# Improving accuracy in 2D wire bending machines
## (Numerical and empirical approach)

**Mahmoud Abdel Mohsen**

Ostwestfalen-Lippe University of Applied Sciences and Arts
Emilienstraße 45, 32756 Detmold, Germany
Mahmoud.abdel@stud.th-owl.de

***Abstract:*** *Accuracy in wire bending process is a complex industrial problem, widely regarded to springback behaviour of materials. Moving from a mass production industrial applications to mass customization in architectural applications, requires a higher degree of accuracy in prototyping. This paper discusses increasing accuracy in the cold bending of $3mm$ metal wire using "D.I.Wire v1" [1] CNC desktop machine for. The default bending process has a maximum relative error of 45% and maximum absolute error of 2°, based on manual material calibration and polynomial curve fitting model to derive motion and springback parameters. The target of this study is to achieve a maximum relative error of 10% and maximum absolute error of 1°. The scope of this paper is to analyse and improve accuracy in 2D wire pending process . The paper consists of three parts: analysis, numerical and empirical approach. The fist part is an analysis for bending machine's hardware, software and process. In the second part, FEA simulation were used to analyse machine and material behaviour. The third part is an empirical approach. In which, computer vision is used instead of manual calibration process to automate bending process and to reduce human measurement error. It was clear that, both numerical and empirical approaches are inseparable and should be used iteratively for best results. Numerical model showed that, motion function had a bigger influence over springback effect in producing bending error. While in empirical approach, computer vision was proved reliable for measurement and calibration processes in 2d wire bending applications.*

***keywords:*** *Springback, CNC wire bending, Computer vision, D.I.Wire, ANSYS.*

## 1 Background

Metal forming in general is used across many industries. However cold forming has a high degree of uncertainty due to the non-linear plastic behaviour of metals. In mass production industries, automotive industry for example FEA and trial and error methodologies are used to achieve the design form, which affects drastically the cost of final product. In mass customization production, trial and error approach is not cost efficient. This paper discusses increasing accuracy in the cold bending of $3mm$ metal wire using out of the box CNC desktop machine for architectural applications (Fig. 1) . A root cause analysis, reverse engineering, FEA modelling and empirical study were carried out to find out the main reasons of inaccuracy. The results show the complexity of the problem. It extends bending error to motor, communication protocol, machine mechanics and springback behaviour of the material[1].


**Figure (1):** *D.I.Wire v1, a desktop 2d bending machine.*

## 2 State of the art

### 2.1 Automotive industry

In addition to sheet metal bending for exterior car body, automotive industry relays heavily on 3D tube bending for coolant, fuel and brake lines (Fig. 2). However, cars mass production would cover up the cost of extensive prototyping process for each part. For example, PASS [2] is a Volks Wagon subcontractor for pipe production, who uses a high end tube 3D bending machine ()Fig. 3), which is manufactured by BLM Group. The machine[3] comes with it's own FEA simulation software for tube and wire bending. Even so, each part needs at least 24 hours of prototyping.

### 2.2 Construction industry

Since the invention of reinforced concrete, manual steel bar cold bending takes a place on most construction sites. Bending angels are mostly standard, 90, 60, and 45°, which make it easier for experienced construction workers to eyeball it. However, in complex geometries, bar bending becomes increasingly time and workmanship demanding process. For example, Stuttgart 21 project, designed by Ingenhoven architects. An architecturally highly sophisticated shell roof, supported by 28

[1] DIWire Bender by Pensa is licensed under CC BY-SA 3.0. To view a copy of this license, visit https://creativecommons.org/licenses/by-sa/3.0

[2] https://pass.de/startseite/

[3] https://www.blmgroup.com/en/tube-bending/smart

**Figure (2):** *A 3D bent coolant tubes for VW group with the help of SMART tube bending machine.*


**Figure (3):** *SMART tube bending machine from BLM Group, used by VW group subcontractor for coolant, fuel, and break lines production.*

geometrically highly complex chalice-shaped columns, which can be divided into 23 standard columns, four flat columns and a larger special column (Fig. 4a). The reinforcing bars are bent in a bending shop specially set up for the project (Fig. 4c) , and checked by means of true-to-scale laser projection. 11,000 different, partly three-dimensionally curved bar shapes, have to be placed per column on the construction site. In order to ensure exact positioning, each component of the shell roof is provided with a coordinate list with Gauss-Krüger coordinates in addition to the reinforcement drawings. With the help of a surveyor, the guide bars can be precisely measured and further bars can be placed between them (Fig. 4b). For the correct assignment of the bars, the beginning and end of the bars are defined in the reinforcement drawings, which in turn are taken over by the bending company by means of a coloured marking in addition to the position number on the bar [2]. In spite of that, 50% of bent bars are rejected and becomes a steel waste. Beside the additional costs and time associated with the process, steel has one of the highest embodied carbon coefficient in building materials, which leads to an unavoidable increase of environmental impact.

### 2.3 Architectural research

There was only a few architectural researches about the topic. One of them was about robotic rod bending process by Maria Smigielska [3]. Her approach was to have a minimum robotic setup, one tool head is used for feeding, bending and rotating. The downside of this setup was instability of boundary condition, which induced an additional lateral error in bending process. The most valuable part of her study was springback measurements dataset (Fig. 5b), which has some remarks:

- Springback measurements starts after 10° bending angle.
- Data has an average error of 2.5°.
- A high degree of uncertainty after 150° bending angle.

We can notice the similarity between here data and the standard engineering uniaxial tensile test graph; in plastic deformation area they both start with slope increase (hardening) until ultimate strength is reached, then followed by slope decrease (softening / necking) till 150° where partial fracture happens and leads to a higher degree of uncertainty in springback values. However its worth to mention that, they have obvious differences, mainly are:

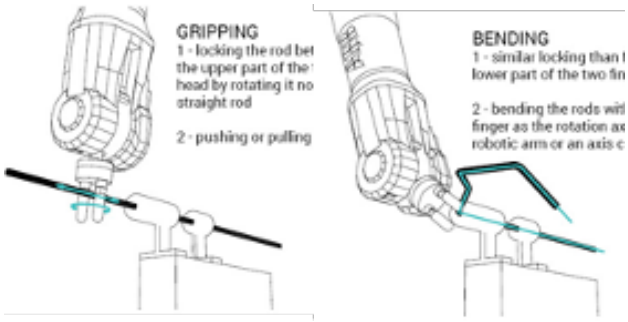
- Research data represents true stress-strain, while steel graph represents engineering stress-strain data; not considering the change in cross section during deformation.
- Bending differs from uniaxial action; in which bar cross section under goes both tensile and compression deformation in bending, while in uniaxial tensile test, cross section is mostly dominated by tension forces.

**(a)** *Stuttgart 21, an architecturally highly sophisticated shell roof, supported by 28 geometrically highly complex chalice-shaped columns.*

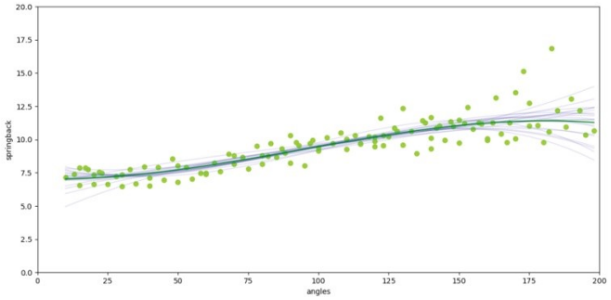
**(b)** *11,000 different curved bar shapes, have to be placed per column on the construction site.*

**(c)** *CAM rebar bending machine used on site for steel bending.*

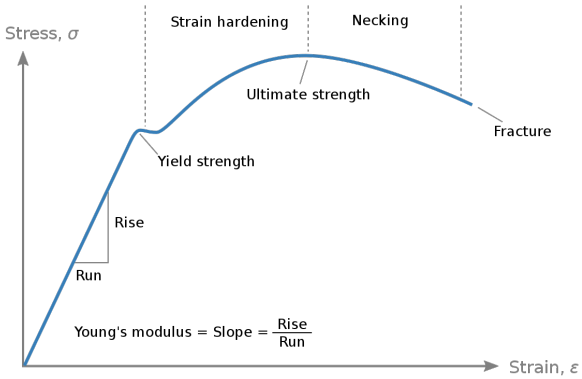
**Figure (4):** *Stuttgart 21 as a part of the Stuttgart-Ulm rail project is one of the largest European infrastructure projects, designed by Ingenhoven architects, Frei Otto was consulted on this project.*




**(a)** *Minimum robotic setup, one tool head is used for feeding, bending and rotating.*

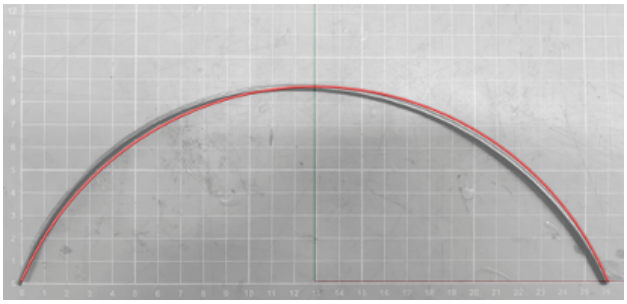



**(b)** *Springback measurements dataset of robotic bar bending research.*

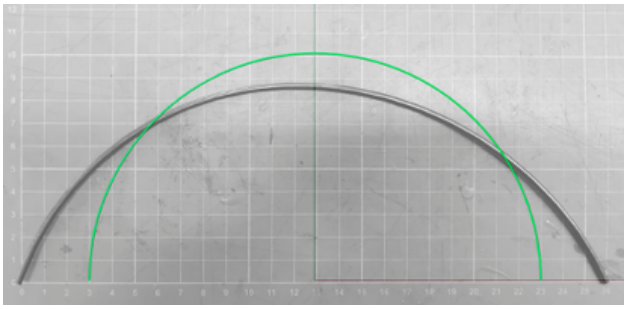



**(c)** *Standard engineering uniaxial tensile test graph.*

**Figure (5):** *Robotic rod bending research by Maria Smigielska.*



**(a)** *Incremental error,bending angel changes gradually, leading to a helix shape. Red line represents the nearest possible geometrical arc shape.*



**(b)** *Constant error, difference in radius between designed arc and bended wire. Green line represents the desired bending shape.*

**Figure (6):** *Different error types in wire bending, using Diwire desktop bending machine.*

## 3 Bending error types

The problem of bending machine inaccuracy was observed during bending high and low curvature arcs, where bending angel is repetitive, which has an incremental error effect. During the bending of semicircles, inconsistency in error between the first and second quadrant was also noticed. Errors can be grouped in two main domains; incremental and constant errors.

### 3.1 Incremental error

It was observed while bending the same angel repetitively, bending angel changes gradually, leading to a helix shape. Could be due to gearbox backlash or motor step skipping (Fig.6a).

### 3.2 Constant error

There is mostly a difference in radius between designed arc and bended wire. That is mainly due to calibration error and software calculation logic (Fig.6b).

## 4 Error Calculation

To calculate the amount of error, desired bending angels were calculated by dividing the curve into equal cord lengths and greater than machine minimum bending length ($1.5\,cm$) , and by subtracting cords slop we get the desired bending angels. To calculate bending angles result, we would need to find the unknown radius of bent

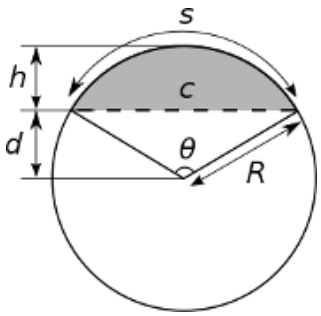



**Figure (7):** *Arc segment diagram, used in bending angels calculations.*

curve. Since arc height and width of bent wire are measurable, we can calculate curve radius from (Equations 1, 2, 3). For relative and absolute error calculations, a python code was used (Source code.1).

$$r = \frac{d^2}{8h} + \frac{h}{2} \tag{1}$$

$$\theta_{arc} = 2\sin(\frac{c^2}{2r}) \tag{2}$$

$$\theta_{bend} = \frac{\theta_{arc}}{n_{div}} \tag{3}$$

Where:
$r =$ arc radius, $d =$ arc width, $h =$ arc height, $\theta_{arc} =$ arch angle, $\theta_{bend} =$ segment bend angle and $n_div =$ number of divisions (cords).

# Part I

# Analysis

## 1 Methodology

Problem simplification by isolation strategy is used to reduce the complexity of the main problem. In order to create an accurate simulation, bending mechanism and software logic of bending machine, has to be clearly understood. Hence, a reverse engineering and extensive measurements process was carried out to understand root cases of bending error.

## 2 Default bending process

Bending process begins with material calibration by creating material profile. The software provides two sizes of material diameter, $1/8\,inch\,(3.18mm)$ and $1/16\,inch\,(1.59\,mm)$. Three modes of calibration are available, quick, Standard and high. The difference is the number of measured springback angels. Quick calibration measures 10 angels in total, 5 angels clockwise, 10, 20, 45, 75 and 110 degree respectively and

```python
import numpy as np

# r : arc radius
# h : arc height
# c : chord length (arc width)
# ceta : total arc angle
# div : number of division (number of cords)
# divd_ceta : divided total = bended wire angle

# half circle
height = np.array([100, 105, 95, 100])
width = np.array([200, 235, 238, 202])
divisions = np.array([20, 20, 20, 20])

# saddle curve
s_height = np.array([20, 12, 15, 21])
s_width = np.array([200, 223, 213, 205])
s_divisions = np.array([13, 13, 13, 13])

def bending_angel(h, c, div):

r = (h / 2) + ((c ** 2) / (8 * h))
ceta = 2 * np.arcsin(c / (2 * r))

divd_ceta = ceta / div
bended_angel = np.rad2deg(divd_ceta)

absolute_error = bended_angel[0] -
↪  bended_angel
relative_error = 100 *
↪  (absolute_error/bended_angel[0])

results = np.vstack([bended_angel,
↪  absolute_error, relative_error])
return results

c_result = bending_angel(height, width,
↪  divisions)
print(c_result)
s_result = bending_angel(s_height, s_width,
↪  s_divisions)
print(s_result)
```

**Source code (1):** *Error calculation, Python snippet is used to calculate bending angles, absolute and relative error.*

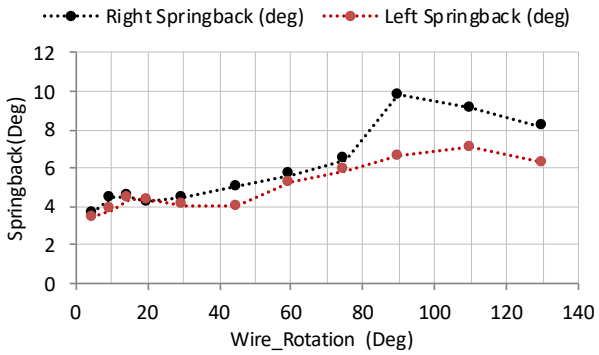



**Figure (8):** *Spingback values in both directions (default process), shows a maximum of 4° difference in springback values between positive and negative bending actions.*

the same angels in counter-clockwise direction. While standard and high calibration measure 11 angels each direction, 5, 10, 15, 20, 30, 45, 60, 75, 90, 110, 130 degree respectively with total of 22 measured angels (Fig.8).

