

INSTITUTO TECNOLÓGICO DE AERONÁUTICA



Adriana Nunes Chaves Lima

**ANALYSIS METHOD OF TEMPERATURES AND
HEAT FLOWS FOR ORTHOGONAL CUTTING
1045 STEEL BY THERMAL IMAGING**

Final Paper
2017

Course of Mechanical Engineering

Adriana Nunes Chaves Lima

**ANALYSIS METHOD OF TEMPERATURES AND
HEAT FLOWS FOR ORTHOGONAL CUTTING
1045 STEEL BY THERMAL IMAGING**

Advisor

Prof. Dr. Anderson Vicente Borille (ITA)

Co-advisor

Dipl. Wirt. Ing. Thorsten Augspurger (WZL)

MECHANICAL ENGINEERING

SÃO JOSÉ DOS CAMPOS
INSTITUTO TECNOLÓGICO DE AERONÁUTICA

2017

Cataloging-in Publication Data
Documentation and Information Division

Nunes Chaves Lima, Adriana

Analysis method of temperatures and heat flows for orthogonal cutting 1045 steel by thermal imaging / Adriana Nunes Chaves Lima.

São José dos Campos, 2017.

39f.

Final paper (Undergraduation study) – Course of Mechanical Engineering– Instituto Tecnológico de Aeronáutica, 2017. Advisor: Prof. Dr. Anderson Vicente Borille. Co-advisor: Dipl. Wirt. Ing. Thorsten Augspurger.

1. Usinagem. 2. Corte Ortogonal. 3. Programas. 4. Análise Térmica. 5. Estado Transiente. I. Instituto Tecnológico de Aeronáutica. II. Title.

BIBLIOGRAPHIC REFERENCE

NUNES CHAVES LIMA, Adriana. **Analysis method of temperatures and heat flows for orthogonal cutting 1045 steel by thermal imaging**. 2017. 39f. Final paper (Undergraduation study) – Instituto Tecnológico de Aeronáutica, São José dos Campos.

CESSION OF RIGHTS

AUTHOR'S NAME: Adriana Nunes Chaves Lima

PUBLICATION TITLE: Analysis method of temperatures and heat flows for orthogonal cutting 1045 steel by thermal imaging.

PUBLICATION KIND/YEAR: Final paper (Undergraduation study) / 2017

It is granted to Instituto Tecnológico de Aeronáutica permission to reproduce copies of this final paper and to only loan or to sell copies for academic and scientific purposes. The author reserves other publication rights and no part of this final paper can be reproduced without the authorization of the author.

Adriana Nunes Chaves Lima
Av. Dr. Eduardo Cury, 350, apt. 257
12.242-001 – São José dos Campos–SP

ANALYSIS METHOD OF TEMPERATURES AND HEAT FLOWS FOR ORTHOGONAL CUTTING 1045 STEEL BY THERMAL IMAGING

This publication was accepted like Final Work of Undergraduation Study

Adriana Nunes Chaves Lima

Author

Anderson Vicente Borille (ITA)

Advisor

Thorsten Augspurger (WZL)

Co-advisor

Prof. Dr. Jesuino Takashi Tomita
Course Coordinator of Mechanical Engineering

São José dos Campos: November 22, 2017.

I dedicate this work to my family, which have always supported me in my decisions and are the most happy ones with this academic achievement.

Acknowledgments

I thank my family for all the support to my personal and academic growth. My father Pedro, my great example of righteousness and perseverance, who made me admire ITA through his life story. My mother Leila, who never hesitated to provide me with the best possible preparation in all aspects of life. My little sister Natália, my first experience as a tutor, who fascinates me today by teaching me so much. Wilza, who raised me as a second mother, with endless love and affect. Thank you for sharing with me so intensely this life changing experience.

I thank the ITA Baja team and members, who became a family to me amidst all endless days and nights spent together at the workshop, often fighting through exhaustion and stress, all focused in a common goal: building the best possible Baja. I thank you for all technical knowledge and experience, besides the resilience and togetherness I acquired during my time with you.

I thank the people who have crossed ways with me in my still short professional path: Ronnie, Borille, Rafael, Diogo, Daniel and Ceyhun. Thank you for offering me guidance and support as I try to grow and improve.

Finally, I thank Nathianne, who has been with me since almost the beginning of my journey in ITA, providing me with motivation and emotional support when the burden got heavier. If not for you, the path would have certainly been a lot rougher. Thank you for sticking with me.

