements and calculate how to resize
the
    // dialog in order to view all elements, in all resolutions and font
sizes
    //
    pNextButton->GetWindowRect(&rectNextButton);
    pStepNumber->GetWindowRect(&rectCommentFrame);
    m_Display1.GetWindowRect(&rectDisplay);
    GetWindowRect(&rectDialog);
    // Check the width of the display, if it is big enough, we will
    // adjust the comment text frame and the next button
    // to fit the width of the diplay
    //
    if(rectDisplay.Width() > DEFAULT_DISPLAY_WIDTH)
        {
        // Calculate the size of the window displacements
        //
        int SizeChange = rectDisplay.right - rectNextButton.right;

        // Change the sizes of the Next button and the comment frame
        //
        rectNextButton.OffsetRect(SizeChange, 0);
        rectCommentFrame.InflateRect(0, 0, SizeChange, 0);

        // Move the windows
        //
        ScreenToClient(&rectNextButton);
        ScreenToClient(&rectCommentFrame);

        pNextButton->MoveWindow(&rectNextButton, FALSE);
        pStepNumber->MoveWindow(&rectCommentFrame, FALSE);

        // Reconvert the rectangles to screen coordinates
        //
        ClientToScreen(&rectCommentFrame);
        ClientToScreen(&rectNextButton);
        }

    nMaxDisplayWidth  = rectDisplay.Width() +
                        2 * abs(rectDialog.left - rectDisplay.left);

    nMaxNextStepWidth = rectNextButton.left - rectDialog.left +
                        rectNextButton.Width() +
                        rectCommentFrame.left - rectDialog.left;

    if (nMaxNextStepWidth > nMaxDisplayWidth)
        {
        SetWindowPos(NULL,                                          //
pWndInsertAfter
                        0,                                          // x
```

```
                    0,                                          // y
                    nMaxNextStepWidth,                          // cx
                    rectDisplay.Height() +
                    rectDisplay.top - rectDialog.top +
                    rectDisplay.left - rectDialog.left,         // cy
                    SWP_SHOWWINDOW);                            // flags
        }
    else
        {
        SetWindowPos(NULL,                                      //
pWndInsertAfter
                    0,                                          // x
                    0,                                          // y
                    nMaxDisplayWidth,                           // cx
                    rectDisplay.Height() +
                    rectDisplay.top - rectDialog.top +
                    abs (rectDisplay.left - rectDialog.left),   // cy
                    SWP_SHOWWINDOW);                            // flags
        }
}
```

Appendix C

MdigProcessDlg.cpp

```
///////////////////////////////////////////////////////////////////////////
///////
//  ActiveMIL Example:  MDigProcess
//
//  Synopsis:   This program shows the usage of the
Digitizer.MultipleBuffering
//              property to do real-time processing.The edge detection
algorithm
//              is integrated to test the processing speed.
//
//  Note :      Look at the History logs of the ActiveMIL controls to
see
//              which properties were set at desing time.
//
///////////////////////////////////////////////////////////////////////////
///////

// MDigProcessDlg.cpp : implementation file
//

#include "stdafx.h"
#include "MDigProcess.h"
#include "MDigProcessDlg.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#endif

//////////////////////
// ActiveMIL constants
//
// Standard display width
//
const int  DEFAULT_DISPLAY_WIDTH = 480;

#define pi 3.1416
```

```
const int BUFFERING_SIZE_MAX = 20;
const char EXAMPLE_AVI[] = "metaltransfer.avi";
long NbParticles;
const char EXAMPLE_IMAGE2[] = "Chip.mim";

// CAboutDlg dialog used for App About

class CAboutDlg : public CDialog
{
public:
    CAboutDlg();

// Dialog Data
    enum { IDD = IDD_ABOUTBOX };

    protected:
    virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV
support

// Implementation
protected:
    DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
END_MESSAGE_MAP()


// CMDigProcessDlg dialog



CMDigProcessDlg::CMDigProcessDlg(CWnd* pParent /*=NULL*/)
```

```
    : CDialog(CMDigProcessDlg::IDD, pParent)
    , m_CommentText("")
    , m_StepNumber(1)
    , m_MaxBufferingCount(0)
    , m_ProcessingInProgress(FALSE)
{
    m_ProcessImages.SetSize(BUFFERING_SIZE_MAX);
    m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}

CMDigProcessDlg::~CMDigProcessDlg()
{
    for(int i = 0; i < BUFFERING_SIZE_MAX; i++)
         {
        m_ProcessImages[i] = NULL;
        }
}
void CMDigProcessDlg::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);

    DDX_Control(pDX, IDC_APPLICATION1, m_Application1);
    DDX_Control(pDX, IDC_SYSTEM1, m_System1);
    DDX_Control(pDX, IDC_IMAGE1, m_Image1);
    DDX_Control(pDX, IDC_DISPLAY1, m_Display1);
    DDX_Control(pDX, IDC_GRAPHICCONTEXT1, m_GraphicContext1);
    DDX_Control(pDX, IDC_DIGITIZER1, m_Digitizer1);
    DDX_Control(pDX, IDC_IMAGEPROCESSING1, m_ImageProcessing1);
    DDX_Text(pDX, IDC_COMMENTTEXT, m_CommentText);
      DDX_Control(pDX, IDC_EDGEFINDER1, m_EdgeFinder1);
      //DDX_Control(pDX, IDC_LIST1, m_listResults);

}

BEGIN_MESSAGE_MAP(CMDigProcessDlg, CDialog)
    ON_WM_SYSCOMMAND()
    ON_WM_PAINT()
    ON_WM_QUERYDRAGICON()
    //}}AFX_MSG_MAP
    ON_BN_CLICKED(IDC_NEXTBUTTON, OnBnClickedNextbutton)
    ON_WM_CLOSE()
```

```
END_MESSAGE_MAP()

BEGIN_EVENTSINK_MAP(CMDigProcessDlg, CDialog)
    ON_EVENT(CMDigProcessDlg, IDC_DIGITIZER1, 16, 
ProcessModifiedImageDigitizer1, VTS_I4)
END_EVENTSINK_MAP()

// CMDigProcessDlg message handlers

BOOL CMDigProcessDlg::OnInitDialog()
{
    CDialog::OnInitDialog();

    // Add "About..." menu item to system menu.