The software will perform initial bending angel, go back to home position and wait for user input. The user controls bending pin by using right and left arrow on the keyboard. Once bending pin touches the wire, user clicks next angel in the software window. The software will read the current bending pin position from the machine and recorded as springback angle. The process to be repeated for all calibration angels and then saved in a JSON file under user defined name.

Next step is import bending drawing. The software accepts SVG and DXF files. Objects should be lines, polylines, arcs and Circles. No closed shapes are accepted, and minimum gap is $4\,mm$. Software converts all curves to segmented lines with minimum $12\,mm$ length and gives you the ability to move and delete end points for better curve precision.

Last user clicks start bending to convert the drawing to G-code and send it to machine through USB connection interface.

## 3 Machine analysis

The machine body is made out of coated steel alloy. Machine dimensions are 9.5 x 1 4.5 x 6.75 inch (W:240 x L:370 x H:170 mm), weights 23lbs, 13oz. (10.5kg), power Requirement is 100-240 V, 1.8 A, 50-60 Hz and recommended operating temp is between 15 - 30°C. Parts on top plate are bending pins, bending head (Die), reset button, feed wheels and wire guides (Fig.9). On the back side: power switch, USB port and power jack. Internal parts are:

- 1x NEMA 23 brushless pi-polar stepper motor. 3.0 Amp max load. 1.6 Nm peak Torque (constant Torque). 1.8 degree/step. Used for bending action.

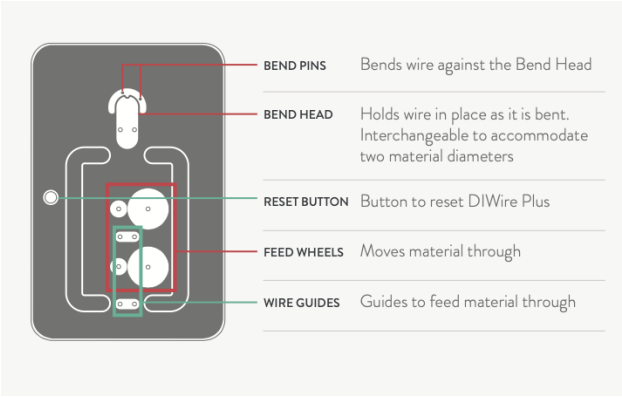



**Figure (9):** *Diwire bending machine top layout, shows active bending parts.*

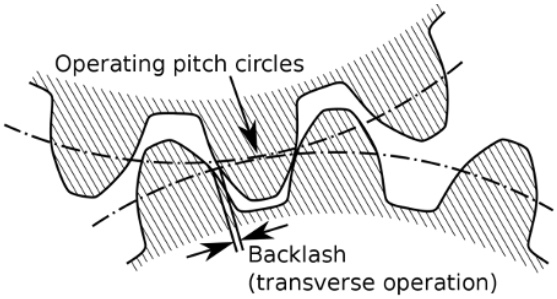



**Figure (10):** *backlash error due tolerance between gear teeth, however it must be allowed to prevent jamming.*

- 1x NEMA 23 Hybrid pi-polar stepper motor. 3.0 Amp max load. 09 to 1.9 N/m Torque (ramped). 1.8 degree/step. Used for wire feeding action.
- 1x closed gearbox connected to bending motor.
- 1x TinyG v8 CNC breakout board with embedded micro-controller (Atmel ATxmega192). Supports Micro-stepping up to 1/8th step (1, 2, 4, 8 Micro-steps).

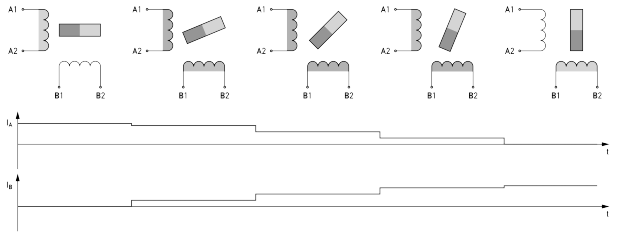
# 4 Machine limitations

## 4.1 Gearbox backlash

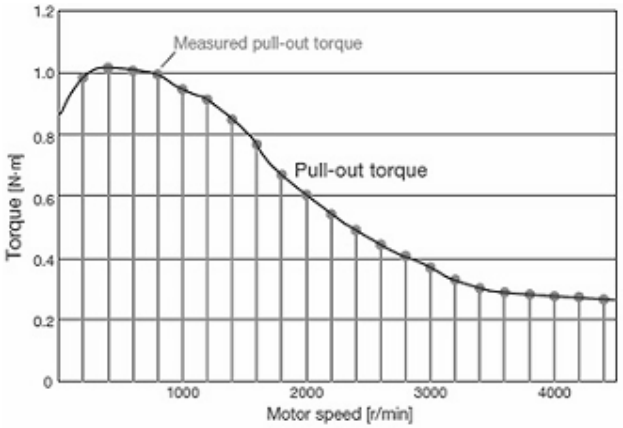
In any mechanical system, unavoidable loss in motion due to tolerance between gear teeth, happens with every change in direction (Fig.10) . The theoretical ideal would be zero backlash, but in actual practice some backlash must be allowed to prevent jamming. This value should be measured and compensated for, in every change in move direction.

## 4.2 Step skipping

This is the most problematic limitation with stepper motors in general. It happens due to drop in motor torque during motion which leads to an accumulative positioning error. Torque droppage in motor is mainly due to microstepping and motor speed.



**(a)** *Micro stepping, current is split between poles to reach an intermediate positions.*




**(b)** *Speed- Torque graph, shows that stepper motors are losing torque at high speeds.*

**Figure (11):** *Microstepping and motor speed are the main reasons for step skipping problem in stepper motors.*

### 4.2.1 Micro stepping

The machine uses a NEMA 23 pi-polar brushless stepper motor with 1.8 degree step to increase rotation precision, a 1/8 microsteps is achieved through PWM (Pulse Width modulation) at 50 KHz which reduces DC voltage between poles to reach an intermediate positions. However, this operation leads to at least 30% peak Torque droppage. Which could lead to step skipping during bending at low voltage positions (Fig.11a).

### 4.2.2 Motor speed

Inductance and lack of time for the current to build in the windings is the leading cause for stepper motors losing torque at high speeds, sufficient current cannot pass through the windings fast enough before the current is switched to the next phase, hence torque loss is imminent at high speeds (Fig.11b).

## 4.3 Data flow control

The control of commands going to the Atmel XMega microcontroller and the responses returned is a complex problem; it becomes significantly harder when machine bending feed rates are slower than G-code communication rates.

RS-232-style serial communication like that employed

by the TinyG between the FTDI USB-to-serial converter and the Atmel XMega microcontroller has several advantages but also a few disadvantages.

The advantages are that there are only two wires required for two-way communication: one to transmit (TX), and one to receive (RX). In order to accomplish talking in either direction over only one wire the two sides have to agree to the timing and format of the signalling.

The disadvantage of this style of communication is that the TX and the RX are each transmitted blindly. Other than the device on the other end sending a response (over the other wire) there's no way to know when a transmission has been received, which is usually just a problem of knowing if the other device is listening. There also no way to tell when the receiving device has been able to handle all of the data that it was sent. Solving this problem is called flow control.
There are many type of flow control, but there are two general methods:

- Software XON/XOFF.
- Hardware RTS/CTS (Request-To-Send, Clear-To-Send).

#### 4.3.1 Software flow control (XON/XOFF)

In software flow control, the two devices will send special data (XON and XOFF characters) over the line to indicate that it is ready for more data or that it doesn't have room for more data, respectively. This is problematic because the XON/XOFF characters are then part of the stream, and in some cases, that means that they go into the end of a transmit buffer. This means that all of the characters ahead of it in the buffer must transmit first.

#### 4.3.2 Hardware flow control (RTS/CTS)

With hardware flow control, two additional wires are used, one for the RX (RTS, or Request To Send) and one for TX (CTS, or Clear To Send). Once device controls RTS, and the other controls CTS. For each device, they set RTS when they have buffer space, and clear it when the buffer is (almost) full. When transmitting, the device reads the CTS line, and if it's set we send data. This means that the response to a buffer full condition is near immediate.

Regardless of which flow control method used, data quantity and speed has to be balanced between computer and the machine at any time point.

### 4.4 Data quantity control

- In-buffer has 28 lines space, half of it has to be kept free, otherwise it could overflow or starve.
- In (XON/XOFF) flow control, signalling is at the end of the message, so each G-code line should not exceed buffer limit.

### 4.5 Data speed control

- accessing buffer at high speed causes microcontroller's processor halt.
- Sending and receiving bitrate speed has to be matched with machine feed rates.
- In hardware flow control (XON/XOFF), due to network latency some data would be sent even after XOFF signal is received.
- In software flow control (RTS/CTS), RTS signal automatically clears buffer when it is almost full, even if the information is not processed yet.

### 4.6 Material limitations

The machine is designed for Imperial unit materials 1/8 inch (3.18 mm) and 1/16 inch (1.59 mm) rods, which are not available in Germany. Instead, a 3mm diameter steel wires were used in this experiment, which leaded to a 0.18 mm gap in bending head and 0.09 mm axis shift in wire bending mechanism. However, this effect was considered during FEA simulation.

## 5 Software analysis

The machine comes with a bundled MAC and PC software. However, the manufacturer provides an unbundled open source version of the software upon request. The software was written in Processing [4] and Java. The open-software version comes under no warranty CC-BY-SA 3.0 licences. It contains 7636 lines of code plus few external libraries. The main functionalities of the software are:

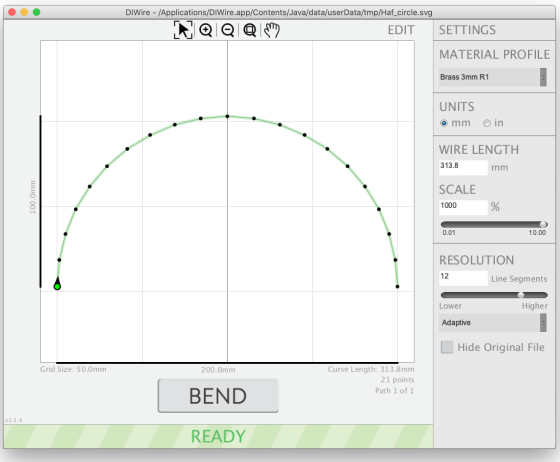
- Interactive GUI (Graphical user interface).
- SVG and CAD drawings interpretation.
- G-Code conversion (GPLv2 with the BeRTOS extension).
- Serial communication with TinyG v8 board using FTDI VCP USB driver.
- Motion calculation; since bending pins and the bent wire have different centre of rotation, this part converts motor rotation to bending angles.
- Springback compensation using a 3rd degree polynomial curve fitting function.

User interface has two modes, the first mode where user can import and bend wires (Fig. 12a), the second mode for calibration process (Fig. 12b).

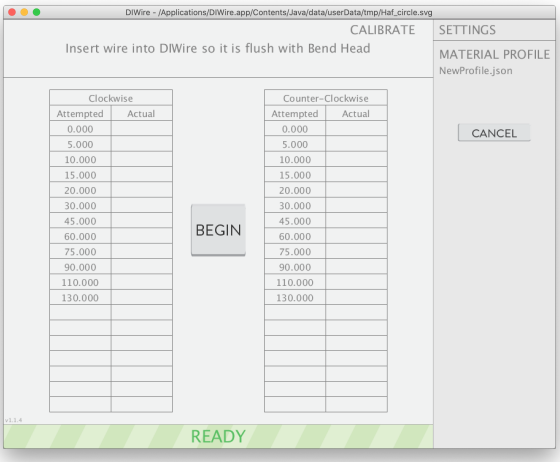
### 5.1 Software limitations

The current version of the software is not supported by the manufacturer any more, instead they have a new commercial version "Dwire 2.0" , which is a paid version of the software. Some of processing libraries are obsolete at the time of paper writing. The software is computationally demanding and it hangs sometimes with no obvious reasons.

[4] https://processing.org/




**(a)** *Bending interface, where user can import and bend wires.*




**(b)** *Calibration interface, for wire calibration process*

**Figure (12):** *Diwire v1, default bending software interface.*

# 6 Measurements

The main objective of this study is not to eliminate the error rather to have a constant error below accepted tolerance. Which means we should be able measure at a resolution equal to or less than the accepted tolerance. Hence, the consistency of measurement tools and methods is crucial for experiment replication, verification and validation.

## 6.1 Accepted tolerance

The total accepted tolerance for final parts is a maximum of 10% relative error and $\pm 1°$ absolute error. Which means that, measurement tolerance should not exceed those limits.

## 6.2 Measurement tools

- 0-300 mm metal ruler includes both mm and inches scale with 0.5 mm accuracy.
- Aristo protractor triangle with 1° angle and 1 mm length measurement accuracy .
- 0-150 mm Dial Calliper with 0.02 mm accuracy.
- Digital A4 scanner with max optical resolution 1200 x 2400 dpi.

## 6.3 Measurement methods

- Direct visual measurement.
- Manual tracing and scanning.
- Photography and CAD tracing.
- Computer vision, feature detection algorithms .

# Part II

# Numerical Approach

# 1 introduction

Although bending problem involves motion, a rigid body diagram can be drawn at any point of time. Hence, problem can be simplified as a statical structural problem by solving virtual work and equilibrium equations. The aim FEA simulation is to isolate and study springback function from motion function.
However, part of plastic deformation work would lead to noticeable heat, which will affect material mechanical properties. A better approach would be a coupled static structural-transit thermal simulation. But due to computational cost, the focus would be mostly on static structural problem, and coupled simulation will be limited to the effect of plastic deformation on material temperature.

## 1.1 Hypothesis

Springback compensation values and wire motion, can be derived directly from FEA simulation.

### 1.2 Assumptions

- Machine precision is within the accepted tolerance threshold.
- Underling mathematical formulation of numerical model corresponds to real physical problem.
- Correct hardening model of bent material.

## 2 Structural FEA Simulation

Wire bending problem could be solved as either an explicit dynamic or an implicit static problem. However, a static simulation was chosen for better stability and easier convergence. ANSYS academic 2020 R1 software is used in this study due to its intuitiveness and academic support. It is worth to mention that ANSYS academic license is free with a 521k DOF simulation limit.

### 2.1 Simulation hardware and software setup

The main computer used in modelling and simulation is a MacBook Pro (Retina, 15-inch, Late 2013) with a 2,3 GHz Quad-Core Intel Core i7, 16 GB 1600 MHz DDR3, running a macOS Catalina version 10.15.4 and a virtual machine VMware Fusion professional version 11.5.1 with a Microsoft Windows 10 installed.

### 2.2 Simulation non-linearity

Due to springback problem complexity, FEA software has to be able solve non-linear problems concurrently.

#### 2.2.1 Geometric non-linearity

Due large deflections, wire cross-section is not perpendicular to bending axis. Hence, transverse shear stresses are not negligible, which leads to a non-linear reduction in wire stiffness.