*“You have to expect things of yourself
before you can do them.”*

— MICHAEL JORDAN

Resumo

A manipulação de arquivos de geometria é usualmente associada ao uso de programas de CAD, que costumam ter licenças caras devido à quantidade de recursos sofisticados oferecidos por eles. Esse gasto é facilmente justificável quando o trabalho envolve a aplicação de um número grande dessas funções, por exemplo, no desenvolvimento de uma nova geometria para um projeto. No entanto, para muitos usuários, a tarefa pode ser simples e requerer poucas funções, por exemplo, para gerar um padrão regular de projeções ao longo de uma superfície de modo a obter um conjunto de pontos para guiar a trajetória de uma ferramenta de usinagem durante o processo. Esses clientes teriam que gastar uma quantia elevada em licenças, para então usar apenas uma pequena porção do que o programa tem a oferecer. Esse trabalho apresenta uma solução que permite a leitura direta de um arquivo IGES e a geração de um conjunto específico de pontos usando funções de código aberto. Embora seja programado em *MATLABTM*, ele é compilado em um arquivo executável, de forma que seja possível seu uso sem a necessidade de licença.

Abstract

The handling of geometry files is usually associated with the usage of CAD software, which commonly have costly licenses due to the amount of sophisticated features that they offer. This expense is easily justifiable when the work involves thorough application of a number of these functions, for example, when developing a new geometry for a given project. However, for many users, the task is simple and only requires a small amount of features, for example, when generating a regular pattern of projections along a surface to obtain a set of points to guide the path of a turning or milling tool throughout a machining process. These clients usually have to spend a lot of money on licenses, only to use a small portion of what the program has to offer. This paper presents a solution which can directly read an IGES geometry file and generate a specific set of points using simple open source functions. Although it is programmed using *MATLABTM*, it is compiled to an independent executable file, thus being capable of running without any license at all.

List of Figures

FIGURE 2.1 – Infrared photography of a cutting process (ABUKHSHIM <i>et al.</i> , 2006)	17
FIGURE 3.1 – Experimental setup (AUGSPURGER <i>et al.</i> , 2016)	20
FIGURE 4.1 – Scaled image showing temperature distribution	22
FIGURE 4.2 – Contour plot	23
FIGURE 4.3 – Placement of tool	24

List of Tables

TABLE 3.1 – Design of experiments (AUGSPURGER <i>et al.</i> , 2016)	21
---	----

List of Abbreviations and Acronyms

ANSI	American National Standards Institute
ASME	American Society of Mechanical Engineers
MATLAB	Numerical computation software from MathWorks
GUI	graphic user interface
GUIDE	graphic user interface development environment
AISI	American iron and steel institute
WZL	Werkzeugmaschinenlabor (Laboratory of Machine Tools and Production Engineering)
FOV	Field of view

List of Symbols

F_c	Cutting force on the power direction [N]
F_p	Passive force [N]
v_c	Cutting velocity [m/min]
P	Power developed along cutting process [W]
w	Width of tool [mm]
t_{uc}	Uncut chip thickness [μm]
T_e	Environment temperature [$^{\circ}C$]
λ_{Tool}	Heat conductivity of tool material
t	Time [s]

Contents

1	INTRODUCTION	15
1.1	Structure	15
2	STATE OF THE ART	17
2.1	Infrared Termography	17
2.2	Temperature Modelling	18
2.3	Image Processing	18
3	EXPERIMENTAL SETUP	20
4	CODE IMPLEMENTATION	22
4.1	MATLAB environment	22
4.2	Auxiliar functions	23
4.2.1	Contour plot	23
4.2.2	Hough lines transformation	23
4.3	Implementation steps	24
4.3.1	Overview	24
4.3.2	Finding tool edges	24
4.3.3	Maximum temperatures	25
4.3.4	Temperature fields	25
4.3.5	Heat flows - Chip and Tool	25
5	RESULTS	26
6	CONCLUSION	27

CONTENTS	xiv
<hr/>	
BIBLIOGRAPHY	28
APPENDIX A – SOURCE CODE	29
A.1 Temperature Analysis	29
GLOSSARY	40

1 Introduction

In many machining cases, orthogonal cutting may be considered a good approximation to performance on the major cutting edge, that is why it has been extensively studied (SHAW; COOKSON, 2005). For instance, planing and facing processes are some examples that orthogonal cutting conditions can be observed.

