EXPERIMENTAL AND SIMULATED AIR BENDING STUDY
ASSISTED BY IMAGE RECOGNITION

Contreras Garza David, Hendrichs Troeglen Nicolás, Siller Carrillo Héctor, Flores Valentín Abiud

Centro de Innovación en Diseño y Tecnología (CIDyT), Tecnológico de Monterrey, Campus Monterrey;
E. Garza Sada 2501 Sur; 64849, Monterrey, N.L.; México.
Teléfono (81) 8358 1400 ext 5324
dvd.cnt@gmail.com, nicolas.hendrichs@itesm.mx, hector.siller@itesm.mx, abiud.flores@itesm.mx

RESUMEN

Los Aceros Avanzados de Alta Resistencia (AHSS por sus siglas en inglés) están siendo utilizados en la industria por su resistencia y bajo peso, pero predecir correctamente su geometría final después del formado todavía requiere más estudio. Por una parte, el doblado al aire genera, por naturaleza, una geometría no-perfecta que es difícil de determinar, por lo que se requiere de formas alternativas de medición.

El software de reconocimiento de imágenes es ampliamente utilizado para aplicaciones biomédicas y biométricas, y ha demostrado medir y caracterizar los diferentes aspectos de una imagen de forma automática y confiable.

La metodología propuesta en este estudio, aplica para predecir la geometría final de AHSS doblados al aire. El proceso fue asistido por un algoritmo de reconocimiento de imagen para detectar la geometría experimental de forma automática y mucho más rápida que los métodos de medición convencionales.

ABSTRACT

Advanced High Strength Steels (AHSS) are being used in industry for their strength and low weight, but correctly predicting their final geometry after forming still requires further study. Moreover, air bending generates a non-perfect geometry by nature which is difficult to determine, so there is a need for alternate ways of measurement.

Image recognition software is widely used for biomedical and biometrics applications and has proven to measure and characterize different aspects of an image automatically and in a reliable way.

The proposed methodology of this study, used to predict the final geometry of AHSS after air bending. The process was assisted by an image recognition algorithm so that the experimental geometry could be detected automatically and much faster than current conventional measurement methods.

INTRODUCTION

The primary reason behind the use of AHSS in automobiles is the reduction of fuel consumption and the safety of occupants (Koppel Conway, 2009). Furthermore, due to their higher strength, manufacturers hope to reduce their manufacturing costs. Due to their superior mechanical properties, crashworthiness and mass avoidance there has been an increase in demands for AHSS in the automotive industries (Hudgins, 2010). However because of their low formability their implementation has been achieved only in some vital parts in the automobile where a difference in crash test performance really has paid off (Koppel Conway, 2009).

Spring-back deformation has becomes a critical problem when processing AHSS (Keeler, 1994). Industry has a specific need to predict it correctly to overcome manufacturing difficulties and one way to study these phenomena is the air bending test (ABT).

RELATED WORK

Many experimental and analytical developments for the ABT have been reported; so for digital image recognition applied to many different fields. However, the best integration of these two areas was done in Girona, Spain back in 2005 (García-Romeu, 2005). García-Romeu’s work surveyed the state of the art in spring-back prediction and developed an algorithm for the detection of curvature and angle of ABT samples.

Sheet Metal Bending

Different methods are commonly used for sheet bending (shown in Figure 1), such as folding (a), air bending (b), rotary bending (c), and flange bending (d) (Marciniak, 2002). Each of these
processes features different characteristics in the final product like; curve profile, residual stress, thickness and geometry (Kalpakjian, 2003).

![Diagram of bending processes]

*Figure 1. Different process of bending sheet-metal: (a) folding, (b) air bending, (c) rotary bending and (d) flange bending (Marciniak, 2002)*.

From the different methods to achieve the desired bend one of the oldest methods is air bend, which does not require a specific bottom die for each bending angle, as opposed to “V” die bending which bottoms on a specific geometry to assure bend radius and bend angle. Nonetheless, air bending is regaining relevance in small lot production because of its speed and low cost. However, air bending requires a deep understanding of the phenomena occurring in the material to achieve the desired final angle and radius.

The geometry obtained from air bending is not a perfect segment of a circle, even though the punch may be so. The sheet metal is normally flexed over the unsupported length as shown in Figure 2.

**Image Recognition**

After each bend the desired final geometry, after spring-back, is a part with defined bend angle and bend radius. The assumption is a circular profile in the bend (Kalpakjian, 2003). This is not normally the case for air bend; however, the best fit circle and the best fit angle are normally used to describe such geometries (Diegel, 2002), (Precision Metalforming Association, 2004).