#### 2.2.2 Material non-linearity

The main goal of steel cold bending process in this study, is to have a permanent wire deformation. Which means material has to undergo plastic deformation. And the relation between stress and strain in plastic area is non-linear (Fig. 5c).

#### 2.2.3 Contact non-linearity

Although load is constant on contact area between wire and circular die head, contact area is not constant; it changes based on wire deformation around the Die, Which causes non-linear changes in stress values.

#### 2.2.4 Thermoplasticity

According to Tresca, Taylor and Quinney, metals subjected to large plastic deformation experience noticeable heating. 73% to 94% of plastic work is converted to heat [4]. which affects both material elasticity and strength non-linearly. In general ferrous-alloys yield strength increases, since they reach maximum strength

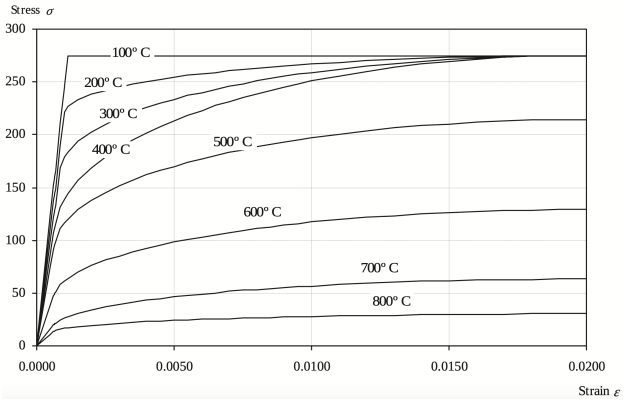



**Figure (13):** *Stress-strain relationship for S275 carbon steel at elevated temperatures.*

around 200°Celsius [5]. But for example, in our case, standard structural steel S275 both yield strength and elasticity modulus decreases under temperature increase(Fig. 13) [5]. It is worth to mention that plastic induced heat is calculated in relation to stain rate mm/mm/s , which considers time variable as a part of the equation.

### 2.3 Geometry

Bending pin, wire and bending head geometry was modelled using Ansys SpaceClaim modeller. All objects are 3d solids. All except the wire were changed to rigid bodies to reduce simulation time. All rigid bodies were fixed with a fixed joints to ensure system stability.

### 2.4 Meshing

Meshing is the backbone of FEA simulation. Unfortunately, minimum efforts were spent on meshing at the beginning of this study due to researcher's lack of experience in FEA modelling. However, an automatic coarse meshing was used to mesh wire object and contact surfaces which created 5058 nodes and 981 elements [6]. Mesh elements are divided into two groups:

- Solid elements: HEX20 (Wire) which is a SOLID187 type.
- Surface elements: Tri6 and Quad8 (Contact surface) which are CONTA174 and TARGE170 types.

However, for thermal coupling SOLID187 type has to be changed to SOLID227 for additional temperature degree of freedom.

### 2.5 Material properties

The used material in this study is a 3mm diameter Low carbon steel welding rod for oxy-acetylene welding. Mechanical properties were not provided by the manufacturer. However, according to DIN EN 12536 there was only three grades for oxy-acetylene welding materials, O I, O II and O III respectively. The price of used material

[5] https://www.engineeringtoolbox.com/metal-temperature-strength-d_1353.html

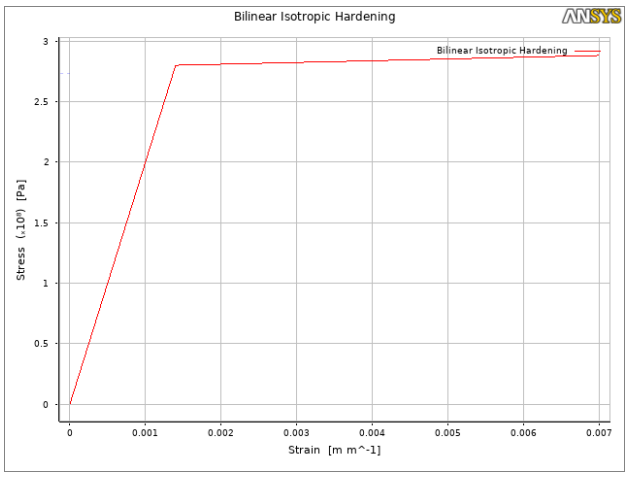



**Figure (14):** *Isotropic bilinear hardening model.*

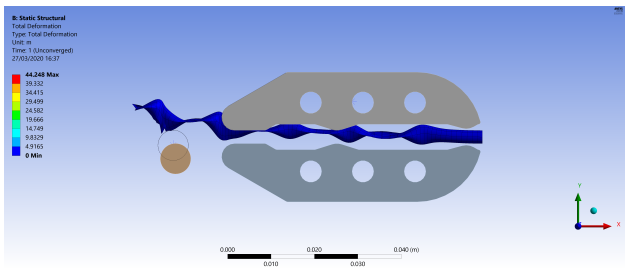


**Figure (15):** *Mesh distortion due to wrong contact settings.*

matches market price for O I grade which has a comparable yield strength as standard S275 steel. Hence, a non-linear standard steel material profile was chosen for the purpose of this simulation from the standard engineering material library in ANSYS. Only Yield strength and ultimate tensile strength were changed to 280 MPa and 400 MPa respectively to match manufacturer data. Standard Bilinear isotropic hardening model was chosen for simplification (Fig. 14).

### 2.6 Structural boundary conditions

Bending head is fixed to ground with a fixed joint contact. The wire is fixed from one end with a fixed support. However, setting contact surface between wire, bending pin and bending head was the most difficult part of this simulation. A time-consuming process of trial and error to find a proper setting was not avoidable (Fig.15). The final settings were a frictionless contact type, augmented Lagrange formulation, nodal-normal to target detection method and a 0.1 normal stiffness factor with update each iteration. With those settings I was able to run simulation smoothly for the 11 calibration angles.

#### 2.6.1 Solver settings

Although it is a non-linear problem, Direct solver could be used to solve equations iteratively through time steps for more accurate results. However, program controlled iterative solver (Newton-Raphson) was chosen due to memory limitations.
Action load is a rotational joint load applied on bending head with 5, 10, 15, 20, 30, 45, 60, 75, 90, 110, 130 degree respectively. However, each load has an additional step for unloading. Hence, 11 simulation were created for each bending angel with 2 steps solution.
Large deflection was turned on to activate non-linear solver and the rest of the settings was left for program control to minimize sub steps and consequently convergence time [6].

## 3 Coupled structural-thermal simulation

A simplified coupled static structural- thermal simulation was carried out to study the increase of wire temperature due to plastic heat generation.
A major benefit of thermal simulation would be the ability to measure physically the difference in material temperature during bending action with the help of FLIR (Forward looking infrared )cameras (Fig. 17) [7], which adds another method of results verification and process control.

### 3.1 Material thermal properties

Most of material thermal properties can be changed directly from engineering data in Ansys workbench. Although, the might get filtered according to solution type and some are only accessible from APDL. The related material properties are:

- Density = 7850 kg/m$^3$.
- Coefficient of thermal expansion = 1.2E-5 1/C.
- Isotropic thermal conductivity = 60.5 W/m/C.
- Specific heat = 434 J/kg/C.
- Taylor-Quinney coefficient (QRATE) = 0.9.

### 3.2 Thermal boundary conditions

- Initial analysis temperature = 22°.
- wire surface temperature = 22°.
- Convention load with heat transfer coefficient = 54E-5 W/mm$^2$/C.

### 3.3 APDL command approach

APDL is Ansys scripting language, could be used within user interface to have an access to most of program commands. In this case, APDL was used to change mesh elements from structural SOLID 187 to coupled elements SOLID227, to add Taylor-Quinney coefficient to wire material properties and finally to set convention load and thermal boundary conditions[8].Those modifications were applied with help of APDL snippet (Source code.2).

```
/PREP7
ET,matid,226,11 ! Chaning Solid structurel
↪  element to coupeld element and
↪  incrreasing DOF keyoption.
MP,QRATE,matid,0.9 ! Tylor-Quinney
↪  coefficient
/SOUL
IC,part_1,temp,22    ! solution initial
↪  thermal condition
D,Outer_conv,temp,22 ! Wire surface thermal
↪  bouandry condition
ALLSEL
SF,Outer_conv,CONV,ARG1,ARG2   ! Surface
↪  convection load
TIMINT,ON,THERM
TIMINT,ON,STRUCT
OUTRES, all, all
```

**Source code (2):** *APDL snippet in Ansys mechanical workbench interface to enable thermal transit coupling with static structural simulation.*

### 3.4 Workbench approach

Another way of doing coupled simulation is by using workbench or Ansys mechanical interface to add a static coupled field analysis item to project tree (Fig. 16). Structural loads to be copied to the new analysis item, convection load and thermal boundary condition to be set respectively.

### 3.5 Conversion problem

Due to additional degree of freedom, a difficulty in solution conversion was noticed. To overcome this problem, mesh improvements was crucial. Face meshing was added to all contact surfaces to convert all triangular meshes to quad elements, which ensures a better alignment with wire Hexahedron mesh elements [9]. Although problem was solved, but it was realized that, additional meshing efforts at the beginning of the project would have saved a considerable amount of time during simulation.

## 4 Structural results

### 4.1 Data extraction

There was no direct method to extract bending angels directly from ANSYS graphical interface. Instead, Cartesian coordinates of two points, with a fixed distance, was extracted. However, to avoid discrepancies at 0 and 90 degrees, arctan2 function was used to calculate angles (Fig. 18). The process was automated by using below python function (Code. 3)

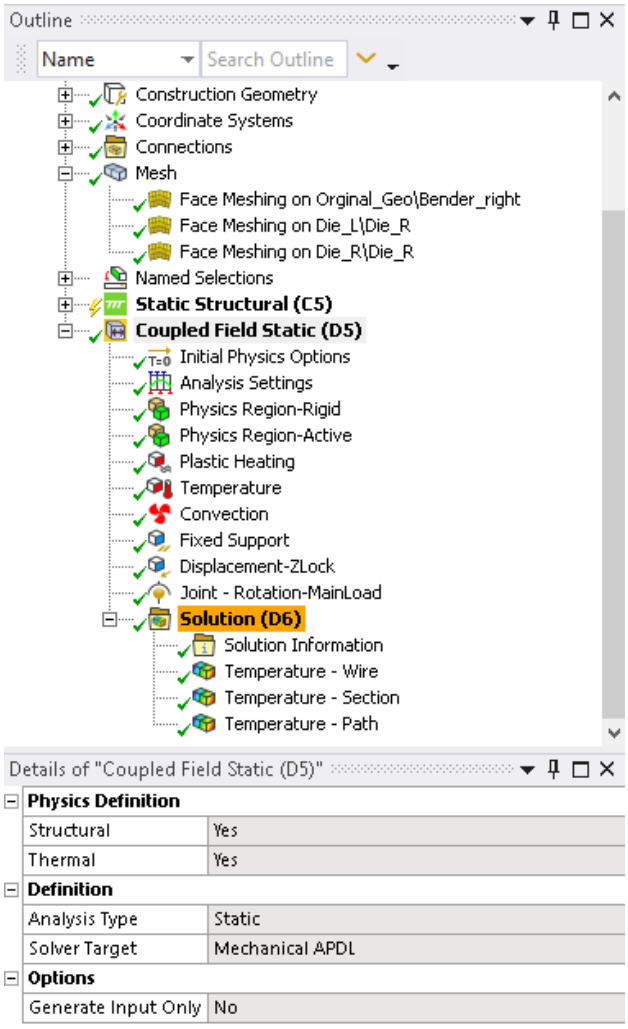



**Figure (16):** *Ansys Mechanical, Coupled static filed analysis setup.*

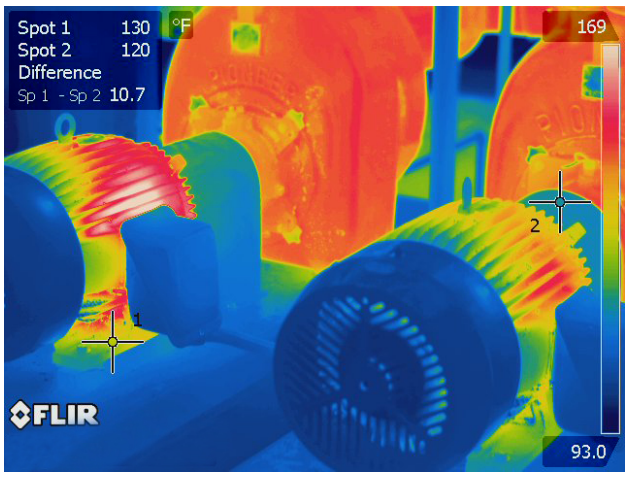



**Figure (17):** *FLIR thermal imaging example, shows that difference in object temperature can be measured by setting multiple scopes.*

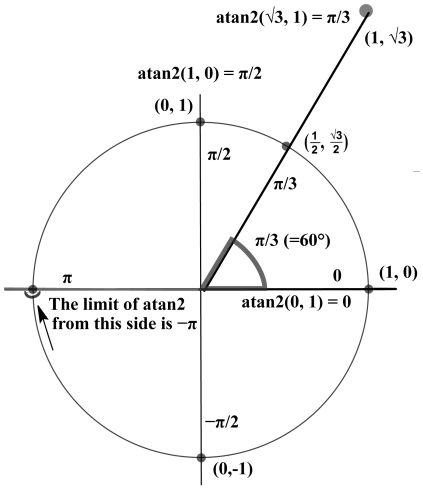



**Figure (18):** *arctan2*

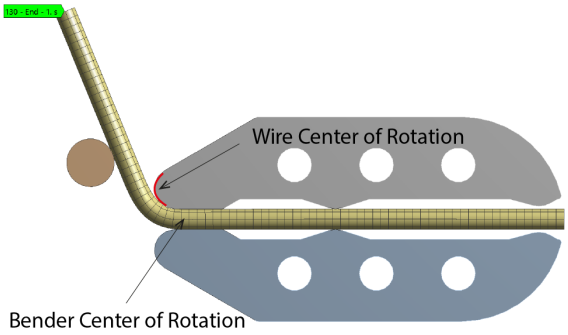
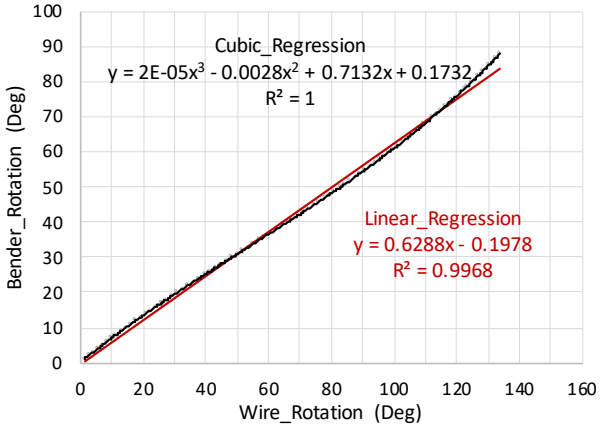