Also, it is known that thermal behavior during the cutting process, as temperature fields and heat flows, has a important influence on tool life, surface finish and metallurgical structure of workpiece and machinability. Then, the study of thermal analysis on orthogonal cutting case shall be able to provide a better comprehension of many studies concerning thermal modeling of metal cutting processes (KOMANDURI; HOU, 2000), (KOMANDURI; HOU, 2001).

The aim of this paper is to develop a method to analyze thermal images generated during the orthogonal cutting of AISI 1045 metal, focusing on the transient state due to the short time of cutting. It will be analyzed temperature distribution along the cutting tool, heat flows through tool, chip and workpiece.

1.1 Structure

This work is divided into 6 Chapters, including this **Introduction**, plus one Appendix.

The second chapter, **State of the Art**, describes the existing technology which is relevant for the scope of this paper and upon which the work was built.

The third chapter, **Experimental Setup**, describes the methodology and materials that conducted the experiments.

The fourth, **Code Implementation**, describes the logical implementation of the final analysis code.

The fifth, **Results**, presents the results and discussions about the outputs of the final code.

The sixth and final chapter, **Conclusion**, sums up what was accomplished in this

work and suggests how it may be expanded for new processes.

The Appendix **Source Code** contains all the code written for the program.

2 State of the Art

2.1 Infrared Termography

Infrared termography is a contactless way to measure infrared eletromagnetic energy. It makes possible to observe contours of different bodies due to their temperature distribution, since every body is able to emit eletromagnetic radiation when its temperature is above absolute zero. For this reason, it is a very important tecnology in military use, because it allows objects be seen even without propoer illumination or in total lack of light situations.

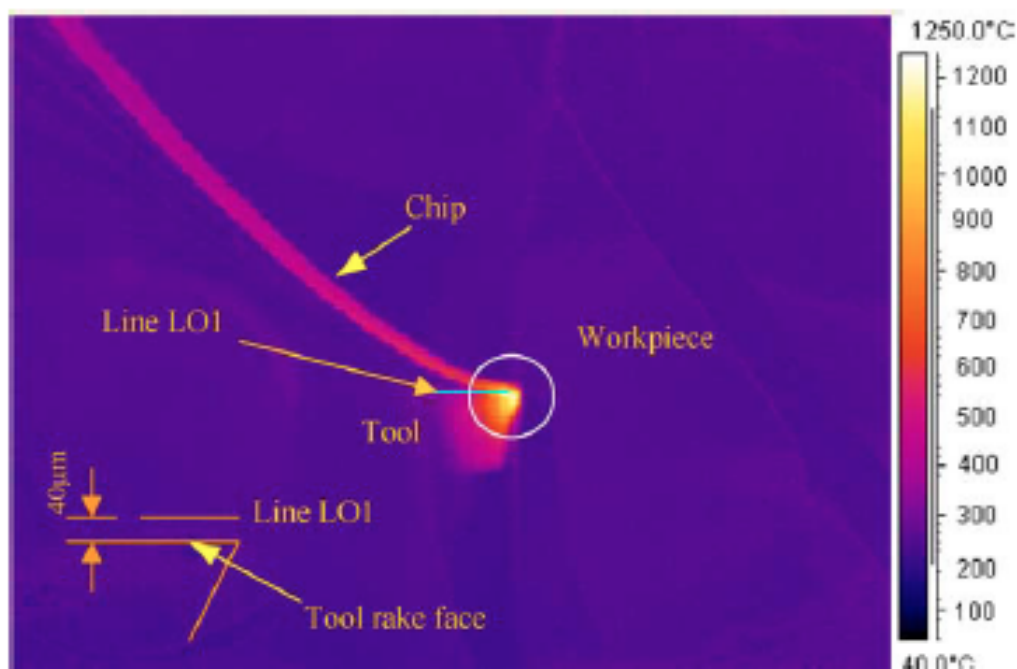


FIGURE 2.1 – Infrared photography of a cutting process (ABUKHSHIM *et al.*, 2006)

The thermography tool is able to work in two dfferent ways: passive and active. The passive way occurs when the subject matter has its temperature different from the environment (often higher). On the other hand, the active way needs an external heat source to induce a reasonable contrast between the object and the background (MALDAGUE,