    // IDM_ABOUTBOX must be in the system command range.
    ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
    ASSERT(IDM_ABOUTBOX < 0xF000);

    CMenu* pSysMenu = GetSystemMenu(FALSE);
    if (pSysMenu != NULL)
    {
        CString strAboutMenu;
        strAboutMenu.LoadString(IDS_ABOUTBOX);
        if (!strAboutMenu.IsEmpty())
        {
            pSysMenu->AppendMenu(MF_SEPARATOR);
            pSysMenu->AppendMenu(MF_STRING, IDM_ABOUTBOX, strAboutMenu);
        }
    }

    // Set the icon for this dialog.  The framework does this 
automatically
    //  when the application's main window is not a dialog
    SetIcon(m_hIcon, TRUE);         // Set big icon
    SetIcon(m_hIcon, FALSE);        // Set small icon

    /////////////////////////////////////
    // ActiveMIL example initialization
    //
    // Resize the dialog
    //
    AdjustDialog();

    m_Digitizer1->GrabContinuous();

    // Update the coment text
    //
    m_CommentText = "Click <Next> or press <Enter> to start multiple
buffering processing.\n";
    UpdateData(FALSE);

    return TRUE;  // return TRUE  unless you set the focus to a control
}

void CMDigProcessDlg::OnSysCommand(UINT nID, LPARAM lParam)
{
    if ((nID & 0xFFF0) == IDM_ABOUTBOX)
    {
        CAboutDlg dlgAbout;
        dlgAbout.DoModal();
    }
    else
    {
        CDialog::OnSysCommand(nID, lParam);
    }
}

// If you add a minimize button to your dialog, you will need the code
below
//  to draw the icon.  For MFC applications using the document/view
model,
//  this is automatically done for you by the framework.

void CMDigProcessDlg::OnPaint()
{
    if (IsIconic())
    {
        CPaintDC dc(this); // device context for painting

        SendMessage(WM_ICONERASEBKGND,
reinterpret_cast<WPARAM>(dc.GetSafeHdc()), 0);
```

```
        // Center icon in client rectangle
        int cxIcon = GetSystemMetrics(SM_CXICON);
        int cyIcon = GetSystemMetrics(SM_CYICON);
        CRect rect;
        GetClientRect(&rect);
        int x = (rect.Width() - cxIcon + 1) / 2;
        int y = (rect.Height() - cyIcon + 1) / 2;

        // Draw the icon
        dc.DrawIcon(x, y, m_hIcon);
    }
    else
    {
        CDialog::OnPaint();
    }
}

// The system calls this function to obtain the cursor to display while
the user drags
//  the minimized window.
HCURSOR CMDigProcessDlg::OnQueryDragIcon()
{
    return static_cast<HCURSOR>(m_hIcon);
}

////////////////////////////////////////////////////////////////////////
/////
// ActiveMIL procedure: In response of clicking the "Next" button.
//                      Drives the example through each step,
//                      performing appropriate actions
//
void CMDigProcessDlg::OnBnClickedNextbutton()
{
    // Increment the step counter and updates text in the dialog
    //
    AdjustStepNumber();

    switch(m_StepNumber)
        {
        case 2:
            {
            // Stop continuous grab
            //
            m_Digitizer1->Halt();

            BOOL ContinueAllocating = TRUE;
            while(ContinueAllocating)
                {
                // Create an image control
                //
                IImagePtr NewImage = m_Application1-
>CreateObject("MIL.Image", FALSE);

                if(NewImage != NULL)
                    {
                    // Set its properties
                    //
                    NewImage->CanGrab = TRUE;
                    NewImage->CanDisplay = FALSE;
                    NewImage->CanProcess = TRUE;
                    NewImage->NumberOfBands = 1;
                    NewImage->SizeX = (long)(m_Digitizer1->SizeX *
m_Digitizer1->ScaleX);
                    NewImage->SizeY = (long)(m_Digitizer1->SizeY *
m_Digitizer1->ScaleY);

                    try
                        {
                        // try allocating the image
                        //
                        NewImage->Allocate();

                        // Add the image to the array of images
                        //
                        m_ProcessImages[m_MaxBufferingCount] = NewImage;

                        // Increment the maximum of procssing images
                        //
                        m_MaxBufferingCount++;

                        if(BUFFERING_SIZE_MAX == m_MaxBufferingCount)
```

```
                                {
                                ContinueAllocating = FALSE;
                                }
                            }
                        catch(COleDispatchException *e)
                            {
                            // An exception was thrown while allocating the
image
                            // Stop allocating new images
                            //
                            ContinueAllocating = FALSE;

                            // Always delete caught exceptions
                            //
                            e->Delete();
                            }
                        // Update the coment text while allocating images
                        //
                        m_CommentText = "Allocating processing images...\n";
                        UpdateData(FALSE);
                        }
                    }

            // Build a CActiveMILObjectsArray
            //
            CActiveMILObjectsArray<IImagePtr>
ActiveMILObjectsArray(m_ProcessImages.GetData(),
m_ProcessImages.GetSize());

            // Start the processing.  The processing event is fired for
every frame grabbed
            //
            m_ProcessingInProgress = TRUE;
            m_Digitizer1->MultipleBuffering->Process(digStart,
ActiveMILObjectsArray);

                  // Update the comment text
                  // changed by zz
                  m_CommentText = "Processing in progress...\n"
                          "Click <Next> or press <Enter> to stop.";
            UpdateData(FALSE);
            }
            break;
        case 3:
            {
            // Stop the procesing
            //
            m_Digitizer1->MultipleBuffering->Halt(FALSE);
            m_ProcessingInProgress = FALSE;