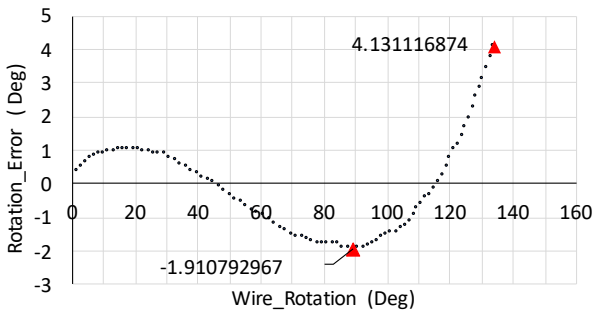

**Figure (19):** *Different centres of rotation between bending pin and wire motion.*

```
def slope_points_rad(x1, x2, y1, y2):
# normal arctan function dose not return
↪  signed values.
# move lines to origin by calculating delta.
dx = x2 - x1
dy = y2 - y1
# Get all signed slopes by using arctan2
↪  function.
slopes = np.arctan2(dy, dx)
# vectorised array operation to get the
↪  change in slope between all lines
↪  (Springback).
slope_1 = slopes[0:-1]
slope_2 = slopes[1:]
springback = slope_1 - slope_2
return springback
```

**Source code (3):** *Python function to calculate signed angels using extracted Cartesian coordinates from Ansys .*




**(a)** *Wire Rotation Function (Simulation).*




**(b)** *Residual plot of linear rotation (Simulation).*

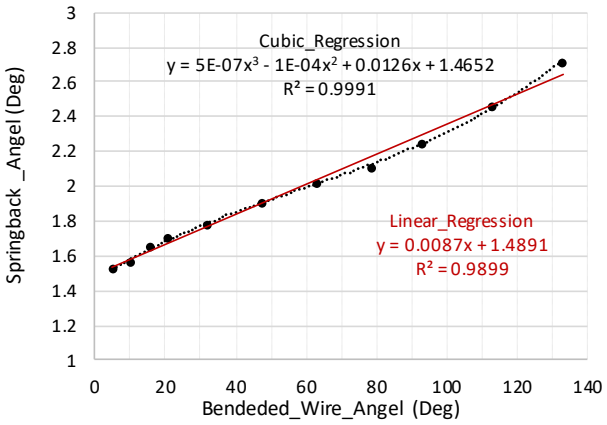
**Figure (20):** *Wire motion results which are extracted from FEA simulation.*

### 4.2 Wire motion

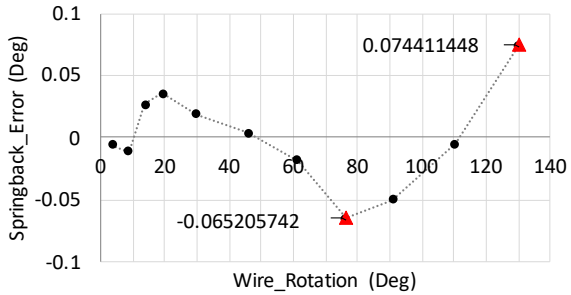
When we look at wire motion graph, we notice that there is almost linear relation between wire rotation and bender rotation , moreover the function statistically fit linearly with $R^2$ value of 0.9968 (Fig.20a) However, when we considering residual plot, linear motion function would lead to an error between -1.9 and 4.1 degrees (Fig.20b). In addition, residuals plot also explains clearly wire motion function; as wire slope changes non-linearly between 0° and 120° due to tangential rotation around bending head, and becomes linear afterwards due to wire rotation around the edge point of bending head.

### 4.3 Springback

Springback simulation graph shows also that, there is an almost linear relation between bending pin rotation and springback value (Fig.21a). It also statistically fit linearly with $R^2$ value of 0.9899. However, it was notice-




**(a)** *Springback function (Simulation)*




**(b)** *Residual plot of linear Springback (Simulation)*

**Figure (21):** *Residual plot Springback values which are extracted from FEA simulation.*

able that springback values has a relatively smaller magnitude comparing to wire motion function; they are between 1.5 and 2.7 degrees for wire rotations between 5 and 130 degrees respectively, which is under the acceptable tolerance of $\pm 1$ ° (Fig.21b).

### 4.4 Equivalent stress (von-Mises)

Study scope is a section in bent wire at maximum stress value (Fig.22). During loading phase, most of wire section is above material's both yield and ultimate strength, which are 280 MPa and 450 MPa respectively, except horizontal centre part, which stays within elastic deformation limits. The upper part shows also additional stresses, which are due to reaction forces from bending head (Fig.23a).
After unloading, a residual stresses are stored in bent wire, as parts of wire section deformed elastically and trying to springback to their initial location.

### 4.5 Equivalent strain

Looking at equivalent strain in wire section (Fig. 24a), we can notice that the top part of the section undergoes higher equivalent plastic stain levels (assumedly compression) due to additional reaction forces from bending head contact. However, when we study equivalent total strain over wire centreline, we find almost 11% partial elongation (Fig. 24b).

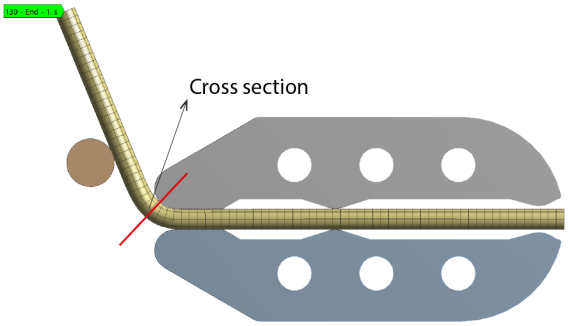
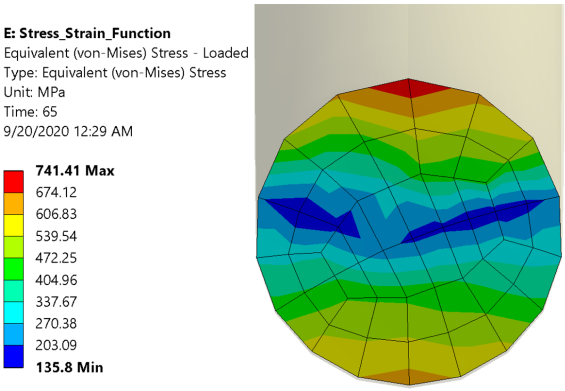
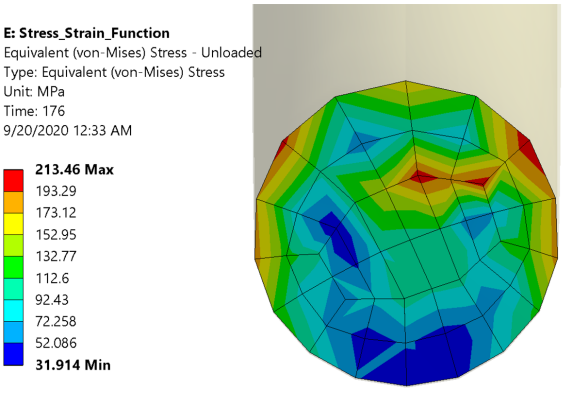



**Figure (22):** *Results scope, a cross section in bent wire at maximum stress-strain values*

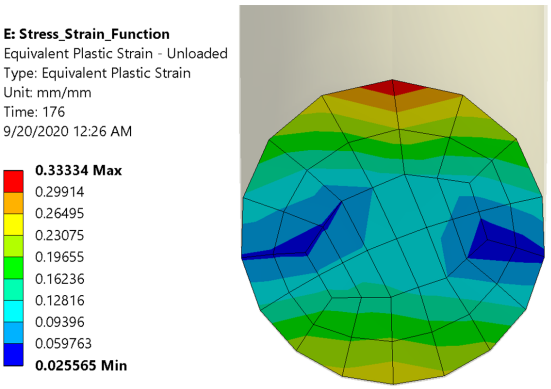



**(a)** *Loaded state, equivalent stress (von-Mises).*

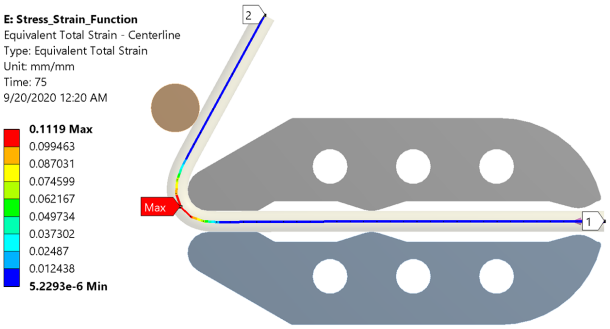



**(b)** *Unloaded state, equivalent stress (von-Mises).*

**Figure (23):** *Equivalent stress (von-Mises) results.*

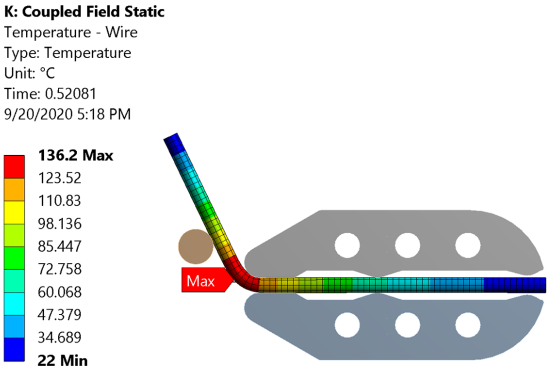



**(a)** *Equivalent plastic stain levels - final state.*

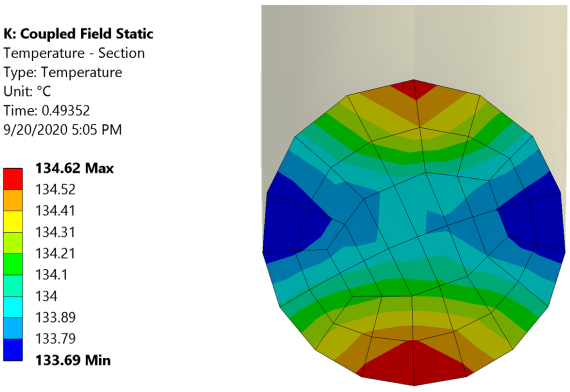



**(b)** *Equivalent total strain over wire centreline.*

**Figure (24):** *Equivalent strain analysis.*




**(a)** *A significant increase in wire temperature during bending action.*




**(b)** *Wire cross section shows heat distribution.*

**Figure (25):** *Thermal analysis results*

## 5 Thermal analysis results

Coupled field thermal analysis shows a significant increase in wire temperature during bending action with a max value of 136.2°C (Fig. 25a). Although, heat is generated from plastic deformation, but we can see gradual heat decrees due to heat flow though wire material.
In cross section (Fig. 25b), temperature levels reflects strain intensity seen in (Fig. 24a), except that, the lower part of the section has a higher temperature. That is because the outer radius of the wire is affected more by strain in length (stretched), therefore it has more tendency for heat accumulation.

## 6 Intermediate conclusion

It was clear in this case that data interpretation of motion function had a bigger influence over springback effect in producing bending error.
In addition, it is difficult to determine the exact effect of heat over springback values without an accurate material properties and exact load speed from empirical experiment. As both yield strength and elasticity modulus will change relative to temperature increase. On the other hand, bending speed will affect material's strain rate, convection rate and heat flow through wire.
However, for a more precise springback simulation, the following points should be considered in future FEA analysis:

- An accurate material mechanical properties.
- improved mesh.
- accurate contact settings.
- considering the effect of plastic heating on material behaviour.

# Part III

# Empirical Approach

## 1 Introduction

The aim is to eliminate human measurement error from empirical data to improve data curve fitting results. Computer vision system is proposed to automate measurement process in a precise manner. However it has its own limitations.

### 1.1 Hypothesis

Parts can be measured automatically and precisely by using a computer vision system. Opposing to numerical approach, this hypothesis accounts for machine imprecision.

### 1.2 Assumptions

- All measured parts are in one plane parallel to camera plane.
- There is a visual distinction between measured parts an background.
- Measured parts are not visually deformed by high reflections and hard shadows.

## 2 Methodology

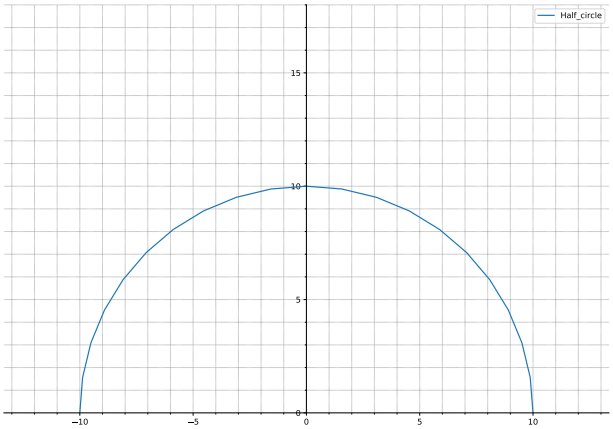
### 2.1 Experiment

The experiment is basically bending two curves with different curvature. The first is a half circle with a 10 cm radius, divided into 20 cord segments and a repetitive bending angle of 9° (Fig. 26a). The second curve is a saddle curve with a 20 cm width and 2 cm height, divided into 13 cord segments and a repetitive bending angle of 3.48° (Fig. 26b). However, bending process has to follow two conditions:

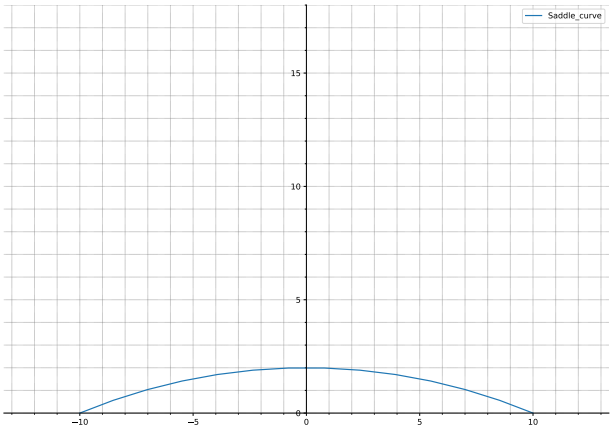
- Automated calibration with the help of computer vision.
- Achieve accepted tolerance, which is a maximum of 10% relative error and $\pm 1°$absolute error.

Additionally, in order to carry out bending experiments, the following steps has to be accomplished:

- Controlling bending machine outside the default application environment.
- Controlling camera remotely from computer.
- G-code compilation for bending geometry.
- Measuring both bending pin movement and wire motion during bending process.
- Springback compensation values calculation (data fitting).




**(a)** *A half circle with a 10 cm radius, divided into 20 cord segments and a repetitive bending angle of 9°.*




**(b)** *A saddle curve with a 20 cm width and 2 cm height, divided into 13 cord segments and a repetitive bending angle of 3.48°.*

**Figure (26):** *The geometry of bending experiments*

**Figure (27):** *Experiment's hardware setup.*

### 2.2 Experiment setup

The main hardware used in experiment is a Nikon D3100 DSLR[6] camera 4k (4,608 × 3,072)14.2 megapixel with 18-55mm f/3.5-5.6 Lens, 2010 model witha 1.25 meter tripod stand (Fig. 27). Software was OpenCV-contrib-python[7] version 4.4.0, a wrapper package for OpenCV python bindings. Additionally, gphoto2[8] python library is used to remotely control camera from computer (Source code 4).