![Image of air bending]

*Figure 2. In air bending, the sheet metal curve is different to the punch radius and extends beyond the punch.*

Image recognition is widely used in biometric and biomedical applications, thus there are some implemented algorithms that recognize measure and modify certain aspects of an image. Thus, the idea is to use this technique to measure air bend samples as already shown by García-Romeu (García-Romeu, 2005).

One of the ways to acquire a good quality digital image for the recognition software is a digital camera (Lin, 2010). The image can later be treated to acquire the desired characteristics of contrast that will be transformed to numbers in a matrix, either in a black and white image or a gray scale.

According to (García-Romeu, 2005), to obtain a digital image of an object, with enough contrast to be detected by image recognition software, a white paint coat on the face to be analyzed will suffice. For maximum contrast, a black opaque surface in the background is adequate.
Binary images are recommended for morphology detection (Russ, 1995). In this way background characteristics are separated completely from the object characteristics. It is also needed to clean the image from defects like dust or cracks. For this, a number of different filtering methods that add, subtract or eliminate fractions of the image automatically can be used (González, 2009).

Once a B&W, clean image is obtained it can be processed by means of any proven general methodology (Reza Fallahi, 2010). These methodologies use image characteristics such as a contrast ratio, color gamut or general pixel (px) size. Then, a specific area of the image can be selected for analysis, be it feature extraction, measurement or classification (see MatLab® code in Appendix A).

**Image Thinning**

Once a single, high contrast white image is achieved the software can analyze different aspects of it. In the case of bend sample measurements (angle and radius) the software makes the image so thin as to retain only 1 px of the feature of interest.

![Figure 3. Example of a black & white filtering and thinning steps for finger-vein image recognition (Cheng-Bo, 2009).](image)

This process is known as thinning and is used to characterize general geometry as if it were thin branches representative of the feature to be characterized (example in Figure 3).

**EXPERIMENT**

The experiment carried out was a typical air-bending test similar to a 3 point bending test. The experimental tool (shown in Figure 4) is capable of performing stretch bend test and air bend tests and was setup in a tension test machine. The test specimen was painted in white on one side and an opaque black surface was attached to the back of the machine to get the best possible contrast on the digital images. Two pictures were taken during the test. The first one was taken after the desired punch penetration was achieved (Figure 5) and the second one after removing the load (Figure 6). Punches available for the bending tool had radiuses of 7.5 mm, 12.0 mm, 15.0 mm and 22.5 mm respectively.

![Figure 4. Air bending tool with test sample.](image)

![Figure 5. Test sample before spring-back.](image)

![Figure 6. Test sample after spring-back.](image)

The test sample were taken from a 3 mm thick DP600 sheet in 3 direction, namely 0°, 45° and 90° with respect to the rolling direction.

**SIMULATION**

Air bend Test (ABT) simulations were performed with DEFORM 3D®, a software especial developed for simulating forming operations involving large displacements. The simulation setup, shown in Figure 7, was a simplified version of the holding tool. The control variable of the simulation was the punch stroke measured during the physical bending tests so as to achieve similitude of both experiments. The simulation did not require any special boundary conditions because in air bending the work piece has to slip into the die.
To optimize the mesh, only the contact zone with the punch had a much denser mesh than the rest of the work piece (finer by a factor of 3), as shown in Figure 8. This approach helped to reduce the processing time of each simulation. Sensitivity analysis was made by testing a different number of elements for the meshes in one scenario. For this simple geometry a number of elements around 1,000 were proven to be adequate. Smaller number of elements generated inconsistent results and even unsolvable systems. Meshes above 1,000 elements had stable results but the solution time did increase significantly.

The material characteristics were taken from Corona’s work (Corona, 2010), as he characterized the anisotropic properties of a 3 mm thick AHSS DP600 sheet. Anisotropy was defined with Hill’s quadratic approach (Figure 9) and flow stress as a curve (Figure 10).

These tests were performed on an ideal geometry designed in Solid Edge®. The piece was exactly 1mm thick, 1mm internal radius and a bend angle of 90°. The algorithm approximated the geometry as shown in Figure 14. The black thin line represents the algorithm output and a thicker color line represents the idealized input image. For this ideal test the error was 0.001% for the angle and 1.53% for the radius.
The algorithm was also tested with images from arbitrary test samples. The algorithm results for radius and bend angle are shown in Table 1 and Table 2. These results were compared to physical measurement of the samples taken on a coordinate measuring machine. The measurements were done by taking 4 points in the bending zone and 3 points on the straight lines.

Table 1. Comparison of the physical vs. algorithm measurements on the radius of test samples.