            // Update the comment text
            //
            m_CommentText.Format("%ld frames grabbed at %.1f frames/sec
(%.1f ms/frame).\n"
                                 "Click <Next> or press <Enter> to
end.",
                                 m_Digitizer1->MultipleBuffering-
>NumberOfFrameProcessed,
                                 m_Digitizer1->MultipleBuffering-
>FrameRate,
                                 1000.0/m_Digitizer1->MultipleBuffering-
>FrameRate);
            UpdateData(FALSE);
            }
            break;
        default :
            // Last Step: End the application
            //
            // Close dialog application
            //
            EndDialog(IDOK);
            break;
        }
}

//////////////////////////////////////////////////////////////////////////
/////
// ActiveMIL procedure: Digitizer.ProcessModifiedImage Event Handler.
//                      Do the processing on the grabbed image.
//
```

```
void CMDigProcessDlg::ProcessModifiedImageDigitizer1(long ImageIndex)
{
    CString strFrameCount;
    strFrameCount.Format("%d", m_Digitizer1->MultipleBuffering-
>NumberOfFrameProcessed);

    // Draw the frame count in the image
    //
    m_GraphicContext1->Image = m_ProcessImages[ImageIndex];
    m_GraphicContext1->Text(static_cast<bstr_t>(strFrameCount));

    m_ImageProcessing1->Source1 =
_variant_t(m_ProcessImages[ImageIndex], true);
                  // perform the edge detection
                  m_ImageProcessing1-
>EdgeDetect(FALSE,FALSE,0,impOverscanDisable);
                  m_ImageProcessing1->Binarize(impGreaterOrEqualTo,
128+20, 0);

            // Label the image
            //
            m_ImageProcessing1->Label(imp8Connected, impGrayscale);

            // The largest label value corresponds to the number of
particles in the image
            //
            m_ImageProcessing1->FindExtremes(FALSE, TRUE);
            VARIANT varMaxLabelNumber = m_ImageProcessing1->Results-
>Item(1);
                  long NbParticles = (long) V_R4(&varMaxLabelNumber);
                  double px=0;
                  double py=0;
                  double area=0;
                  double angc=pi/4;

                  //for (int i=1; i<=NbParticles; i++)
                  //{
                  int i=1;
                  double m=0;
```

```
                        m_ImageProcessing1-
>CalculateStats(impNumberOfPixels,impInRange,i,i);
                      m = m_ImageProcessing1->Results->Stats-
>GetNumberOfPixels();
                        if (m>10)
                        {
                   m_ImageProcessing1->Free();
                   m_ImageProcessing1->ResultType = impAnyResultType;
                   //m_ImageProcessing1->ResultType = impExtremeList;
                   m_ImageProcessing1->ResultSize = 200;
                   m_ImageProcessing1->Allocate();

                  m_ImageProcessing1->LocateEvents(impInRange,i,i);

                  double sumx=0;
                  double sumy=0;
                  double isize=0;
                  if (m<=200)
                        isize = m;
                  else
                        isize=200;
                  double xarray[200];
                  double yarray[200];
                  double sumarray[200];
                  double minx = 1000;
                  double miny = 1000;
                  double maxx = 0;
                  double maxy = 0;

                  for (int j=1;j<=isize;j++)
                  {
            ILocateEventsResultPtr ResultItem =
            (ILocateEventsResultPtr)(V_DISPATCH(&m_ImageProcessing1-
>Results->Item(j)));

                  double PositionX =ResultItem->GetPositionX();
                  double PositionY =ResultItem->GetPositionY();
                  xarray[j] = PositionX;
                  yarray[j] = PositionY;
                  sumx+=PositionX;
                  sumy+=PositionY;
```

```
if (minx > PositionX)
      minx = PositionX;
if (miny > PositionY)
      miny = PositionY;
if (maxx < PositionX)
      maxx = PositionX;
if (maxy < PositionY)
      maxy = PositionY;
}

sumx/=isize;
sumy/=isize;

double xdis=maxx-minx;

Matrix xo(isize, 1);
Matrix yo(isize,1);
for (int i=1; i<=isize; i++)
{
      xo(i,1)=(double) (xarray[i]-sumx);
      yo(i,1)=(double) (yarray[i]-sumy);
}

Matrix ang(isize,1);
for (int j=1; j<=isize; j++)
{
      ang(j,1) = (double) atan(yo(j,1)/xo(j,1));
}

Matrix th(isize, 3);
for (int k=1; k<=isize; k++)
{
      th(k,1) = (double) 1;
      th(k,2) = (double) ang(k, 1);
      th(k,3) = (double) (ang(k,1)*ang(k,1));
}

Matrix dradius(isize,1);
for (int h=1; h<=isize; h++)
{
```

```
                              dradius(h,1) = (double)
sqrt(xo(h,1)*xo(h,1)+yo(h,1)*yo(h,1));
                        }

                        Matrix inter1 = th.t();
                        Matrix inter2 = inter1*th;
                        Matrix inter3 = inter2.i();
                        Matrix inter4 = inter3*inter1;

                        Matrix amatrix = inter4*dradius;

                        double dsize=0;
                        double a1 = (double) amatrix(1,1);
                        double a2 = (double) amatrix(2,1);
                        double a3 = (double) amatrix(3,1);
                        dsize =
a1*a1*pi+(1/3)*a2*a2*(pi*pi*pi)+(1/5)*a3*a3*pi*pi*pi*pi*pi+(2/3)*a1*a3*p
i*pi*pi;

                        Matrix sortedang(isize,1);
                        for (int i=1; i<=isize; i++)
                        {
                              if (xo(i,1)>0 && yo(i,1)>0)
                              {
                                    sortedang(i,1)=(double)
(atan(yo(i,1)/xo(i,1)));
                              }
                              if (xo(i,1)>0 && yo(i,1)<0)
                              {
                                    sortedang(i,1)=(double)
(2*pi+atan(yo(i,1)/xo(i,1)));
                              }
                              if (xo(i,1)<0 && yo(i,1)>0)
                              {
                                    sortedang(i,1)=(double)
(pi+atan(yo(i,1)/xo(i,1)));
                              }
                              if (xo(i,1)<0 && yo(i,1)<0)
                              {
                                    sortedang(i,1)=(double) (3*pi/2-
atan(yo(i,1)/xo(i,1)));
```