### 2.3 Computer vision limitations

Industrial High precision measurement computer vision applications usually use a dual RGB camera systems or RGB-D (D for depth channel) camera sensors. They are mostly expensive and come with a proprietary software application.

In this study, the goal is to use the readily available single RGB camera to create a computer vision based measuring system which satisfies the accepted tolerance limits in (Section 6).

In this regards, an understanding of computer vision limitations is important.

[6] https://www.nikon.de/de_DE/product/discontinued/digital-cameras/2015/d3100

[7] https://pypi.org/project/opencv-contrib-python/

[8] https://pypi.org/project/gphoto2/

```python
import logging
import os
import gphoto2 as gp

def imgcapture_loop(i):
# Manage exceptios
logging.basicConfig(format='%(levelname)s:
↪  %(name)s: %(message)s',
↪  level=logging.WARNING)
callback_obj =
↪  gp.check_result(gp.use_python_logging())

# Create camera object
camera = gp.Camera()
camera.init()

# Create capturing loop
for j in range(i):
print(f'Capturing image:{j + 1}')

# Capture images and get image name
file_path =
↪  camera.capture(gp.GP_CAPTURE_IMAGE)
print(f'Camera file path:
↪  {file_path.folder}/{file_path.name}')

# Prepare target path
cwd = os.path.join(os.getcwd(),
↪  'fromCamera')
print(cwd)
target = os.path.join(cwd, file_path.name)
print(f'Copying image {j + 1} : to', target)

# Copy images from camera to target folder
camera_file =
↪  camera.file_get(file_path.folder,
↪  file_path.name, gp.GP_FILE_TYPE_NORMAL)
camera_file.save(target)

camera.exit()
print('done')
```

**Source code (4):** *Python function to control Nikon d3100 camera remotely, the function captures images and copies it to project folder for a later computer vision processes.*

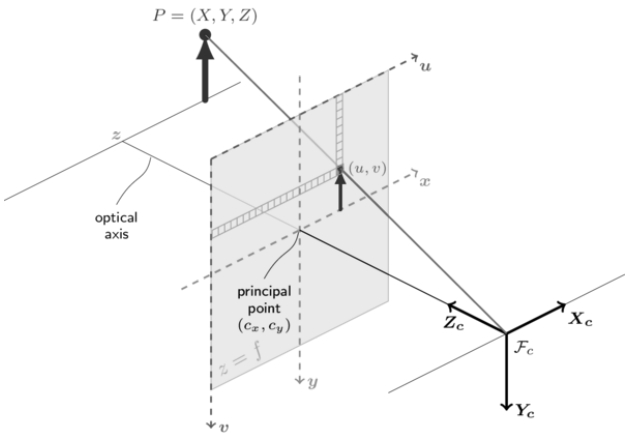



**Figure (28):** *Pinhole camera model, shows the relation between real space and homogeneous coordinates.*

### 2.3.1 Homogeneous coordinates

When we take a picture of a 3d real space, Z dimensional values will be flattened (Fig. 28). Additionally, in projective space, all parallel lines meet at infinity point ($\infty$, $\infty$) , that is the reason why an additional $w$ scaler depth variable is considered in addition to ($x$, $y$) coordinates. in other words, a picture is actually a 2d Homogenous coordinates representing the 3d Cartesian space and all geometrical calculations should follow projective geometry rules.

The immediate problem is that, nearby objects will have a bigger dimensions than distant ones, even if they are exact in size, as they have a different depth $w$ value.

To avoid this issue, all measured parts has to be in one plane facing camera plane, in this case $w$ becomes a constant value which could be factored out.

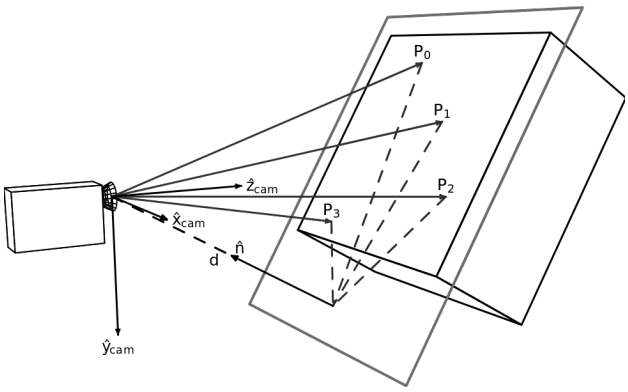
### 2.3.2 Non-affine homography

If camera plane is not parallel to measurement target plane (Fig. 29a), Image will get a non-affine projective distortion, which could be solved by general homography algorithm (mapping). But it ia an additional interpolation step which could effect image quality and increases computational cost.

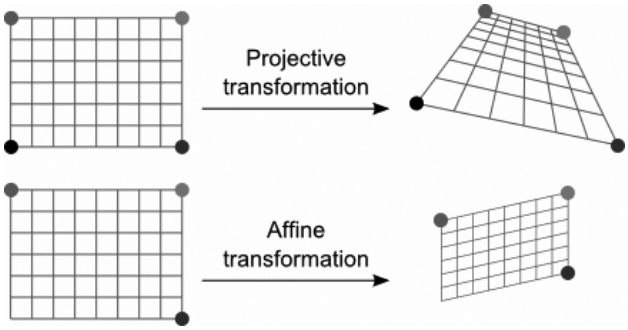
However, Non-affine homography can be avoided, if camera plane is parallel to measurement target plane. In this case, we will use only affine transformations to address rotation and scale issues if needed (Fig. 29b).

### 2.3.3 Camera distortion

**Radial distortion:** the smaller focal length is, the more light bends at the edge of the lens, which usually leads to barrel shape (Fig. 30a).
**Tangential distortion:** Due to imperfections, camera sensor or lens could be slightly inclined, which leads to a slight perspective distortion (Fig. 30b).




**(a)** *When Camera plane is not parallel to measurement target plane, we get a projective distortion.*




**(b)** *This diagram shows the difference between affine and non-affine transformation.*

**Figure (29):** *Non-affine homography (Mapping).*

Nonetheless, camera distortion can be reduced by:

- Using a longer focal length lens and or,
- Keeping measured parts near optical centreline and or,
- Running camera calibration routine.

### 2.3.4 Lighting conditions

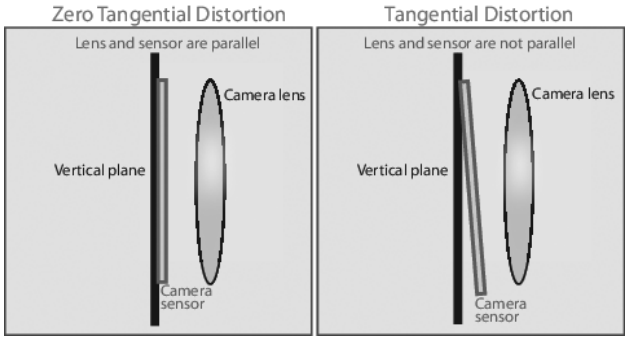
Light intensity and temperature, are two important factors which affect image quality. High light intensity can cause image burn out and increases both reflections and hard shadows, while low light intensity increases image noise and destabilises camera's auto focus option.

Light temperatures over 5000 K are called "cool colors" (bluish), while lower color temperatures (2700–3000 K) are called "warm colors" (yellowish). In a clear sky daylight, light temperature changes form sunrise (warm) to midday (cool) and again to sunset (warm). whereas, in indoor lighting light temperature depends on light appliances. Light temperature can affect colour detection, as they alter both blue and yellow colour content in any image.

### 2.3.5 Surface reflections

Unfortunately, most of metals are not only have a high reflectance ratio, but also prone to corrosion. Which increases the probability of having either uneven or a high


**(a)** *Positive and negative radial distortion.*




**(b)** *Tangential distortion due to camera imperfection.*

**Figure (30):** *Types of camera distortion.*

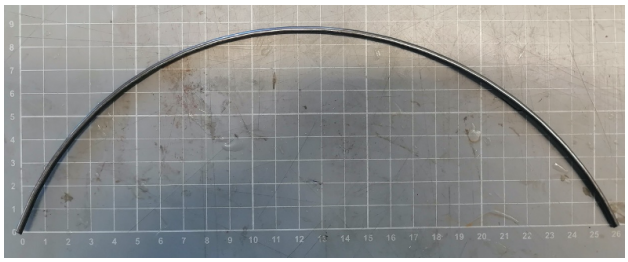


**Figure (31):** *A worst case scenario example, used for feature detection experiments, bent wire colour is similar to back ground colour, hard self shadows, uneven surface reflection, random soft shadows and noisy background.*

reflective surface (Fig. 31). Both cases undermine shape and colour detection algorithms.

# 3 Feature detection

Considering computer vision limitation (Section 2.3), the goal of feature detection, is to systematically identify and measure both bent wire angles and bending pin rotation (Fig. 31).

## 3.1 Canny edge detection

Canny edge detection method ia a common multi step algorithm, used to detect edges in an image by finding the intensity of change in pixel values. In OpenCV, canny edge detector has a simple single line interface (Source code 5), although it compute internally the following algorithms:

- **Gaussian blur:** is used to filter out image noise.

```
import cv2
# Read image as gray.
img = cv2.imread('image.png', 0)
# Canny detector with min/max thresholds.
edges = cv2.Canny(img, 254, 255)
# Display function.
cv2.namedWindow('Canny', cv2.WINDOW_NORMAL)
cv2.imshow('Canny', edges)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

**Source code (5):** *Python code for canny edge detector.*

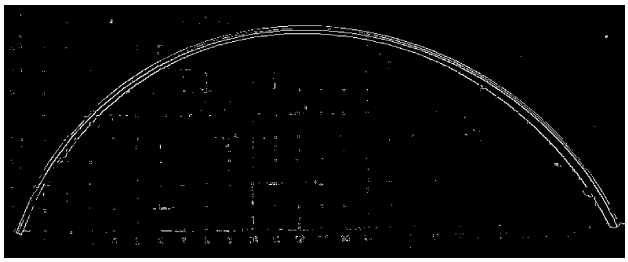


**Figure (32):** *Canny edge detector managed to isolate wire edges from background at 254, 255 hysteresis thresholds.*

- **Soble filter:** The core component. It emphasises edges by approximating the intensity of change in pixel values (gradient).
- **Non-maximum suppression:** This removes pixels that are not considered to be part of an edge. Hence, only thin lines (candidate edges) will remain.
- **Edge hysteresis control:** A lower and an upper threshold limits are manually set to distinguish between week and strong edges.

Although canny edge detector detects wire's shape, but it returns only pixel information as a binary image (Fig. 32 ). Which is not suitable for any geometrical calculation. Hence, additional feature detection algorithm is required.

## 3.2 Hough line transform

In contrast to canny edge detector, Hough line transform returns lines in a mathematical form. In OpenCV There are two types of Hough line transformation; standard and probabilistic line transformation.

### 3.2.1 Standard Hough line transform

It takes grey image or preferably canny edges, generate random set points on all edges then runs a few different sloped lines through each point. Then calculates the minimum distance between each line $\rho$ and origin point $r = x\cos\theta + y\sin\theta$ . Since points on the same line will result the same slope $\theta$ and minimum distance $\rho$ , the algorithm will list all lines and check the maximum repeated lines, which goes through maximum number of

```
import cv2
import numpy as np
# Read image as coloured.
img = cv2.imread('image.png')
# Convert colour space to grey.
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
# Find edges.
edges = cv2.Canny(gray, 250, 255)
# Standard Hough line transform.
lines = cv2.HoughLines(edges,1,np.pi/45,50)
# Draw standard Hough lines.
for rho, theta in lines[:, 0]:
a = np.cos(theta)
b = np.sin(theta)
x0 = a * rho
y0 = b * rho
x1 = int(x0 + 1000 * (-b))
y1 = int(y0 + 1000 * (a))
x2 = int(x0 - 1000 * (-b))
y2 = int(y0 - 1000 * (a))
cv2.line(img,(x1,y1),(x2,y2),(0,0,255),1)
# Display function.
cv2.namedWindow('Hough',cv2.WINDOW_NORMAL)
cv2.imshow('Hough', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

**Source code (6):** *Python code for standard Hough line transform.*

points (votes) , discard single point lines and then returns lines in the form of $(\rho, \theta)$ (Fig. 33).
The problem with this method that it dose not return the start and end points of line segments (Source code 6), and dose not work well curves (Fig. 34).

### 3.2.2 Probabilistic Hough line transform

It is very similar standard Hough line transform, but we can specify maximum segments length and minimum gap between segments (Source code 7). it returns both start and end points, but unfortunately, it did not recognize complete arc segments as well (Fig. 35 ).

## 3.3 Contour finding method

Contour finding algorithm in OpenCV ensures a closed boundary around features. It takes binary images or canny edges, prevent all high values pixels from (white) from becoming boundary pixels, draws a boundary on the most outer image's pixels and then shrinks down towards those white pixels until distance is minimum. It returns a list of contours, each has an array of boundary points (Source code 8). However, due to noisy background and hard shadow, it could not define the exact shape of the arc (Fig. 36). Additional strategy was required to isolate only the top curve of the arc, since it is cleaner due to current lighting conditions.