2000). For the case under study, high speed thermography has its positive and negative points. On the positive side, it may be mentioned:

- Fast inspection rate (reasonable number of images of high speed cutting)
- Contactless (no interference during the cutting process)
- Easy interpretation of the results (indexed image with temperatures in each pixel)

But it is also important to mention the difficulties that in this method still prevail:

- Only a limited thickness can be measured (under the main surface)
- Determine a suitable emissivity is a challenge (it changes with temperature variation)

2.2 Temperature Modelling

write

2.3 Image Processing

Infrared thermography is a contactless way to measure infrared electromagnetic energy. It makes possible to observe contours of different bodies due to their temperature distribution, since every body is able to emit electromagnetic radiation when its temperature is above absolute zero. For this reason, it is a very important technology in military use, because it allows objects be seen even without proper illumination or in total lack of light situations. The thermography tool is able to work in two different ways: passive and active. The passive way occurs when the subject matter has its temperature different from the environment (often higher). On the other hand, the active way needs an external heat source to induce a reasonable contrast between the object and the background (MALDAGUE, 2000). For the case under study, high speed thermography has its positive and negative points. On the positive side, it may be mentioned:

- Fast inspection rate (reasonable number of images of high speed cutting)
- Contactless (no interference during the cutting process)
- Easy interpretation of the results (indexed image with temperatures in each pixel)

But it is also important to mention the difficulties that this method still have:

-
- Only a limited thickness can be measured (under the main surface)
 - Determine a suitable emissivity is a challenge (it changes with temperature variation)

3 Experimental Setup

The experiments were carried out on WZL shop floor, located in Aachen in Germany, acquiring thermal images by means of high speed infrared camera FLIR SC7600 (with framerate of 328 fps and a resolution of 640 x 512 pixels), it was equipped with a macro lens 1:1 and FOV 9.6 x 7.7 mm. The test bench works in a way that the tool stays in a fixed position in relation to the camera, keeping the relative distance between tool and camera constant, then the scale factor provided by this setting was 15 $\mu\text{m}/\text{pixel}$. It allows the metric conversion for future post processing of images.

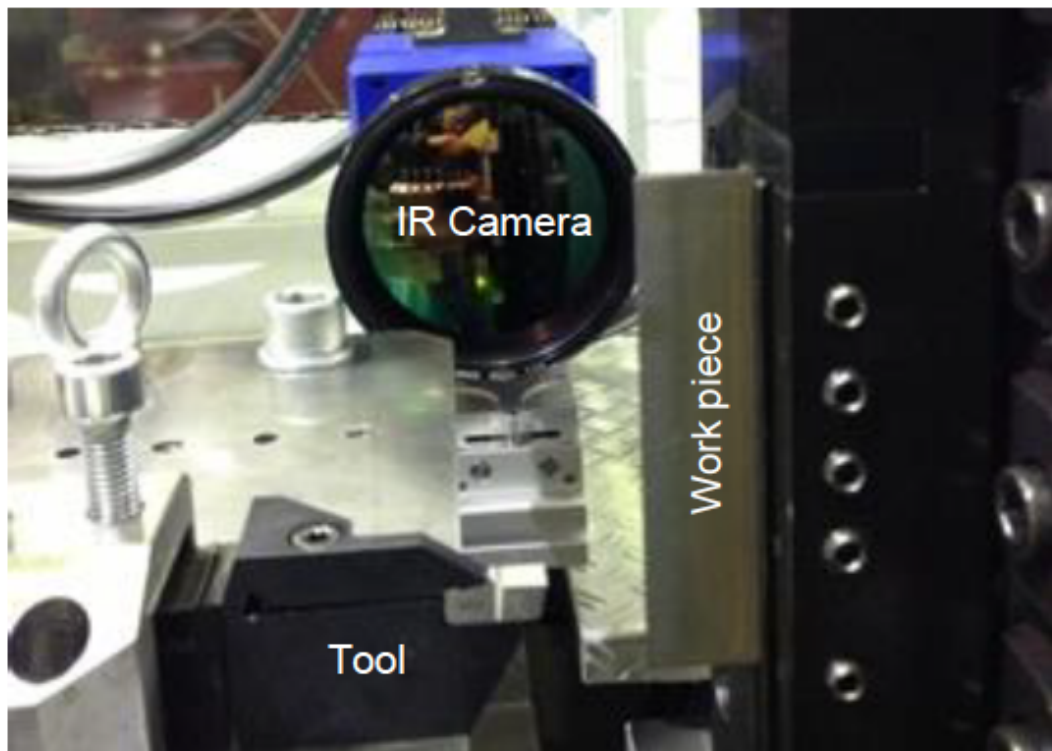


FIGURE 3.1 – Experimental setup (AUGSPURGER *et al.*, 2016)