<table>
<thead>
<tr>
<th>Bend Radius</th>
<th>Thickness (mm)</th>
<th>Physical (mm)</th>
<th>Algorithm (mm)</th>
<th>Error (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1.20</td>
<td>10.53</td>
<td>9.85</td>
<td>6.45</td>
</tr>
<tr>
<td></td>
<td>2.15</td>
<td>6.57</td>
<td>6.08</td>
<td>6.53</td>
</tr>
<tr>
<td></td>
<td>4.68</td>
<td>7.65</td>
<td>8.03</td>
<td>4.95</td>
</tr>
<tr>
<td></td>
<td>7.89</td>
<td>29.85</td>
<td>30.57</td>
<td>2.42</td>
</tr>
</tbody>
</table>

Table 2. Comparison of the physical vs. algorithm measurements on the bend angle of test samples.

<table>
<thead>
<tr>
<th>Bend Angle</th>
<th>Thickness (mm)</th>
<th>Physical (deg)</th>
<th>Algorithm (deg)</th>
<th>Error (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1.20</td>
<td>119.54</td>
<td>119.43</td>
<td>0.09</td>
</tr>
<tr>
<td></td>
<td>2.15</td>
<td>102.69</td>
<td>104.40</td>
<td>1.66</td>
</tr>
<tr>
<td></td>
<td>4.68</td>
<td>119.91</td>
<td>120.32</td>
<td>0.34</td>
</tr>
<tr>
<td></td>
<td>7.89</td>
<td>89.77</td>
<td>89.74</td>
<td>0.04</td>
</tr>
</tbody>
</table>

RESULTS

Force and displacement for each DP600 sample was measured on the tension test machine. Figure 15 is a summary of the data acquired during the experiment; sample ID includes orientation with respect to the rolling direction and punch radius.

The simulation software (DEFORM 3D®) rendered the same forces and displacements output, which is in good agreement with the experimental evidence (Figure 16).

The experiments and the simulation are in good agreement; as the displacement-force results compare rather good for all different punch radiiuses (Figure 17, 18, 19 and 20).

Furthermore, to compare the physical results with the simulations, pictures were taken from the DEFORM 3D® displays and analyzed with the image recognition algorithm; both before and after spring-back.

When comparing the measurements of radius (Figure 21 and Figure 22) it can be seen that there is an area of opportunity in the simulations prediction; in all cases the simulated radius is 30%, or more, smaller than the corresponding experimental value.
This procedure may lead to false assumptions. For bending, where spring-back is very noticeable, results (Hudgins, 2010; Corona, 2010; Gonzalez Angel, 2010) compare the experiments, the simulations and experimental measurements with the image recognition software can be seen.

However, there is a noticeably bigger discrepancy between the final angle measurements, as shown in Figure 24; which compares the experiments, the simulations and the CMM values. These later compare well to the experiments, but differ greatly from the simulations.

**CONCLUSIONS**

It is common practice to use force and displacement values con “calibrate” simulation results (Hudgins, 2010; Corona, 2010; Gonzalez Angel, 2010). The present study indicates that for bending, were spring-back is very noticeable, this procedure may lead to false assumptions.

The data acquired from the tensile testing machine during the air bending experiments is in very good agreement with the estimates from the 3D simulations (Figure 17, Figure 18, Figure 19 and Figure 20). However, when working with final geometry considerations, mayor differences may arise due to a lack of precision of the constitutive model used in the simulating environment. As seen in Figures 21, 22, 23 and 24, the differences between experimental and simulation results can be significant. Further experimentation and simulation should be carried out to improve the estimation precision.

However, rapid measurement methods such as the one developed here, that allow comparing physical experimentation results to simulation results via image recognition software, seems appropriate for such further research.

The implementation of image recognition software may also be useful for other kinds of deformation processes as long as they are visible,
and there is a clear and orthogonal line of sight. Potentially, the software could be automated for different visual experiments and the code could also be tailored to interpret video so that it could assist the analysis of each step of any deformation process. Furthermore, it could be integrated into the hardware and software of specialized machines for real time control functions.

Although CMMs are very accurate, said accuracy depends on the points taken by the technician which are only a few (typically less than 10). The algorithm here developed can help eliminating the need for these highly qualified and experienced technicians for CMM measurements, as it does not depend on the zones chosen to acquire the data. The algorithm simply measures the whole curve based on tens or even hundreds of points.

At the present stage of development, the software still requires a fair amount of user input to preprocess the images (generate enough contrast) and to find the circles and lines within the overall geometry.

However, as the algorithms get more elaborate and find their way into more sophisticated applications, they also will become faster, self-sufficient and more precise.