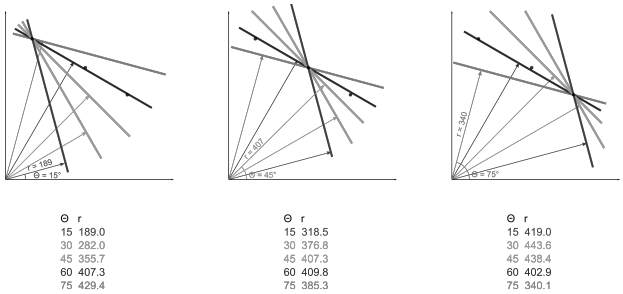



**Figure (33):** *Hough line transformation, algorithm will list all lines and check the maximum repeated lines, which goes through maximum number of points.*

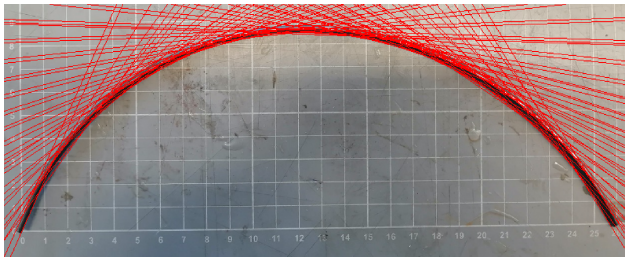


**Figure (34):** *Standard Hough line transform, using canny edges as input, dose not return the start and end points of arc segments.*

```
import cv2
import numpy as np
# Read image as coloured.
img = cv2.imread('image.png')
# Convert colour space to grey.
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
# Find edges.
edges = cv2.Canny(gray, 254, 255)
# probabilistic Hough line transform.
lines = cv2.HoughLinesP(edges, 1, np.pi /
↪  720, 5, 10, 5)
# Draw probabilistic Hough lines.
for x1, y1, x2, y2 in lines[:, 0]:
cv2.line(img,(x1,y1),(x2,y2),(0,0,255),2)
# display function.
cv2.namedWindow('Hough', cv2.WINDOW_NORMAL)
cv2.imshow('Hough', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

**Source code (7):** *Python code for probabilistic Hough line transform.*

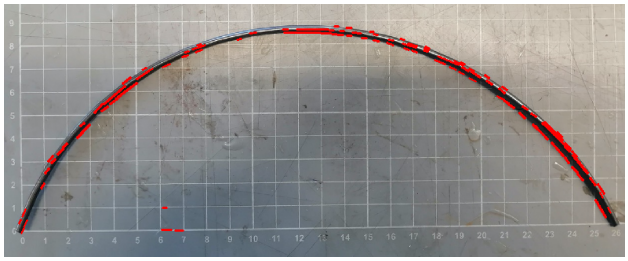


**Figure (35):** *Probabilistic Hough line transform, did not recognize arc segments.*

```
import cv2
import numpy as np
# Read image as coloured.
image = cv2.imread('image.png')
# Convert colour space to greyscale.
gray = cv2.cvtColor(image,
↪  cv2.COLOR_BGR2GRAY)
# Find edges (Optional) or,
edged = cv2.Canny(gray, 250, 255)
# Create Binary image filter.
ret, thresh = cv2.threshold(gray, 100, 200,
↪  cv2.THRESH_BINARY_INV)
# Find all contours.
contours, hierarchy =
↪  cv2.findContours(thresh, cv2.RETR_LIST,
↪  cv2.CHAIN_APPROX_SIMPLE)
# Find index of wire's contour.
for i, k in enumerate(contours):
txt = k.shape[0]
if k.shape[0] > 100:
print(i, txt)
# Make a copy of the image.
image_cont = image.copy()
# Draw wire contour only.
cv2.drawContours(image_cont, contours, 780,
↪  (0, 0, 255), 2)
# Display function.
cv2.namedWindow('Contours',
↪  cv2.WINDOW_NORMAL)
cv2.imshow('Contours', image_cont)
cv2.waitKey(0)
```

**Source code (8):** *Python code for Contour finding algorithm.*

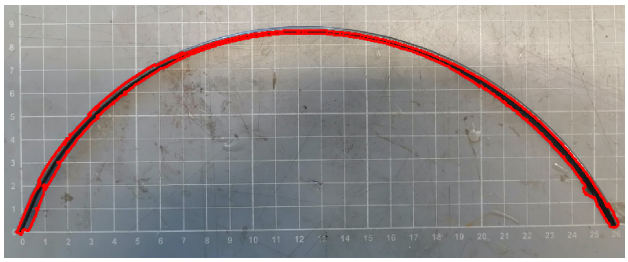


**Figure (36):** *Contour finding algorithm in OpenCV ensures a closed boundary around features, however hard shadows were included in arc shape.*

```
# Calculate Convex hall from wire contour.
hull = cv2.convexHull(contours[780], False)
# Draw Convex hull function.
image_conv = image.copy()
cv2.drawContours(image_conv, [hull], 0, (0,
↪  0, 255), 2)
# Draw Convex hull points.
hull_t = np.vstack(hull).squeeze()
for i in hull_t:
cv2.drawMarker(image_conv,(i[0],i[1]),(0,
↪  255, 255), markerType=cv2.MARKER_CROSS,
↪  markerSize=15,thickness=2,
↪  line_type=cv2.LINE_AA)
# Display function.
cv2.namedWindow('Convex', cv2.WINDOW_NORMAL)
cv2.imshow('Convex', image_conv)
cv2.waitKey(0)
```

**Source code (9):** *Python code for Convex hull calculation.*

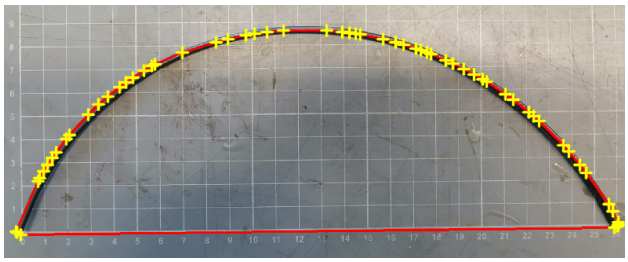


**Figure (37):** *Convex hull uses contour data to return a convex function (red), and points (yellow), a good approximation of wire shape.*

### 3.3.1 Convex hull

As contour line is a continuous function, we can consider the top part of contour as convex and the lower part as concave or vice versa (Source code 9). Convex hull function takes contour points and returns only the points which are convex or concave based on selected direction (Fig. 37).

## 3.4 Ellipse fit

OpenCV provides a quick ellipse fit function which draws a complete ellipse around a set of points, since we have an arc shape, so the drawn ellipse will simply go through the convex point set from the previous function (Fig. 38). The function returns a general ellipse parameters; centre point, rotation angle, major and minor axis lengths.

### 3.4.1 Finding arc's start and end points

One way to find start and end points is to sort all points for maximum $x$ and minimum $y$ for start point, and minimum $x$ and $y$ for end point. But, it works only for horizontal arcs.

However, if we draw a minimum enclosing circle around points, it will go through both arc's start and

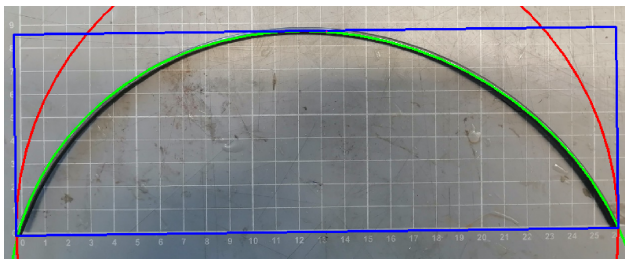


**Figure (38):** *(green) is ellipse fit function through convex hull points, (red) is a minimum enclosing circle around points, (blue) is a rotated rectangle fit function.*

end points, since maximum distance is between them (Fig. 38). Now we can define both start and end points by finding the intersection between the enclosing circle and ellipse (Source code 10). That can be done by using geometry module from Sympy, but the problem that ellipse function in Sympy dose not support th general form of ellipse [9], which means it supports only horizontal and vertical ellipses.

### 3.4.2 Bivariate quadratic equations solution

Another method to find intersection points between the circle and ellipse is to solve a bivariate quadratic equations square system. First, we have to expand general form circle and ellipse equations (Equations 4, 6) into (Equations 5, 7). Then, we can use for example, Scipy.optimize.fsolve function to solve equations with Newton-like method, which takes to inputs. The first input is a callable function in a vector-valued form, in this case, $f([x_1, x_2]) = [x_1^2 + x_2^2 + \alpha_0 x_1 + \alpha_1 x_2 + \alpha_2,\ \beta_0 x_1^2 + \beta_1 x_2^2 + \beta_2 x_1 x_2 + \beta_3 x_1 + \beta_4 x_2 + \beta_5]$. The second input is an initial guess, since maximum intersection between circle and ellipse is 4, we will need 4 points near intersections. Those points we can get them by running a rotated fit rectangle function from OpenCV (Fig. 38).

$$(x - h_1)^2 + (y - k_1)^2 = r^2 \tag{4}$$

Where:
$h_1, k_1 =$ circle centre point $(x, y)$, $r =$ circle radius.

$$x^2 + y^2 + \alpha_0 x + \alpha_1 y + \alpha_2 = 0 \tag{5}$$

Where:
$\alpha_0 = -2h_1$
$\alpha_1 = -2k_1$
$\alpha_2 = -r^2 + h_1^2 + k_1^2$

$$\frac{((x - h_2)\cos(\theta) + (y - k_2)\sin(\theta))^2}{(a^2)} + \frac{((x - h_2)\sin(\theta) - (y - k_2)\cos(\theta))^2}{(b^2)} = 1 \tag{6}$$

[9] https://docs.sympy.org/latest/modules/geometry/

```
# Fit ellipse through Convex hull points.
ellipse = cv2.fitEllipse(hull)
# Draw ellipse.
cv2.ellipse(image, ellipse, (0, 255, 0), 2)
# Calculate Minimum enclosing circle.
(x, y), radius =
↪  cv2.minEnclosingCircle(hull)
# Draw circle.
center = (int(x), int(y))
radius = int(radius)
cv2.circle(image, center, radius, (0, 0,
↪  255), 2)
# Calculate Minimum rotated rectangle.
rect = cv2.minAreaRect(hull)
# Draw rectangle.
box = cv2.boxPoints(rect)
box = np.int0(box)
cv2.drawContours(image, [box], 0, (255, 0,
↪  0), 2)
# Display function
cv2.namedWindow('Ellipse fit',
↪  cv2.WINDOW_NORMAL)
cv2.imshow('Ellipse fit', image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

**Source code (10):** *Python code for elliptical arc detection algorithm.*

Where:
$h_2, k_e =$ ellipse centre point $(x, y)$, $\theta =$ ellipse rotation angle, $a =$ semi major axis, $b =$ semi minor axis.

$$\beta_0 x^2 + \beta_1 y^2 + \beta_2 xy + \beta_3 x + \beta_4 y + \beta_5 = 0 \tag{7}$$

Where[10]:

$$\beta_0 = \frac{\cos^2(\theta)}{a^2} + \frac{\sin^2(\theta)}{b^2}$$

$$\beta_1 = \frac{\sin^2(\theta)}{a^2} + \frac{\cos^2(\theta)}{b^2}$$

$$\beta_2 = \frac{\sin(2\theta)}{a^2} - \frac{\sin(2\theta)}{b^2}$$

$$\beta_3 = -\frac{2h_2\cos^2(\theta)}{a^2} - \frac{k_2\sin(2\theta)}{a^2} - \frac{2h_2\sin^2(\theta)}{b^2} + \frac{k_2\sin(2\theta)}{b^2}$$

$$\beta_4 = -\frac{h_2\sin(2\theta)}{a^2} - \frac{2k_2\sin^2(\theta)}{a^2} + \frac{h_2\sin(2\theta)}{b^2} - \frac{2k_2\cos^2(\theta)}{b^2}$$

$$\beta_5 = \frac{h_2^2\cos^2(\theta)}{a^2} + \frac{h_2k_2\sin(2\theta)}{a^2} + \frac{k_2^2\sin^2(\theta)}{a^2} + \frac{h_2^2\sin^2(\theta)}{b^2} - \frac{h_2k_2\sin(2\theta)}{b^2} + \frac{k_2^2\cos^2(\theta)}{b^2} - 1$$

[10] https://math.stackexchange.com/questions/426150/what-is-the-general-equation-of-the-ellipse-that-is-not-in-the-origin-and-rotate

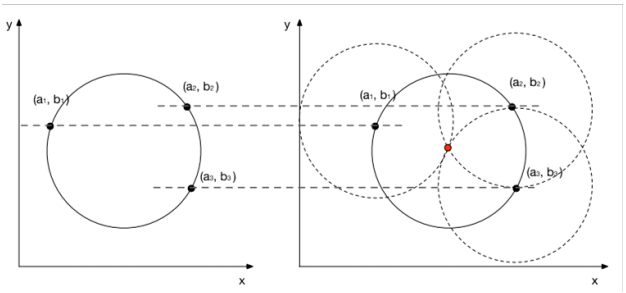



**Figure (39):** *Hough circle transform, draws a number of circles with a fixed radius on edge points, then calculate intersection points (red point).*

### 3.5 Approximate circular arc fit

For a less accurate solution but computationally efficient, we could take the width and height of the bounding rectangle (Fig. ) and consider them as a circular arc width and height respectively. Then use (Equations 8, 9,10) and methodology from (section 4). This solution would be optimal for a real time computation, whereas elliptical arc solution could be used for final results.

$$r = \frac{d^2}{8h} + \frac{h}{2} \tag{8}$$

$$\theta_{arc} = 2\sin(\frac{c^2}{2r}) \tag{9}$$

$$\theta_{bent} = \frac{\theta_{arc}}{n_{div}} \tag{10}$$

Where:
$r$ = arc radius, $d$ = arc width (rectangle width), $h$ = arc height (rectangle height), $\theta_{arc}$ = arch angle, $\theta_{bent}$ = segment bent angle and $n_div$ = number of divisions (cords).

### 3.6 Hough circles detection

The next task would be detecting the circular bending bin, for this Hough circle transform function is used. It uses the same concept of Hough line transform, but instead it draws a number of circles with a fixed radius on edge points, then calculate intersection points using general form circle equation $(x-h)^2+(y-k)^2=r^2$ where origin $x, y$ is 0 and $r$ is known (Fig. 39 ). The function returns a list of centre points and radii. It is a relatively stable function comparing to hough line transform (Fig. 40), and it also has an interface to control the minimum, maximum radius and the minimum distance between circles (Source code 11).