```
                        }
                }
                sort_ascending(sortedang);
                Matrix diff(isize,1);
                for (int i=1; i<=(isize-1);i++)
                {
                        diff(i,1)=sortedang(i+1,1)-sortedang(i,1);
                }
                diff(isize,1)=2*pi-sortedang(isize,1)+sortedang(1,1);

                double maxang = diff.maximum();

                if (maxang < angc)
                {
                        px=sumx;
                        py=sumy;
                        area=dsize;
                        angc=maxang;
                }
                }
}

void CMDigProcessDlg::imageprocessing()
{
        m_ImageProcessing1->Binarize(impGreaterOrEqualTo, 208, 0);
        m_ImageProcessing1->Label(imp8Connected, impGrayscale);
}

////////////////////////////////////////////////////////////////////////
/////
// ActiveMIL procedure: Stop the MultipleBuffering when the application
is
//                      closed
//
void CMDigProcessDlg::OnClose()
{
     if(m_ProcessingInProgress)
          {
          // Stop the procesing
          //
          m_Digitizer1->MultipleBuffering->Halt(FALSE);
        m_ProcessingInProgress = FALSE;
        }

    CDialog::OnClose();
}

////////////////////////////////////////////////////////////////////////
/////
// ActiveMIL procedure: Increments the step counter and updates text in
//                      the dialog according to the new value
//
void CMDigProcessDlg::AdjustStepNumber()
{
    // Increment the step count
    //
    m_StepNumber++;

    // Write the step count in the comment frame
    //
    CString str;
    str.Format("Step %d:", m_StepNumber);
    CWnd *pStepNumber = GetDlgItem(IDC_STEPNUMBER);
    pStepNumber->SetWindowText(str);
}

////////////////////////////////////////////////////////////////////////
///////
// ActiveMIL procedure :    Calculate how to resize the dialog in order
to
//                          view all element in all resolutions and font
sizes
//
void CMDigProcessDlg::AdjustDialog()
{
    // Set the icon for this dialog.  The framework does this
automatically
    // when the application's main window is not a dialog
    //
    HICON hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
    SetIcon(hIcon, TRUE);            // Set big icon
    SetIcon(hIcon, FALSE);           // Set small icon
    // Retrieve the relevant dialog controls
    //
    CWnd *pNextButton = GetDlgItem(IDC_NEXTBUTTON);
    CWnd *pStepNumber = GetDlgItem(IDC_COMMENTTEXT);

    int MimImageSizeX = (int)(m_Digitizer1->SizeX * m_Digitizer1-
>ScaleX);
    int MimImageSizeY = (int)(m_Digitizer1->SizeY * m_Digitizer1-
>ScaleY);
    long nMaxDisplayWidth;
    long nMaxNextStepWidth;
    CRect rectNextButton;
    CRect rectCommentFrame;
    CRect rectDisplay;
    CRect rectDialog;
      CRect rectList;

    // Adjust the size of the Display
    //
    m_Display1.SetWindowPos(NULL,                               //
pWndInsertAfter
                            0,                                  // x
                            0,                                  // y
                            MimImageSizeX,                      // cx
                            MimImageSizeY,                      // cy
                            SWP_NOMOVE | SWP_NOOWNERZORDER);    // flags

    // Resize the image to the size of the digitizer
    //
    if(m_Image1->IsAllocated)
        {
        m_Image1->Free();
        }

    m_Image1->SizeX = (long)(m_Digitizer1->SizeX * m_Digitizer1-
>ScaleX);
    m_Image1->SizeY = (long)(m_Digitizer1->SizeY * m_Digitizer1-
>ScaleY);
    m_Image1->NumberOfBands = m_Digitizer1->NumberOfBands;
    m_Image1->Allocate();

    // Get the sizes of relevant elements and calculate how to resize
the
    // dialog in order to view all elements, in all resolutions and font
sizes
    //
    pNextButton->GetWindowRect(&rectNextButton);
    pStepNumber->GetWindowRect(&rectCommentFrame);
    m_Display1.GetWindowRect(&rectDisplay);
    GetWindowRect(&rectDialog);

    // Check the width of the display, if it is big enough, we will
    // adjust the comment text frame and the next button
    // to fit the width of the diplay
    //
    if(rectDisplay.Width() > DEFAULT_DISPLAY_WIDTH)
        {
        // Calculate the size of the window displacements
        //
        int SizeChange = rectDisplay.right - rectNextButton.right;

        // Change the sizes of the Next button and the comment frame
        //
        rectNextButton.OffsetRect(SizeChange, 0);
        rectCommentFrame.InflateRect(0, 0, SizeChange, 0);

        // Move the windows
        //
        ScreenToClient(&rectNextButton);
        ScreenToClient(&rectCommentFrame);

        pNextButton->MoveWindow(&rectNextButton, FALSE);
        pStepNumber->MoveWindow(&rectCommentFrame, FALSE);