REFERENCES


Appendix A: MathLab® code for Image Recognition

The code included reads a group of preprocessed images (with sufficient contrast) and generates the bend angle and the bend radius for each of them.

```matlab
clear,clc
muestra=['A22.jpg'];
%list of test files ['sample1.jpg';'sample2.jpg']
[tam m]=size(muestra);
espesor=[3.01]; %sheet metal thickness [2:3]
for k=1:tam %for a sample size tam
tic%to take time t1
root=imread(muestra(k,:));%gets k-th image
t=espesor(k,:);%gets k-th thickness
mod=limpiar(root);%calls a subroutine to get a binary image and clean defects
limwidth=800; %limits the image width
[trash x_mod]=size(mod);
if (x_mod >= limwidth)
    mod=imresize(mod,limwidth/x_mod);
    root=imresize(root,limwidth/x_mod);
end;
clear x_mod
mod=aislar(mod);%gets only the biggest area

```

for k=1:tam %for a sample size tam
tic%to take time t1
root=imread(muestra(k,:));%gets k-th image
t=espesor(k,:);%gets k-th thickness
mod=limpiar(root);%calls a subroutine to get a binary image and clean defects
limwidth=800; %limits the image width
[trash x_mod]=size(mod);
if (x_mod >= limwidth)
    mod=imresize(mod,limwidth/x_mod);
    root=imresize(root,limwidth/x_mod);
end;
clear x_mod
```

for k=1:tam %for a sample size tam
tic%to take time t1
root=imread(muestra(k,:));%gets k-th image
t=espesor(k,:);%gets k-th thickness
mod=limpiar(root);%calls a subroutine to get a binary image and clean defects
limwidth=800; %limits the image width
[trash x_mod]=size(mod);
if (x_mod >= limwidth)
    mod=imresize(mod,limwidth/x_mod);
    root=imresize(root,limwidth/x_mod);
end;
clear x_mod
```

mod=aislar(mod);%gets only the biggest area

t_pv=sum(mod(:,1));%gets the vertical thickness in pixels

```
mod=aislar(mod);%gets only the biggest area

t_pv=sum(mod(:,1));%gets the vertical thickness in pixels

```

```
t_pv=sum(mod(:,1));%gets the vertical thickness in pixels

```

```matlab
for k=1:tam %for a sample size tam
tic%to take time t1
root=imread(muestra(k,:));%gets k-th image
t=espesor(k,:);%gets k-th thickness
mod=limpiar(root);%calls a subroutine to get a binary image and clean defects
limwidth=800; %limits the image width
[trash x_mod]=size(mod);
if (x_mod >= limwidth)
    mod=imresize(mod,limwidth/x_mod);
    root=imresize(root,limwidth/x_mod);
end;
clear x_mod
```

```matlab
mod=aislar(mod);%gets only the biggest area

```

```matlab
t_pv=sum(mod(:,1));%gets the vertical thickness in pixels

```

```matlab
for k=1:tam %for a sample size tam
tic%to take time t1
root=imread(muestra(k,:));%gets k-th image
t=espesor(k,:);%gets k-th thickness
mod=limpiar(root);%calls a subroutine to get a binary image and clean defects
limwidth=800; %limits the image width
[trash x_mod]=size(mod);
if (x_mod >= limwidth)
    mod=imresize(mod,limwidth/x_mod);
    root=imresize(root,limwidth/x_mod);
end;
clear x_mod
```

```matlab
mod=aislar(mod);%gets only the biggest area

```

```matlab
t_pv=sum(mod(:,1));%gets the vertical thickness in pixels

```

```matlab
for k=1:tam %for a sample size tam
tic%to take time t1
root=imread(muestra(k,:));%gets k-th image
t=espesor(k,:);%gets k-th thickness
mod=limpiar(root);%calls a subroutine to get a binary image and clean defects
limwidth=800; %limits the image width
[trash x_mod]=size(mod);
if (x_mod >= limwidth)
    mod=imresize(mod,limwidth/x_mod);
    root=imresize(root,limwidth/x_mod);
end;
clear x_mod
```

mod=aislar(mod);%gets only the biggest area

t_pv=sum(mod(:,1));%gets the vertical thickness in pixels

```
mod=aislar(mod);%gets only the biggest area

t_pv=sum(mod(:,1));%gets the vertical thickness in pixels