### 3.7 Assistive markers

As circle detection proved to have a better performance in general, and to have a consistency in feature detection methodology, an assistive colour coded marking system was introduced. Markers are 3D printed parts, which can be fitted to die, wire and pending pin (Fig. 41a). Colour coding is a printed coloured circles and glued to the surface of 3D printed markers. Colour coding helps with point identification and image colour based segmentation (Fig. 41b) .

```python
import cv2
import numpy as np
# Read image as Grayscale.
img = cv2.imread('DSC_0149.JPG', 0)
# Apply image blur to reduce noise.
img = cv2.medianBlur(img, 5)
# Detect circles with Hough circles
↪  transform.
circles = cv2.HoughCircles(img,
↪  cv2.HOUGH_GRADIENT, 1, 200, param1=50,
↪  param2=35, minRadius=30, maxRadius=33)
# Draw circles and center points.
circles = np.uint16(np.around(circles))
cimg = cv2.cvtColor(img, cv2.COLOR_GRAY2BGR)
for i in circles[0, :]:
# draw the outer circle
cv2.circle(cimg, (i[0], i[1]), i[2], (0,
↪  255, 0), 4)
# draw the center of the circle
cv2.circle(cimg, (i[0], i[1]), 2, (0, 0,
↪  255), 4)
# Display function
cv2.namedWindow("window",
↪  cv2.WINDOW_GUI_EXPANDED)
cv2.imshow('window', cimg)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

**Source code (11):** *Python code for Hough circles detection algorithm.*

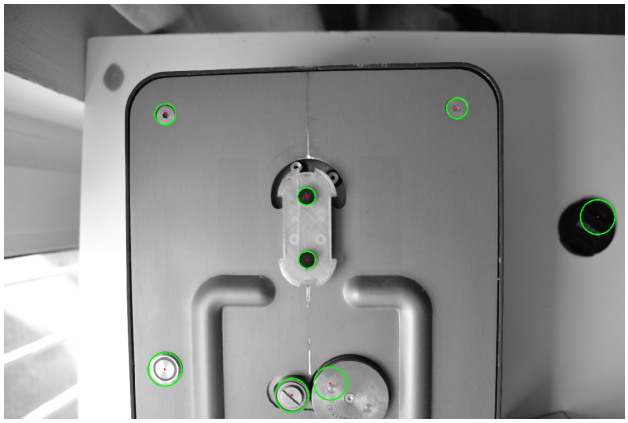
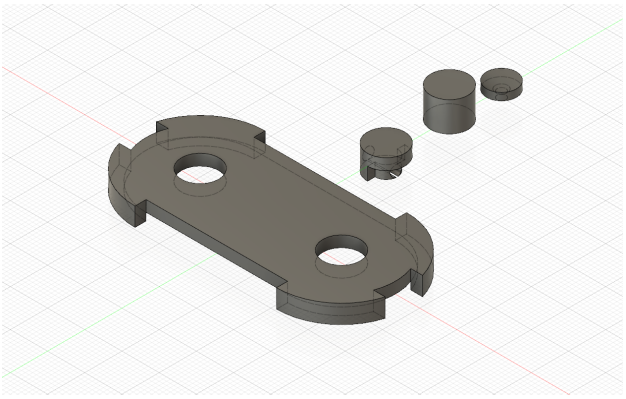
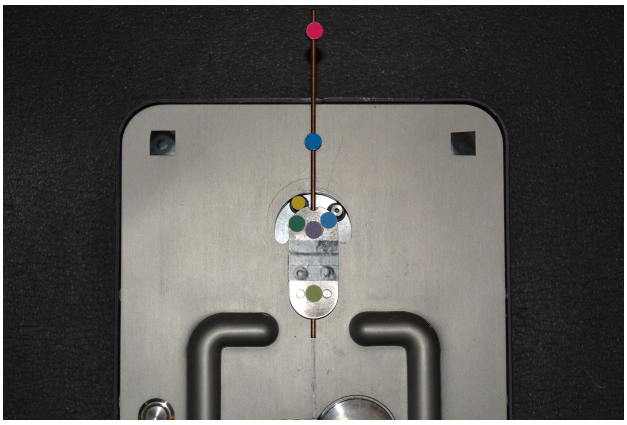


**Figure (40):** *Hough circle detection, (green) are detected circles, (red) are centre points, although image has an obvious radial distortion, the function was able to detect all complete circles.*



**(a)** *3D printed parts, from left to right: die (bending head) marker placeholder with tow holes to identify bending pin rotation centre and vertical axis, wire marker, die markers and bending pin marker.*



**(b)** *Installed markers, ready for measuring.*

**Figure (41):** *Assistive colour coded marking system.*

# 4 Colour based segmentation

The main reason of using colour based segmentation is to reduce computational cost be reducing the region of interest which will be processed by computer. Additionally, it increases the accuracy of feature detecting algorithms by isolating the parts which has to be detected.

## 4.1 General workflow

Work flow starts with colour coded markers preparation. Marker colours were designed in a CMYK colour space, so we can get almost the same print results with any printer uses the same printing technology, inkjet printing in this case. Markers were printed on two different paper types; matt and glossy, since glossy paper has a better colour saturation results than matt paper which is easier for colour segmentation, but matt paper has a better results in camera flash lighting condition.

Second step was to take photos of the markers in different lighting conditions; daylight, artificial lighting and camera flash lighting condition (Fig. 42a).

The third step was to split image's colour channels and manually identify colour threshold for each marker based on the chosen colour spaces, RGB and LAB in this case. This process was quit tedious and time consuming, it should be improved in future works (Fig. 42b).

The last step was to create colour filters for image segmentation and test feature detection algorithm on image sequence for verification (Source code 12).

## 4.2 RGB (BRG) thersholding

The RGB colour space is an additive colour space where colours are obtained by a linear combination of Red, Green, and Blue values.The three channels are correlated by the amount of light hitting the surface. which means any increase in light intensity will consequently lead to an increase in the three channels respectively (Fig. 42b. Although, the use of RGB colour space in image segmentation is computationally efficient, since it is the industry standard in storing images and dose not need any additional conversion or memory storage costs, but it has its own disadvantages:

- RGB colour space is very sensitive to any change in light intensity.
- it is sensitive to light temperature, since RGB has no a separate yellow colour channel.

## 4.3 LAB thersholding

The Lab colour space is quite different from the RGB colour space. In RGB colour space the colour information is separated into three channels but the same three channels also encode brightness information. Whereas, in Lab colour space, the L channel is independent of colour information and encodes brightness only. The other two channels encode colour. A channel encodes

colours from green to magenta and B Channel encodes colours from Blue to yellow. Control over both light intensity and yellow colour channel makes LAB colour space relatively resilient to changes in lighting conditions (Fig. 42c).

# 5 Computer vision measurements verification

In order to replace manual calibration with automatic computer vision process, we have to verify if computer is able to read a comparable measurements from photos or not. Hence, a later manual measurement process was carried out on bent wires from springback experiment (Fig. 43a).

The results shows that the difference between manual and computer vision measurements is between -0.6 ° and 1.7°, which exceeds the accepted tolerance. However, the difference in measurements between bender and wire is always below 1°, which is sufficient for springback calculation (Fig. 43b).

# 6 Data fitting

Data fitting, curve fitting, regression and interpolation, are terms which, are used in general to describe the operation of deriving new values from known data set[10]. However, regression term is mostly associated with statistics and machine learning applications for both data interpolation and extrapolation. On the other hand, interpolation term is used when interpolation function is meant to coincide with all data points, which means a higher order function to be computed.

The selection of data fitting method in this study was based on the following factors:

- Input data quality (observation's certainty data noise).
- Input data quantity (number of data points).
- Computational cost.
- Output data quality (accepted level of approximation).

Considering the aforementioned factors, two data fitting models were chosen, polynomial data fitting (cubic) and multi linear data fitting model.

## 6.1 Polynomial (cubic) data fitting model

In our case we have 11 known data points $(x_{bending\,angle}, y_{springback\,value})$ and they do not fall on one line. Which means that there is no exact linear function to describe the relationship between those points. In a mater of fact there is a very low chance of finding an exact non-linear function below a polynomial function of power 11. Hence, instead of $y_{exact}$ we would be looking for best fit $y_{fit}$ value with minimum residual error, which turns it to a least square problem.

Although $y_{fit}$ is a non-linear function in $f(x)$ (Equation 11), it is still a linear function in $f(\beta_n)$. Hence, it could be solved by solving an overdetermined system of linear equations, since we know that there is no exact solution. Which means the number of known data points has to exceed the number of unknown coefficients $\beta_n$. To compute this problem a python snippet was used (Source code 13) .

$$Y_{fit} = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 \tag{11}$$

Where:
$Y_{fit}$ = unknown springback value, $\beta_n$ = unknown coefficients and $X$ = desired bending angle (given).

## 6.2 Multi linear data fitting model

It is very similar to polynomial data fitting model but with a fewer unknown coefficients (Equation 12), and could be solved with the same python code (Source code 13).

But the main difference here is, it is not applied on data points at once, rather, multiple times, between every 2 data points. Although, a least square solution were used but that was only for continuity and code readability. A much efficient approach would finding $Y_{exact}$ be solving for $y$ intercept $(b)$ first, since slope is known $(m = \frac{\Delta y}{\Delta x})$ (Equation 13).

$$Y_{fit} = \beta_0 + \beta_1 X \tag{12}$$

Where:
$Y_{fit}$ = unknown springback value, $\beta_n$ = unknown coefficients and $X$ = desired bending angle (given).

$$Y_{exact} = b + mX \tag{13}$$

Where:
$Y_{exact}$ = unknown springback value, $b$ = $y$ intercept, $m$ = slope , $X$ = desired bending angle (given).

# 7 Serial communication

Communication with bending machine is possible from python with help of Pyserial library [11]. However, due to data flow control difficulties (Section 4.3) , G-code was compiled manually and sent through a cross platform serial communication server called Cool term [12].

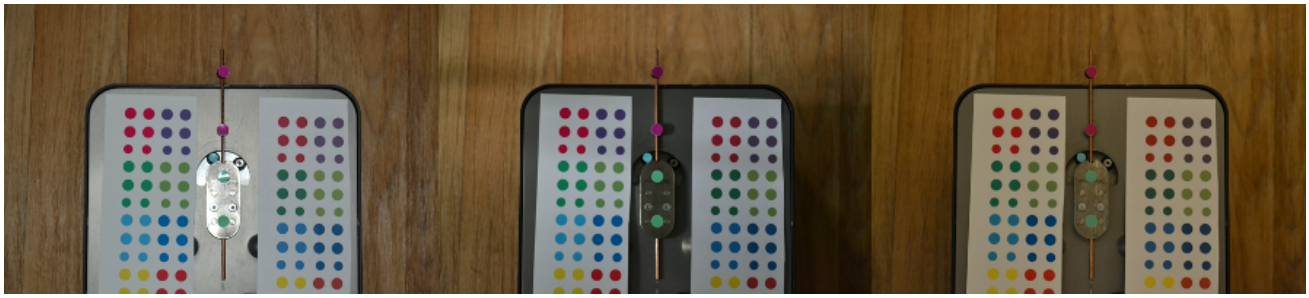
# 8 Results

The first bending experiment was done to establish baseline case (Fig. 44a). It was done through the default machine software following standard manual calibration process, and embedded cubic polynomial data fitting model for springback compensation. The

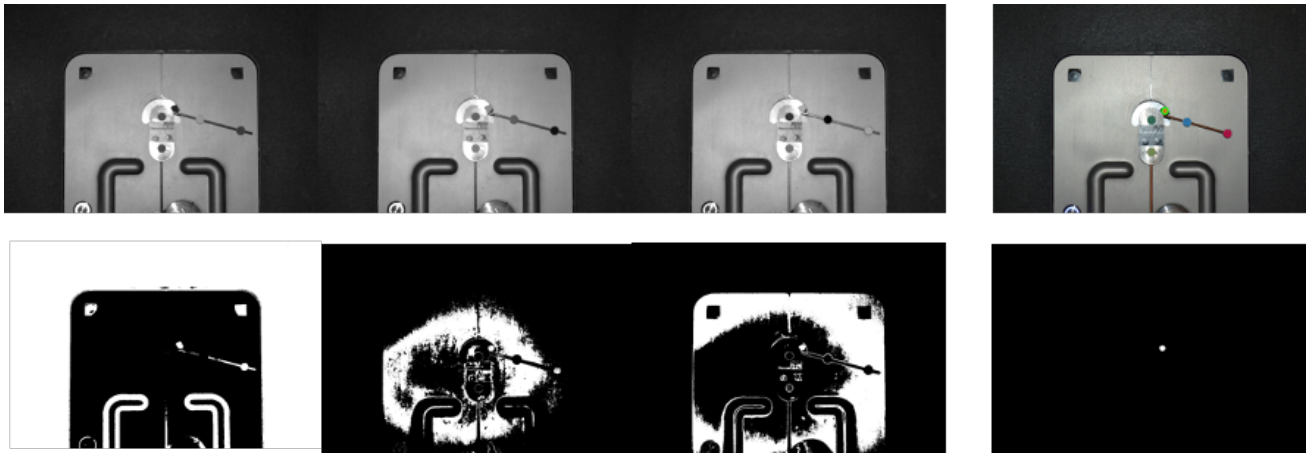
[11] https://pypi.org/project/pyserial/
[12] https://freeware.the-meiers.org/

```
import numpy as np
import cv2

# Colour thresholds
c1 = [70, 120, 100, 120, 120, 140]  # d_green - Origin
c2 = [110, 150, 110, 130, 140, 160]  # y_green - Origin 2
c3 = [130, 190, 120, 135, 150, 190]  # yellow - bending pin
c4 = [70, 130, 110, 130, 80, 110]  # Blue - Point 1
c5 = [70, 115, 150, 200, 130, 150]  # Magenta - point 2
colors = [c1, c2, c3, c4, c5]

# LAB colour markers detection
def circle_detect_LAB_list(image, color_th_list):
# Colour space conversion from BRG to LAB
lab = cv2.cvtColor(img, cv2.COLOR_BGR2LAB)

# Min and Max therholds
l_min, l_max = color_th_list[0], color_th_list[1]
a_min, a_max = color_th_list[2], color_th_list[3]
b_min, b_max = color_th_list[4], color_th_list[5]

# Channels split
l = image[:, :, 0]
a = image[:, :, 1]
b = image[:, :, 2]
lab_split = np.concatenate((l, a, b), axis=1)
cv2.namedWindow("LAB-split", cv2.WINDOW_NORMAL)
cv2.imshow("LAB-split", lab_split)

# L channel binary filter
ret, min_l = cv2.threshold(l, l_min, 255, cv2.THRESH_BINARY)
ret, max_l = cv2.threshold(l, l_max, 255, cv2.THRESH_BINARY_INV)
l_filter = cv2.bitwise_and(min_l, max_l)

# A channel binary filter
ret, min_a = cv2.threshold(a, a_min, 255, cv2.THRESH_BINARY)
ret, max_a = cv2.threshold(a, a_max, 255, cv2.THRESH_BINARY_INV)
a_filter = cv2.bitwise_and(min_a, max_a)

# B channel binary filter
ret, min_b = cv2.threshold(b, b_min, 255, cv2.THRESH_BINARY)
ret, max_b = cv2.threshold(b, b_max, 255, cv2.THRESH_BINARY_INV)
b_filter = cv2.bitwise_and(min_b, max_b)

# Binary AND operation
int_filter = cv2.bitwise_and(l_filter, a_filter)
final_img = cv2.bitwise_and(int_filter, b_filter)