        // Reconvert the rectangles to screen coordinates
        //
        ClientToScreen(&rectCommentFrame);
        ClientToScreen(&rectNextButton);
        }
    // We can resize the dialog
    //
    nMaxDisplayWidth  = rectDisplay.Width() +
                        2 * abs(rectDialog.left - rectDisplay.left);

    nMaxNextStepWidth = rectNextButton.left - rectDialog.left +
                        rectNextButton.Width() +
                        rectCommentFrame.left - rectDialog.left;

    if (nMaxNextStepWidth > nMaxDisplayWidth)
        {
        SetWindowPos(NULL,                                      //
pWndInsertAfter
                     0,                                         // x
                     0,                                         // y
                     nMaxNextStepWidth,                         // cx
                     rectDisplay.Height() +
                     rectDisplay.top - rectDialog.top +
                     rectDisplay.left - rectDialog.left,        // cy
                     SWP_SHOWWINDOW|SWP_NOMOVE);                // flags
        }
    else
        {
        SetWindowPos(NULL,                                      //
pWndInsertAfter
                     0,                                         // x
                     0,                                         // y
                     nMaxDisplayWidth,                          // cx
                     rectDisplay.Height() +
                     rectDisplay.top - rectDialog.top +
                     abs (rectDisplay.left - rectDialog.left),  // cy
                     SWP_SHOWWINDOW|SWP_NOMOVE);                // flags
        }
}
```

Appendix D

Derivation of Kalman Gain

1. **Deriving the posterior estimate covariance matrix**

Starting with our invariant on the error covariance $P_{k|k}$

$$P_{k|k} = \operatorname{cov}(x_k - \hat{x}_{k|k}) \tag{1}$$

Substitute in the definition of $\hat{x}_{k|k}$, the equation becomes:

$$P_{k|k} = \operatorname{cov}(x_k - \hat{x}_{k|k-1} - K_k \tilde{y}_k) \tag{2}$$

and substitute $\tilde{y}_k$, the equation becomes:

$$P_{k|k} = \operatorname{cov}(x_k - \hat{x}_{k|k-1} - K_k (z_k - H_k \hat{x}_{k|k-1})) \tag{3}$$

And substitute z_k, the equation becomes:

$$P_{k|k} = \operatorname{cov}(x_k - \hat{x}_{k|k-1} - K_k (H_k x_k + v_k - H_k \hat{x}_{k|k-1})) \tag{4}$$

The above equation can be reduced to:

$$P_{k|k} = \operatorname{cov}((I - K_k H_k)(x_k - \hat{x}_{k|k-1}) - K_k v_k) \tag{5}$$

Since the measurement error v_k is uncorrelated with the other terms, this becomes

$$P_{k|k} = \operatorname{cov}((I - K_k H_k)(x_k - \hat{x}_{k|k-1})) + \operatorname{cov}(K_k v_k) \tag{6}$$

By the properties of vector covariance this becomes:

$$P_{k|k} = (I - K_k H_k) \operatorname{cov}(x_k - \hat{x}_{k|k-1})(I - K_k H_k)^T + K_k \operatorname{cov}(v_k) K_k{}^T \tag{7}$$

Which, using our invariant on $P_{k|k-1}$ and the definition of R_k, the equation becomes:

2. Kalman Gain Derivation

The Kalman filter is a minimum mean-square error estimator. The error in the posterior estimation is

$x_k - \hat{x}_{k|k}$

We seek to minimize the expected value of the square of the magnitude of this vector, $E[|x_k - \hat{x}_{k|k}|^2]$. This is equivalent to minimizing the trace of the posterior estimate covariance matrix $P_{k|k}$. By expanding out the terms in the equation above and collecting, we get:

$$P_{k|k} = (I - K_k H_k) P_{k|k-1} (I - K_k H_k)^T + K_k R_k K_k{}^T \tag{8}$$

$$\begin{aligned} P_{k|k} &= P_{k|k-1} - K_k H_k P_{k|k-1} - P_{k|k-1} H_k{}^T K_k{}^T + K_k (H_k P_{k|k-1} H_k{}^T + R_k) K_k{}^T \\ &= P_{k|k-1} - K_k H_k P_{k|k-1} - P_{k|k-1} H_k{}^T K_k{}^T + K_k S_k K_k{}^T \end{aligned} \tag{9}$$

The trace is minimized when the matrix derivative is zero:

$$\frac{\partial tr(P_{k|k})}{\partial K_k} = -2(H_k P_{k|k-1})^T + 2 K_k S_k = 0 \tag{10}$$

Solving this for K_k yields the Kalman gain:

$$K_k S_k = (H_k P_{k|k-1})^T = P_{k|k-1} H_k{}^T \tag{11}$$

$$K_k = P_{k|k-1} H_k{}^T S_k{}^{-1} \tag{12}$$

References

[1] Saeed G. and Zhang Y. M.: "Mathematical formulation and simulation of specular reflection based measurement system for gas tungsten arc weld pool surface," Measurement Science and Technology, 14(8): pp.1671-1682, 2003

[2] Kim JS, Son YT, Cho HS, et al.: "A robust visual seam tracking system for robotic arc welding," Mechatronics, 6 (2): pp.141-163, 1996

[3] Tsai CH, Hou KH and Chuang HT: "Fuzzy control of pulsed GTA welds by using real-time root bead image feedback," Journal of Materials Processing Technology 176 (1-3): pp.158-167, 2006

[4] Zhang GJ, Yan ZH and Wu L: "Reconstructing a three-dimensional P-GMAW weld pool shape from a two-dimensional visual image," Measurement Science & Technology 17 (7): pp.877-1882 July, 2006

[5] Zhang, Y.M. and P.J. Li:"Modified active control of metal transfer and pulsed GMAW of titanium," Welding Journal 80(2): pp. 54S-61S, 2001

[6] Amin, M.: "Pulse current parameters for arc stability and controlled metal transfer in arc welding," Metal Construction, 15: pp. 272-278, 1983

[7] Jones, L.A., Eagar T.W., and Lang J.H.: "A dynamic model of drops detaching from a gas metal arc welding electrode," Journal of Physics D-Applied Physics, 31(1): pp. 107-1231998