An important factor for a reliable temperature measurement is the correct choice of the components emissivity. To make it easier, the tool was coated with a black ink, allowing the emissivity valuation for this case, which provided a value of $\epsilon = 0.85$. For the chip case, it was evaluated in its temperature range $\epsilon = 0.4$. It is also important to highlight the

camera settings, factors as integration time and filters are essential to determine a reliable measurements due to the amount of eletromagnetic radiation received on camera's sensors. The higher are the temperatures higher is the energy produced, then smaller should be the integration time, which is the time that sensor of energy receives radiation and converts into temperature afterwards. These configurations allowed measurements in a range from 200 °C until 900 °C.

The tool material was uncoated carbide insert (Sandvik H13A), with rake angle 6° , clearance angle 3° , cutting radius $r_\beta < 5\mu\text{m}$ and width 4.4 mm. The workpiece material was AISI 1045 normalized and its dimensions were 3.5 x 200 x 80 mm (width, lengh, heigth respectively).

For force acquisition during the process, it was used a three-component piezoelectric force platform, determining the cutting force and passive force. Since the cutting process is carried out in a linear and constant motion, it is possible to determine the overall power P with velocity and cutting force.

All the experiments were were held without coolant, with velocities of 100 m/min and 150 m/min and $a_p = [0.2, 0.3, 0.4, 0.5]$ mm. The analysis method was built on MATLAB platform with the support of its image processing toolbox. This was the chosen software due the easy connection between FLIR software and MATLAB, since FLIR software can export its images to .mat format, which are indexed matrices prejected to MATLAB environment. Each pixel from the exported images contains information about position and its temperature.

The design of the experiments are listed on the table bellow:

Experiments	Cutting Velocity [m/min]	Uncut chip thickness [μm]	Integration time [μs]	Cutting Force [N]	Passive Force [N]	Heat treatment
VP41_1_H200_V100_C45_MF_425	100	200	425	1500	1000	Normalized
VP41_2_H200_V100_C45_MF_425	100	200	425	1565	1005	Normalized
VP42_1_H300_V100_C45_MF_425	100	300	425	2250	1159	Normalized
VP42_2_H300_V100_C45_MF_285	100	300	285	2136	1079	Normalized
VP43_1_H400_V100_C45_MF_285	100	400	285	2716	1118	Normalized
VP45_2_H200_V150_C45_MF_425	150	200	425	1448	688	Normalized
VP46_1_H300_V150_C45_MF_285	150	300	285	2006	801	Normalized
VP46_2_H300_V150_C45_MF_285	150	300	285	2004	875	Normalized
VP49_1_H400_V150_C45_MF_285	150	400	285	2675	1046	Normalized
VP49_2_H400_V150_C45_MF_285	150	400	285	2590	1000	Normalized
VP50_1_H500_V150_C45_MF_285	150	500	285	3220	1120	Normalized
VP50_2_H500_V150_C45_MF_285	150	500	285	3178	1162	Normalized

TABLE 3.1 – Design of experiments (AUGSPURGER *et al.*, 2016)

4 Code Implementation

4.1 MATLAB environment

As mentioned on chapter 3, FLIR software provides as output variables indexed matrices in .mat format, which is MATLAB variable format. Each pixel contains temperature information about itself, it is possible to visualize an example on a scaled image on the following figure:

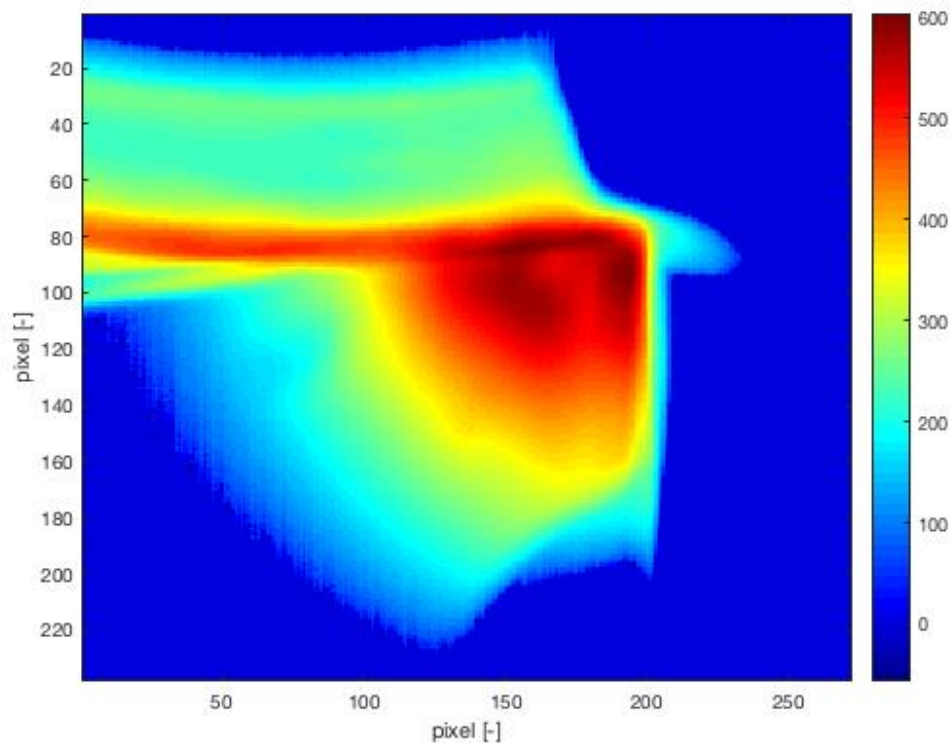


FIGURE 4.1 – Scaled image showing temperature distribution

From figure 4.1 with MATLAB Image Processing Toolbox support it is possible to extract many informations about the image, such as:

- Edges recognition

- Image segmentation for tool, chip and workpiece
- Detection of tool tip
- Determine isotherms along tool

4.2 Auxiliar functions

4.2.1 Contour plot

This is an important tool for this paper, contour plot is able to provide same level curves. Since the variable used on the process is a temperature matrix, this tool will calculate continuous lines, whose each pixel has a very close temperature measurements. Doing it with a determined and small tolerance, the lines calculated are isotherms of the image. Then, with these lines it is also possible to get its coordinates, which it will be essential to calculate heat carried away from volume control by means of tool.

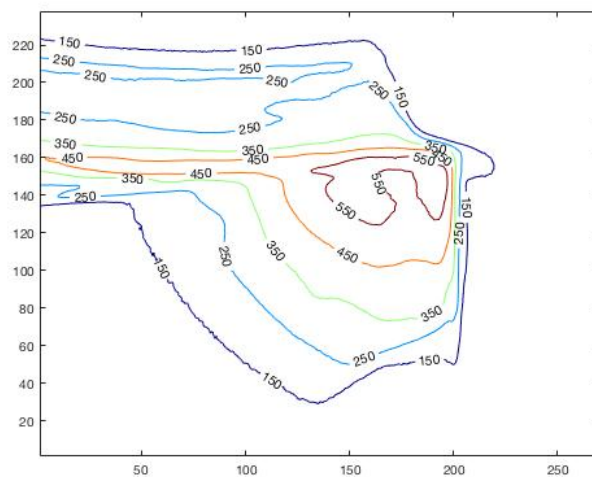


FIGURE 4.2 – Contour plot

4.2.2 Hough lines transformation

Hough transform is a extensively method used in computer vision. It is a extraction feature for complex geometries, using normal parametrization for straight lines (DUDA; HART, 1972). Concerning about the images, the rake and clearance face can be mapped by means of hough lines transformation in MATLAB. It is necessary to provide a probable angle range in what the angular coefficient of seeked lines are defined. More precise is this angle range, more reliable and faster will be the output.

The test bench, where the experiments were held, allows a fixed placement of tool.