```

```matlab
t_pv=sum(mod(:,1));%gets the vertical thickness in pixels

```

```matlab
t_pv=sum(mod(:,1));%gets the vertical thickness in pixels

```

```matlab
t_pv=sum(mod(:,1));%gets the vertical thickness in pixels

```

```matlab
t_pv=sum(mod(:,1));%gets the vertical thickness in pixels

```

```matlab
t_pv=sum(mod(:,1));%gets the vertical thickness in pixels

```

```matlab
t_pv=sum(mod(:,1));%gets the vertical thickness in pixels
```
rpi=rpi-0.5;
%Non linear regresion for exterior radius
mod=root;
mod=sub(mod); %gets the bottom border
mod=doblez(mod,angs,t_p);
mod=aislar(mod);
np=nnz(mod);
[rpe,Ye,Xe]=regresion(mod); %makes a non linear regression for a circle
rpe=rpe+0.5;
npt=np+npi;
r_p=(rpe-t_p)*npe/npt+rpi*npi/npt;
r=r_p*pix2mm;
t2=toc;
tiempo(k)=t1+t2; %gets the time for the k-th iteration
radio(k)=r;
angulo(k)=ang;
npix(k)=npi+npe;

figure(2) %shows results
m=ceil(sqrt(tam));
subplot(m,m,k),
plot(Xi,Yi+t_p,'.r',Xe,Ye,'.g');
hold on
plot(Xi,rad(rpi,Xi)+t_p,'-b',Xe,rad(rpe,Xe),'-b');
title({muestra(k,:);['Angº= ',num2str(ang)];['Rad int= ',num2str(r,4)];});
axis equal;
hold off
end

function [img]=aislar(img)
%This functions isolates the biggest white area in a BW image
L=bwlabel(img);
propied=regionprops(L,'Area','BoundingBox');
Amax=max([propied.Area]);
s=find([propied.Area]<Amax);
for n=1:size(s,2)
d=round(propied(s(n)).BoundingBox);
img(d(2):d(2)+d(4),d(1):d(1)+d(3))=0;
end

function [img]=borde(img,t)
%deletes imperfections in the border
[y_img,x_img] = size(img);
img=imcrop(img,[t,0,x_img-2*t,y_img]);

function [img] = doblez(img,angs,t)
%only the bending zone

function [y] = rad(r,x)
%Function for the circle for the lower quadrants
y=r-sqrt(r.^2-x.^2);

function [r_p,Y,X] = regresion(img)
%gets the non linear regression for a circle
[Y X]=find(img==1); %get's the XY coordinates
Y=Y-Min(Y); %adjusts the Y values to start in 0
X=X-mean(X(find(Y==0))); %centers the X values
[r0,n]=size(X);r_p=nlinfit(X,Y,@rad,r0); %makes the non linear regression

function [img]=sub(img)
%gets the bottom border
B=[0,1,0; 0,1,0; 0,-1,0];
img=bwhitmiss(img,B);

%This function converts the image to BW
img=rgb2gray(img);
umb=graythresh(img);
img=im2bw(img,umb);

function [img]=media(img)
%Makes a thin line
img=bwmorph(img,'thin',Inf);

function [c]=minrect(img)
%get's the minimum coordinates for the image
[m,n]=find(img==1);
c=[min(n),min(m),range(n),range(m)];

function [y] = rad(r,x)
%Function for the circle for the lower quadrants
y=r-sqrt(r.^2-x.^2);

function [r_p,Y,X] = regresion(img)
%gets the non linear regression for a circle
[Y X]=find(img==1); %get's the XY coordinates
Y=Y-Min(Y); %adjusts the Y values to start in 0
X=X-mean(X(find(Y==0))); %centers the X values
[r0,n]=size(X);r_p=nlinfit(X,Y,@rad,r0); %makes the non linear regression

function [img]=sub(img)
%gets the bottom border
B=[0,1,0; 0,1,0; 0,-1,0];
img=bwhitmiss(img,B);

function [img]=borde(img,t)
%deletes imperfections in the border
[y_img,x_img] = size(img);
img=imcrop(img,[t,0,x_img-2*t,y_img]);

function [img] = doblez(img,angs,t)
%only the bending zone

function [y] = rad(r,x)
%Function for the circle for the lower quadrants
y=r-sqrt(r.^2-x.^2);

function [r_p,Y,X] = regresion(img)
%gets the non linear regression for a circle
[Y X]=find(img==1); %get's the XY coordinates
Y=Y-Min(Y); %adjusts the Y values to start in 0
X=X-mean(X(find(Y==0))); %centers the X values
[r0,n]=size(X);r_p=nlinfit(X,Y,@rad,r0); %makes the non linear regression

function [img]=sub(img)
%gets the bottom border
B=[0,1,0; 0,1,0; 0,-1,0];
img=bwhitmiss(img,B);