[8] Wang, G., Huang P.G., and Zhang Y.M.: "Numerical analysis of metal transfer in gas metal arc welding under modified pulsed current conditions," Metallurgical and Materials Transactions B-Process Metallurgy and Materials Processing Science, 35(5): pp.

857-866, 2004

[9] Chakraborty, S.: "Analytical investigations on breakup of viscous liquid droplets on surface tension modulation during welding metal transfer,"Applied Physics Letters, 86(17): pp. Art. No. 174104 APR 25, 2005

[10] Wu C.S., Chen M.A., and Lu Y.F.: "Effect of current waveforms on metal transfer in pulsed gas metal arc welding," Measurement Science & Technology, 16(12): pp. 2459-2465, 2005

[11] Jones, L.A., Eagar T.W., and Lang J.H.: "Investigation of drop detachment control in gas metal arc welding," in Proceedings of the 3rd International Conference on Trends in Welding Research, 1992

[12] Yudodibroto BYB, Hermans MJM, Hirata Y, et al.: "Pendant droplet oscillation during GMAW," Science and Technology of Welding and Joining, 11 (3): pp.308-314, May, 2006

[13] Rhee S. and Kannatey-asibue: "Observation of metal transfer during gas metal arc-welding, "welding journal 71 (10): S381-S386, October, 1992

[14] Zhang, Y.M., Liguo E, and Kovacevic, R: “Active metal transfer control by monitoring excited droplet oscillation,” Welding Journal, 77(9): 388s-395s, 1998

[15] Li, K., Chen, J.S. and Zhang, Y.M.:"Double-electrode GMAW process and control," accepted for publication, Welding Journal

[16] Kovacevic, R. and Zhang, Y. M.: "Real-time image processing for monitoring of free weld pool surface," ASME Journal of Manufacturing Science and Engineering, 119(2): pp.161-169, 1997

[17] Wren C., Azarbayejani A., Darrell T. and Pentland A.: "Pfinder: Real-Time

Tracking of the Human Body," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 19, No.7, July, 1997

[18] Beymer D. and Konolige K.: "Real-Time Tracking of Multiple People Using Continuous Detection," Artificial Intelligence Center, SRI International Report, 1998

[19] Azarbayejani A. and Pentland A.: "Real-Time Self-Calibrating Stereo Person Tracking Using 3D Shape Estimation from Blob Features," Proc. of the ICPR, IEEE, 1996

[20] Wang, D. L. and Terman, D.: "Image segmentation based on oscillatory correlation," In Proc. of World Congress on Neural Net, pp. 521-525, 1995

[21] Savvides M., Vijaya Kumar B.V.K. and Khosla P.: "Face Verification Using Correlation Filters," Proc. Third IEEE Automatic Identification Advanced Technologies, pp. 56-61, 2002

[22] Qian, R.J., and Huang, T.S.: "Optimal Edge Detection in Two-Dimensional Images," IEEE Trans. Image Processing, vol.5, no. 7, pp.1215-1220, 1996

[23] Marr, D., and Hildreth, E.: "Theory of Edge Detection," Proc.R.Soc. Lond., vol. B207, pp.187-217, 1980

[24] Singer, A. C., Kozat, S. S. and Feder, M.: " Universal linear least squares prediction: upper and lower bounds, " *IT.*, vol. 48, pp. 2354-2362, AUG 2002

[25] Gruen A.: "Adaptive least squares correlation: a powerful image matching technique." South African Journal of Photogrammetry, Remote Sensing and Cartography, 14(3), pp. 175-187, 1985

[26] D'Apuzzo N., Plänkers R. and Fua P.: "Least Squares Matching Tracking Algorithm for Human Body Modeling, " International Archives of Photogrammetry and

Remote Sensing, Vol. 33, Part B5/1, Amsterdam, The Netherlands, pp.164-171, 2000

[27] Gavrila D. M. and Davis. L.: "3-D model-based tracking of humans in action: a multi-view approach." Proc. Conference on Computer Vision and Pattern Recognition, San Francisco, CA, USA, pp. 73-80, 1996

[28] Mitchell H. and Houtekamer P.: "An adaptive ensemble Kalman filter." Monthly Weather Review, 126:416–433, 1998

[29] Manfredi V., Mahadevan S. and Kurose J.: "Kalman filters for prediction and tracking in an adaptive sensor network." *Technical report, U. of Massachusetts Amherst*, 2005.

[30] Murphy K.: "Switching Kalman filters." Technical report, U. C. Berkeley, 1998

[31] Houtekamer P. and Mitchell H.: "Data assimilation using an ensemble Kalman filter technique." *Monthly Weather Review*, 126:796–811, 1998

[32] Evensen G.: "The ensemble Kalman filter: Theoretical formulation and practical implementation." *Ocean Dynamics*, 53:343–367, 2003

[33] Ott E., Hunt B., Szunyogh I., Zimin A., Kostelich E., Corazza M., Kalnay E., Patil D., and Yorke J.: "A local ensemble Kalman filter for atmospheric data assimilation." *Tellus*, 2004

[34] Ristic B., Arulampalam S., and Gordon N.:" *Beyond the Kalman Filter.* " *Particle Filters for Tracking Applications*. Artech House, Boston, Massachusetts, 2004

[35] Coltuc D., Bolon Ph.: "An Inverse Problem: Histogram Equalization," in Signal Processing IX, Theories and Applications, EUSIPCO'98 11**:** pp. 861-864, 1998

[36] Coltuc D., Bolon Ph.: "Watermarking by histogram specification," in SPIE Proceedings, vol. 3657, Conf. on Security and Watermarking of Multimedia Contents, USA, Jan. 1999.

[37] Rafael C. and Richard E.:"Digital Image Processing," Second Edition, 2002

[38] Nadadur D.and Haralick R.M.: "Recursive Binary Dilation and Erosion Using Digital Line Structuring Elements in Arbitrary Orientations," IEEE Trans. Image Processing, vol. 9, no.5, May, 2000