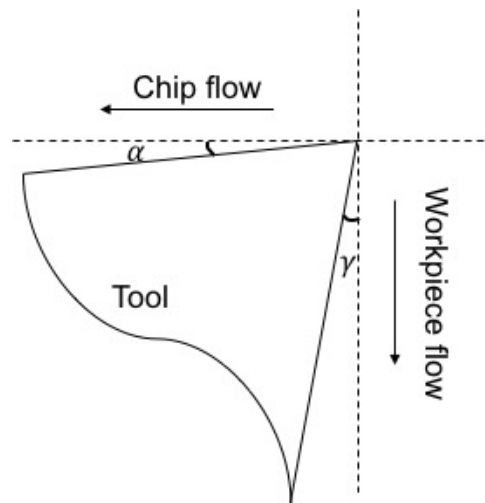


FIGURE 4.3 – Placement of tool

It means the angle between the rake face and horizontal line and the angle between clearance face and vertical line are always the designed rake and clearance angles, respectively. In other words, the tool does not rotate in relation to the reference axes. Because of this, it is possible to perform hough transformation on the image, being very accurate. As the rake and clearance angle are always 6° and 3° , respectively, the hough transform processing will last a shorter time with predetermined angles than otherwise.

4.3 Implementation steps

4.3.1 Overview

The final aim of the program was to be able to identify the tool and chip shapes, then the analysis could extract and provide features that were essential for the results of this paper. By means of image processing and some input data, features like maximum cutting zone temperature, maximum chip temperature, heat flows through chip and tool are some examples of what the code is able to provide.

FAZER tabela COM INPUT/OUTPUT

4.3.2 Finding tool edges

As mentioned in the subsection Hough lines transformation, the method to find tool edges has to provide an accurate range of angles that the rake and clearance angles are inserted. The process is simple and it is demonstrated as follows:

```

1      function obj = calculateCoordinates(obj)
2          obj.BW = edge(obj.frame,'sobel');
3          %-----Finding the clearance face-----
4          [H, THETA, RHO] = hough(obj.BW,'Theta',2:5);%Hough transformation
5          P = houghpeaks(H, 10);
6          obj.lines = houghlines(obj.BW, THETA, RHO, P, 'FillGap', 15,'MinLength'
,10);%Here we can find the lines of cutting edge and afterwards find the
coordinate of the tool tip
7          l = length(obj.lines);
8          obj.coordCF = [];
9          for i=1:l
10             Theta = obj.lines(i).theta;
11             t1 = obj.lines(i).point1;
12             t2 = obj.lines(i).point2;
13             rho = obj.lines(i).rho;
14             if rho < 204 && rho > 198
15                 obj.coordCF = [t1;t2];
16                 obj.ClearanceAngle = Theta;
17             end
18         end
19         %-----Finding the rake face-----
20         [H, THETA, RHO] = hough(obj.BW,'Theta',81:85);%Hough transformation
21         P = houghpeaks(H, 10);
22         obj.lines = houghlines(obj.BW, THETA, RHO, P, 'FillGap', 15,'MinLength'
,10);%Here we can find the lines of cutting edge and afterwards find the
coordinate of the tool tip
23         l = length(obj.lines);
24         obj.coordRF = [];
25         for i=1:l
26             Theta=obj.lines(i).theta;
27             t1 = obj.lines(i).point1;
28             t2 = obj.lines(i).point2;
29             rho = obj.lines(i).rho;
30             if rho < 103 && rho > 98
31                 obj.coordRF = [t1;t2];
32                 obj.RakeAngle = 90 - Theta;
33             end
34         end
35     end

```

Since the rake angle is 6° and the clearance is 3° ranges of [81:85] and [2:5] were given to each respectively. Concerning about the rake angle, the range of angles is given by the complementar(?) angles due to its reference in hough method.

4.3.2.1 Rake and clearance face

4.3.2.2 Tool tip coordinates

4.3.3 Maximum temperatures

4.3.4 Temperature fields

4.3.5 Heat flows - Chip and Tool

4.3.5.1 Heat partitions

5 Results

write

6 Conclusion

write

Bibliography

ABUKHSHIM, N.; MATIVENGA, P.; SHEIKH, M. Heat generation and temperature prediction in metal cutting: A review and implications for high speed machining. **International Journal of Machine Tools and Manufacture**, Elsevier, v. 46, n. 7, p. 782–800, 2006.

AUGSPURGER, T.; KLOCKE, F.; DÖBBELER, B.; BROCKMANN, M.; LIMA, A. **Experimental investigation of temperatures and heat flows for orthogonal cutting 1045 steel by thermal imaging**. 08 2016. 6 p.

DUDA, R. O.; HART, P. E. Use of the hough transformation to detect lines and curves in pictures. **Communications of the ACM**, ACM, v. 15, n. 1, p. 11–15, 1972.