# blureing to reduce noise
blured_img = cv2.medianBlur(final_img, 7)

# Feature detection function
circles = cv2.HoughCircles(blured_img, cv2.HOUGH_GRADIENT, 1, 100, param1=100, param2=15,
↪  minRadius=40, maxRadius=100)

return circles
```

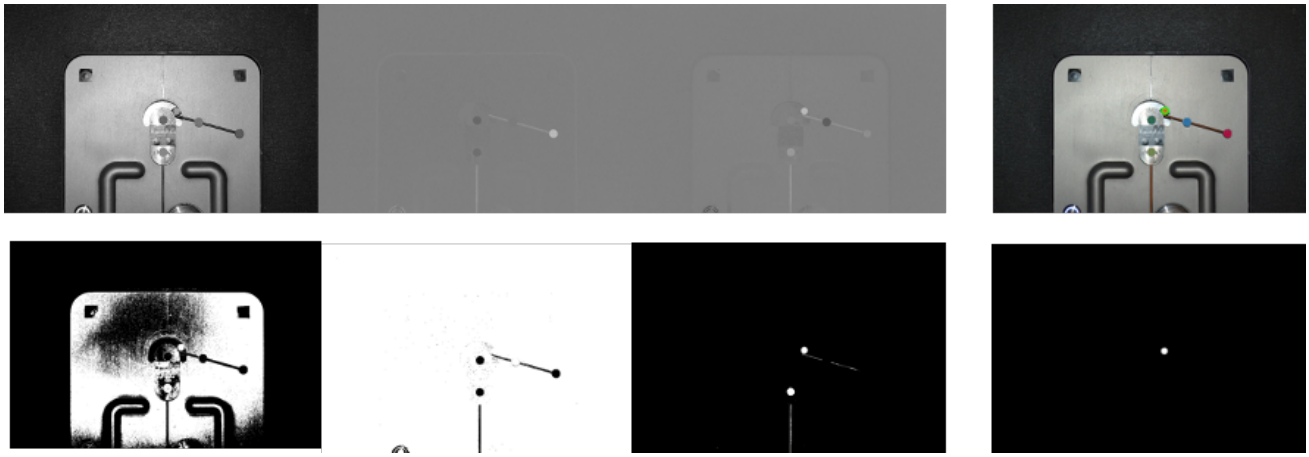
**Source code (12):** *Colour based segmentation, Python snippet is used to detect colour markers using LAB colour space based segmentation.*



**(a)** *Markers isolation test, different lighting conditions, each image has a glossy marker set on the left and a matt markers set on the right, lighting conditions from left to right: camera flash lighting, daylight condition and artificial warm light (2700 K).*

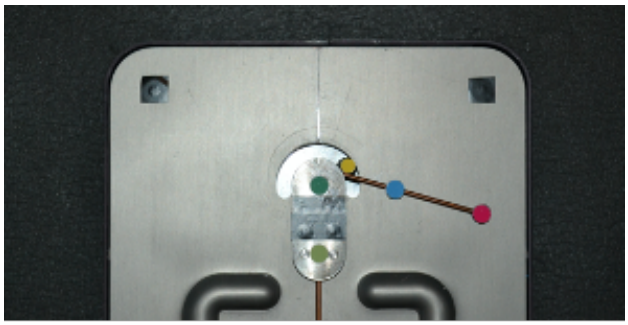


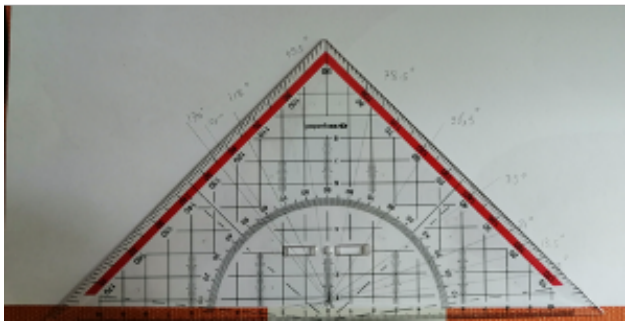
**(b)** *RGB (BRG) colour based segmentation, Top left: from lest to right, Blue, Red and green channel split (white is the maximum value), Bottom left: binary colour filter using manually measured marker's colour thresholds for each colour channel (white represents the pixels which falls between higher and lower thresholds), Top right: green circle represents the successful detection of yellow marker, Bottom right: A bitwise AND operation for the three colour filters, which is used for hough circle detection algorithm.*



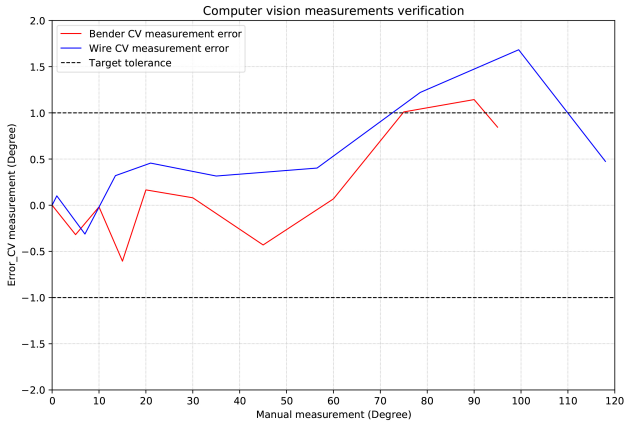
**(c)** *LAB colour based segmentation, Top left: from lest to L channel(Light intensity, white is the maximum value), A channel (Green to magenta, black is green and white is magenta) and B Channel (Blue to yellow. black is blue and white is yellow), Bottom left: binary colour filter using manually measured marker's colour thresholds for each colour channel (white represents the pixels which falls between higher and lower thresholds), Top right: green circle represents the successful detection of yellow marker, Bottom right: A bitwise AND operation for the three colour filters, which is used for hough circle detection algorithm.*

**Figure (42):** *Yellow marker isolation experiment, colour based segmentation process to isolate colour coded assistive markers.*





**(a)** *Computer vision, measurement verification by taking manual readings.*




**(b)** *Measurement verification graph, x: manual measurements and y: the difference in measurement between manual and CV measurements.*

**Figure (43):** *The process of CV measurement verification.*

```
import numpy as np

# Polynomial (cubic) data fitting model.
x = np.linspace(3, 130, 200)
X = calib_angels
Y = x_positive
A = np.vstack([X ** 0, X ** 1, X ** 2, X **
↪  3])
sol, r, rank, sv = la.lstsq(A.T, Y)
y_fit = sol[0] + sol[1] * x + sol[2] * x **
↪  2 + sol[3] * x ** 3

# Multi linear data fitting model.
x = np.linspace(3, 130, 200)
X = calib_angels
Y = x_positive
A = np.vstack([X ** 0, X ** 1])
sol, r, rank, sv = la.lstsq(A.T, Y)
y_fit = sol[0] + sol[1] * x
y_pre = sol[0] + sol[1] * X
```

**Source code (13):** *Calculating both polynomial data fitting model and multi linear data fitting models with the help of numpy library*

results were a relative error of 7.14% and absolute error of 0.64° in half circle bending ,and a relative error of 45.68% and absolute error of 1.59° in saddle curve bending.

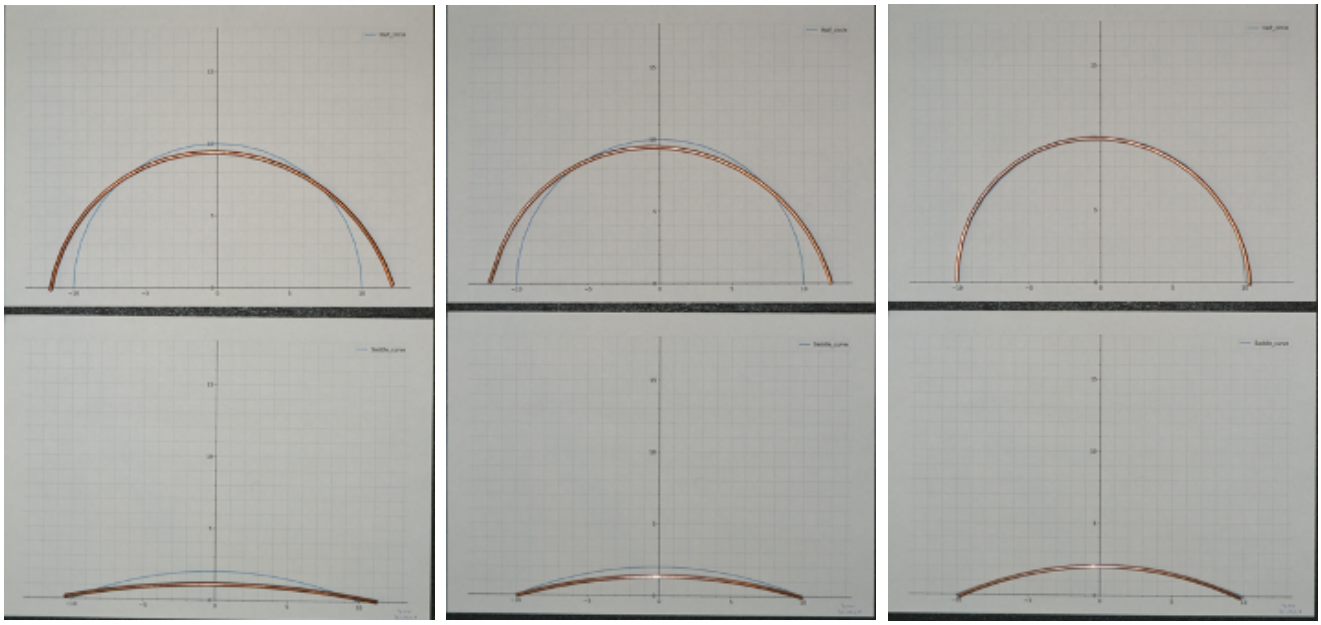
The second bending experiment (Fig.44b) was done with the help of computer vision automatic calibration process, and cubic polynomial data fitting model for springback compensation in python. The results were a relative error of 14.21% and absolute error of 1.28° in half circle bending ,and a relative error of 29.11% and absolute error of 1.01° in saddle curve bending.

The third bending experiment (Fig.44c) was similar to the second one but a multi-linear data fitting model for springback compensation instead of polynomial data fitting. The results were a relative error of 0.63% and absolute error of 0.05° in half circle bending ,and a relative error of 2.37% and absolute error of 0.08° in saddle curve bending.

Results graph shows that bending error did not improve in the second experiment but it not only has significantly improved in the third experiment but also successfully met accepted tolerance target (Fig.45).

## 9 Conclusion

It was clear that both numerical and empirical approaches compliment each other. In ideal situation, FEA simulation is better for isolation and analysis of physical problems but empirical experiments are required for verification and data feedback in FEA simulation. Computer vision proved to be a reliable method for



**(a)** *First bending experiment: (baseline case), standard manual calibration process, and embedded cubic polynomial data fitting model for springback compensation.*

**(b)** *Second bending experiment: computer vision automatic calibration process, and cubic polynomial data fitting model for springback compensation.*

**(c)** *Third bending experiment: computer vision automatic calibration process, and a multi-linear data fitting model for springback compensation.*

**Figure (44):** *Empirical wire bending experiments.*

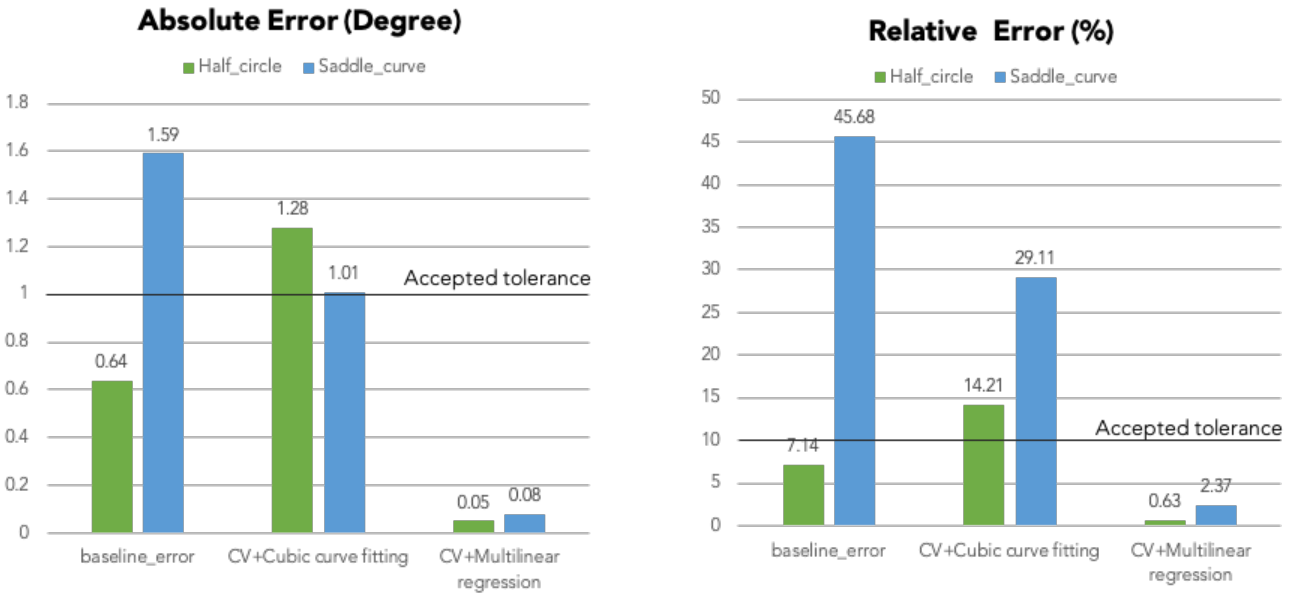



**Figure (45):** *Absolute and relative error comparison between the three experiments, showing the accepted tolerance levels.*

measurement to automate, time intensive manual calibration process.
Although that the methods used in this study was successful in bending semicircles and saddle curves, but further bending experiments are required for verification.
Finally, results are satisfactory for research purposes, but for industrial application, user interface has to be thought off considering knowledge level of construction workers. In addition material manufacturing and handling for construction application is different than lab environment. Therefore, material manufacturing, transportation and storage parameters should be considered. An example would be, the material used in this research is a cold rolled steel, while in construction hot rolled steel bars are commonly used, which means would be a difference in shape tolerances and material workability. Another example is material storage conditions. During this study, it was observed that almost 25% of purchased material were not stored, bundled or transported properly. It came with a partial plastic deformation of more than 1°, which not only exceeding the target tolerance but also will have a higher yielding partially due plastic hardening effect.

In summery, I assume that steel bending precision is not a stochastic problem but rather a complex one which relays heavily on the quality of measurements.

### 9.1 Lessons learnt

- Data fitting methods has a significant effect on bending accuracy.
- Computer vision measurement can be precise if used carefully.
- FEA simulation is useful to isolate physical phenomena and keeps a record of internal stress for corrective actions.

### 9.2 Future works

- Use computer vision as a feedback loop for corrective actions.
- Develop grasshopper plug-in and an open source python application for 2d wire bending.
- Improve markers quality and develop 3d computer vision measuring system.

### 9.3 Research contribution

- In depth analysis for 2D CNC wire bending process.
- Developing colour coded marking system for computer vision measurement application.(Calibration automation).
- The use of multi-linear data fitting model on springback compensation values instead of polynomial fitting. (hardening model).

## References


[1] I a Burchitz. *Improvement of Springback Prediction in Sheet Metal Forming*. 2008.

[2] Werner Sobek Ag and Allplan Bimplus. STUTTGART 21 : DIGITAL WORKFLOW ON A MEGA PROJECT.

[3] Maria Smigielska. Humanizing Digital Reality. *Humanizing Digital Reality*, (September), 2018.

[4] Aleksander Zubelewicz. Century-long taylor-Quinney interpretation of plasticity-induced heating reexamined.

[5] Jean-Marc Franssen and Paulo Vila Real. Annex C: Mechanical Properties of Carbon Steel and Stainless Steel. In *Fire Design of Steel Structures*, pages 407–427. Wiley-VCH Verlag GmbH & Co. KGaA, Weinheim, Germany, oct 2016.

[6] Technology Drive Canonsburg. ANSYS Meshing User Guide. *Knowledge Creation Diffusion Utilization*, 15317(January):724–746, 2012.

[7] Bernd Arno Behrens, Alexander Chugreev, Florian Bohne, and Ralf Lorenz. Approach for modelling the Taylor-Quinney coefficient of high strength steels. In *Procedia Manufacturing*, volume 29, pages 464–471. Elsevier B.V., 2019.

[8] ANSYS. System Coupling User ' s Guide Release 15.0. (November), 2013.

[9] Peter Kohnke. ANSYS Mechanical APDL Theory Reference. *ANSYS Mechanical APDL Theory Reference*, (19.0):297–299, 2018.

[10] Johansson Robert. *Numerical Python: : Scientific Computing and Data Science Applications with Numpy, SciPy and Matplotlib*, volume 10. Apress, Urayasu-shi, Chiba, Japan, second edi edition, 2019.