[39] Gil J.Y. and Kimmel R.: "Efficient dilation, erosion, opening and closing algorithms," IEEE Transactions on Pattern Recognition and Machine Intelligence, 24(12):1606–1617, December, 2002

[40] Razaz M. and Hagyard D.M.P.: "Morphological Structuring Element Decomposition: Implementation and Comparison," Signal Processing VIII; Theories & Applications, Vol. 1 , pp. 288-291, 1996

[41] Razaz M. and Hagyard D.M.P.: "Structuring element decomposition by tree searching," Proc. IEEE Nonlinear Signal. & Image Processing, 1997

[42] DeFatta D. J., Lucas J.G., Hodgkiss W.S.: "Digital Signal Processing: A System Design Approach," pp 305-315, Wiley, 1988.

[43] Woods, R.E., and Gonzalez, R.C.: "Real-Time Digital Image Enhancement," proc. IEEE, vol.69, no.5, pp.643-654, 1981

[44] Prewitt, J.M.S.: "Object Enhancement and Extraction," In Picture Processing and Psychopictorics, Lipkin, B.S., and Rosenfeld, A (eds.), Academic Press, New York, 1970

[45] Narendra, P.M. and Fitch, R.C.: "Real-Time Adaptive Contrast Enhancement," IEEE Trans. Pattern Anal. Mach. Intell., vol. PAMI-3, no. 6, pp.655-661, 1981

[46] Highnam, R., and Brady, M.: "Model-Based Image Enhancement of Far Infrared Images," IEEE Trans. Pattern Anal. Machine Intell., vol. 19, no. 4, pp.410-415, 1997

[47] Gonzalez, R.C., and Fittes, B.A.: "Gray-Level Transformations for Interactive Image Enhancement," Mechanism and Machine Theory, vol. 12, pp. 111-122, 1977

[48] Centeno, J.A.S., and Haertel, V.: "An Adaptive Image Enhancement Algorithm," Pattern Recog., vol.30, no.7 pp.1183-1189, 1997

[49] Zhu, H., Chan F.H.Y., and Lam, F.K.: "Image Contrast Enhancement by Constrained Local Histogram Equalization," Computer Vision and Image Understanding, vol.73, no.2, pp.281-290, 1999

[50] Stark, J.A.: "Adaptive Image Contrast Enhancement Using Generalizations of Histogram Equalization," IEEE Trans. Image Processing, vol.9, no.5 pp.889-896, 2000

[51] Shafarenko, L., Petrou, M., and Kittler, J.: "Histogram-Based Segmentation in a Perceptually Uniform Color Space," IEEE. Trans. Image Processing, vol.7, no.9, pp.1354-1358, 1998

[52] Hummel, R.A.: "Histogram Modification Techniques," Technical Report TR-329. F-44620-72C-0062,Computer Science Center, University of Maryland, College Park, Md, 1974

[53] Piech, M.A.: "Decomposing the Laplacian," IEEE Trans. Pattern Anal. Machine Intell., vol. 12, no. 8, pp. 830-831, 1990

[54] Kim, J.K., Park, J.M., Song, K.S., and Park, H. W.: "Adaptive Mammographic Image Enhancement Using First Derivative and Local Statistics," IEEE Trans. Medical Imaging, vol.16, no.5, pp.495-502, 1997

[55] Gunn, S.R.: "Edge Detection Error in the Discrete Laplacian of a Gaussian," Proc. 1998 Int'l Conference on Image Processing, vol. 2, pp. 515-519, 1998

[56] Gunn, S.R.: "On the discrete Representation of the Laplacian of a Gaussian," Pattern Recog., vol. 32, no. 8, pp.1463-1472, 1999

[57] Cannon, T.M.: "Digital Image De-blurring by Non-linear Homomorphic Filtering," Ph.D. thesis, University of Utah, 1974

[58] Marion B, Rummel S, Anderberg A.: "Current-voltage curve translation by bilinear interpolation," Progress in Photovoltaics: Research and Applications, Volume: 12, pp.1–16, 2004

[59] H. Knutsson and C-F. Westin.: "Normalized and differential convolution: Methods for interpolation and filtering of incomplete and uncertain data," In Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 515–523, June, 1993

[60] Luo F., Wu S.J., Jiao L.C. and Zhang L.R.: "Implementation of de-noise DWT chip based on adaptivesoft-threshold," International conference, Signal Processing Proceedings. Vol.1, pp. 614-618, 2000

[61] Tommiska M., Loukola M. and Koskivirta T.: "An FPGA-based simulation and implementation of AAL type 2 receiver," Journal of Communications and Networks, pp. 63-67, 1999

[62] MacQueen, S.: "Some methods for classification and analysis of multivariate observations," Proc. Fifth Berkeley Symp. On Mathematical Statistics and Probability, vol. 1, pp. 281–297, 1967

[63] Selim, S.Z., and Alsulta, K.S.: "A simulated annealing algorithm for the clustering problem," Pattern Recognit., 24, (10), pp. 1003–1008, 1991

[64] Girolami, M.: "Mercer kernel-based clustering in feature space," IEEE Trans. Neural Netw., 13, (3), pp. 780–784, 2002

[65] Ben-Hur, A., Horn, D., Siegelmann, H.T., and Vapnik, V.N.: "Support vector clustering," J. Mach. Learn. Res., 2, pp. 125–137, 2001

[66] Horn, D.: "Clustering via Hilbert space," Physica A, 302, pp. 70–79, 2001

[67] Chiang, J.H., and Hao, P.Y.: "A new kernel-based fuzzy clustering approach: support vector clustering with cell growing," IEEE Trans. Fuzzy Syst., 11, (4), pp. 518–527, 2003

[68] Camastra, F., and Verri, A.: "A novel kernel method for clustering," IEEE Trans. Pattern Anal. Mach. Intell., 27, (5), pp. 801–805, 2005

[69] Duda, R.O., Hart, P.E., and Stork, D.G.: "Pattern classification," John Wiley and Sons, Inc., NY, 2nd edition, 2001

[70] Gander W., Golub G.H., and Strebel R.: "Least-Square Fitting of Circles and Ellipses," BIT, no. 43, pp. 558-578, 1994

[71] Rosin P.L.: "A Note on the Least Squares Fitting of Ellipses," Pattern Recognition Letters, no. 14, pp. 799-808, Oct. 1993

[72] Bookstein F.L.: "Fitting Conic Sections to Scattered Data," Computer Graphics and Image Processing, no. 9, pp. 56-71, 1979

[73] Fitzgibbon A.W.: "Stable Segmentation of 2D Curves," PhD thesis, Dept. of Artificial Intelligence, Univ. of Edinburgh, 1998

[74] Fitzgibbon A.W. and Fisher R.B.: "A Buyer's Guide to Conic Fitting, " Proc. British Machine Vision Conf, Birmingham, England, 1995

[75] Kanatani K.: "Statistical Bias of Conic Fitting and Renormalization," IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 16, no. 3, pp. 320-326, 1994

[76] Sampson P.D.: "Fitting Conic Sections to Very Scattered Data: An Iterative Refinement of the Bookstein Algorithm," Computer Graphics and Image Processing, no. 18, pp. 97-108, 1982

[77] Nave, C.R.: "http://hyperphysics.phy-astr.gsu.edu/hbase/forces/isq.html,"[on-line], Georgia State University, 2005.

[78] Wiki: "http://en.wikipedia.org/wiki/lambertian_reflectance".

[79] Wiki: "http://en.wikipedia.org/wiki/planck's_law_of_black_body_radiation".

[80] Jain, A.K.: "Fundamentals of Digital Image Processing," Prentice Hall, 1989

[81] Pratt, W.K.: "Digital Image Processing," 3rd Edition, John Wiley, 2001.

[82] Sezgin, M. and Sankur, B., "Survey over image thresholding techniques and quantitative performance evaluation," *Journal of Electronic Imaging,* vol.13, pp. 146-165, Jan, 2004

[83] Sieracki, M.E., Reichenbach, S.E. and Webb K.L., "Evaluation of automated threshold selection methods for accurately sizing microscopic fluorescent cells by image analysis," *Appl. Environ.Microbiol.* vol.55, pp.2762-2772, 1989

[84] Rosenfeld, A. and Torre, P.D., "Histogram concavity analysis as an aid in threshold selection," *IEEE Trans. Syst. Man Cybern.* SMC-9, pp. 38-52, 1983

[85] Ramesh, N., Yoo, J.H. and Sethi, I.K., "Thresholding based on histogram approximation," *IEE Proc-Vis. Image Signal Process.,* Vol. 142, No. 5, pp.271-279, 1995

[86] Otsu, N., "A threshold selection method from gray level histogram," *IEEE Trans. Syst. Man Cybern.* SMC-9, pp.62-66, 1979

[87] Sahoo, P., Wilkins, C. and Yeager, J., "Threshold selection using Renyi's entropy," *Pattern Recogn.* Vol. 30, pp.71-84, 1997

[88] Huang, L.K. and Lam, F.K., "Image thresholding by minimizing the measures of fuzziness," *Pattern Recogn.* Vol. 28, pp.41-51, 1995

[89] Cheng, H.D. and Chen, Y.H., "Fuzzy partition of two-dimensional histogram and its application to thresholding," *Pattern Recogn.* Vol. 32, pp.825-843, 1999

[90] Jain, A.K., Duin, R.P.W. and Mao, J.: "Statistical pattern recognition: A review," *IEEE Trans. Pattern Anal. Machine Intell.*, vol.22, no. 1, pp.4-37, 2000.

[91] Zhang, Y.M., Kovacevic, R.: "Real time sensing of sag geometry during GTA welding," Journal of Manufacturing Science and Engineering, Transactions of the ASME, 119: 151-160, 1997.

VITA

Zhenzhou Wang was born on the 1st of November 1977 in China. He receives his bachelorship and mastership from the electrical engineering department of Tianjin University in 2000 and 2003 respectively. He worked as R&D engineer in Tianjin Sanyo in 2003 and software engineer in Beijing CECW in 2004 and 2005. He becomes a PhD student of the electrical and computer engineering department in University of Kentucky from August, 2005. He has the following publications:

[1] Zhenzhou Wang and Guiling Li: "A New Cross Based Gradient Descent Search Algorithm for Block Matching in MPEG-4 Encoder," Chinese Journal of Electronics. Vol.12, No.4, Oct. 2003, Indexed by EI

[2] Zhenzhou Wang and Guiling Li: "Rate Control for MPEG-4 Bit Stream", TRANSACTIONS OF TIANJIN UNIVERSITY 2003 Vol.9, No.4

[3] Zhenzhou Wang and Guiling Li: "New Diamond Block Based Gradient Descent Search Algorithm for Motion Estimation in the MPEG-4 Encoder", Transactions of Tianjin University, 2003 Vol.9, No. 3

[4] Zhenzhou Wang and Guiling Li: "The Optimization of Algorithms of MPEG-4 Encoder", the sixth national television, satellite, overlay, audio and video technical seminar

[5] Zhenzhou Wang and Guiling Li: "MPEG-4 Video Stream Analysis Tool", Electron measurement technology.

[6] Delicia Siaw-Chiing Woon, Laurence G. Hassebrook, Daniel L. Lau, and Zhenzhou Wang: "Implementation of Three Dimensional Linear Phase Coefficient Composite Filter

For Head Pose Estimation ", Automatic Target Recognition XVI, SPIE Defense and Security Symposium

[7] Zhenzhou Wang and Yuming Zhang: "Image Processing Algorithm for Automated Monitoring of Metal Transfer in Double-Electrode GMAW", Journal of Measurement Science and Technology

[8] Zhenzhou Wang and Yuming Zhang: "Brightness Based Selection and Edge Detection Based Enhancement Separation Algorithm for Low Resolution Metal Transfer Images", Accepted by IEEE Transactions on Automation Science and Engineering

[9] Zhenzhou Wang and Yuming Zhang: "Processing of Low Resolution Metal Transfer Images," IEEE IECON 2007, TaiPei

[10] Zhenzhou Wang and Yuming Zhang: "Robust and Automatic Segmentation of A Class of Fuzzy Edge Images," in review for IEEE Transactions on Automation Science and Engineering

[11] Zhenzhou Wang and Yuming Zhang: "Monitoring of Metal Transfer Process: Simplified Optical System and Automatic Image Processing Algorithms," ready for journal submission

Among the published papers, [7] is about the algorithms proposed in Chapter 3 and Chapter 4. [8] is about the algorithm proposed in Chapter 5. [10] and [11] are about the algorithms proposed in Chapter 7.