KOMANDURI, R.; HOU, Z. B. Thermal modeling of the metal cutting process: part i temperature rise distribution due to shear plane heat source. **International Journal of Mechanical Sciences**, Elsevier, v. 42, n. 9, p. 1715–1752, 2000.

KOMANDURI, R.; HOU, Z. B. Thermal modeling of the metal cutting process part ii: temperature rise distribution due to frictional heat source at the tool chip interface. **International Journal of Mechanical Sciences**, Elsevier, v. 43, n. 1, p. 57–88, 2001.

MALDAGUE, X. Applications of infrared thermography in nondestructive evaluation. **Trends in optical nondestructive testing**, Elsevier Science, p. 591–609, 2000.

SHAW, M. C.; COOKSON, J. **Metal cutting principles**. [S.l.]: Oxford university press New York, 2005.

Appendix A - Source Code

A.1 Temperature Analysis

```
1 classdef TemperatureAnalyze
2     % This class was built to analyze the temperature inside the tool
3     % shape, the temperature gradient, the isotherms...
4
5     properties(GetAccess = 'public', SetAccess = 'private')
6         CoordinateToolTip;
7         TemperatureToolTip;
8         RakeAngle;%Rake face slope
9         ClearanceAngle;%Clearance face slope
10        ShearAngle;
11        FrictionAngle;
12        MeanTemperatureTool;
13        MaximumTemperatureTool;
14        MaximumTemperatureChip;
15        MaximumTemperatureCuttingZone;
16        HeatCarriedAwayByChip;
17        HeatFluxAwayFromToolTip;
18        HeatFluxThroughWorkpiece;
19        TotalPowerBalance;
20        InternalEnergyTool;
21        CuttingForcePowerDirection;
22        CuttingForceUncutChipThicknessDirection;
23        CuttingForceParallelToolFace;
24        CuttingForceParallelShearPlane;
25        CuttingForcePerpendicularShearPlane;
26        CuttingForcePerpendicularToolFace;
27        CoefficientFriction;
28        ShearStress;
29        NormalStress;
30        PecletNumber;
31        RatioR;
32        ShearEnergyVolume;
33        FrictionEnergyVolume;
34        CuttingVelocity;
35        UnCutChipThickness;
36        ContactLength;
37    end
38
39    properties(GetAccess = 'private', SetAccess = 'private')
40        coordRF;
41        coordCF;
42        BW;
43        lines;
44        frame;
45        pointCF;%auxiliar to plot the cutting edge
46        pointRF;
47        pointM;
48        Tx;%auxiliar to plot the gradients of the frame
49        Ty;
50        biImageTool;%Binary image of the tool shape
51        biImageChip;
```

```

52     biShearLine;
53     xyMaxTemp;%coordinates of the point inside the chip with maximum Temperature
54     lineChip;
55     lineTool;
56     validTemperature;
57     heatCapacity;
58     nExcPoints;
59     heatAccumulatedPerLine;
60     ptosLines;
61     extPtosLineChip;
62     line200;
63 end
64
65 methods
66     % methods, including the constructor are defined in this block
67
68     function obj = TemperatureAnalyze2(Frame,index)%constructor
69         %Inputs -----
70         Fp = 3220;%Cutting force in the power direction (Newtons)
71         Fq = 1120;%Passive force (Newtons)
72         widthTool = 4.4*10^-3;%meters
73         Vp = 150/60;%meters/second
74         tuc = 500*10^-6;%meters
75         clength = 0.00251;%Define as an empty vector if we do not have
76         %the mean value
77         tt = [197 78];
78         obj.validTemperature = 200;% For any experiment
79         A = 0.1;%percentage of the deformation energy that is converted in heat
80         %-----
81         obj.CuttingVelocity = Vp*60;%m/minute
82         obj.UnCutChipThickness = tuc;
83         obj.frame = Frame(index).f;
84         if isequal(clength,[])
85             clength = obj.contactLength();
86         end
87         obj.ContactLength = clength;
88         % obj = obj.calculateCoordinates();
89         % if isempty(obj.coordRF)==0 && isempty(obj.coordCF)==0
90         % % obj = obj.coordinateToolTip();
91         % else %Default conditions
92         % if isempty(obj.coordRF)
93         % obj.RakeAngle = 6;
94         % end
95         % if isempty(obj.coordCF)
96         % obj.ClearanceAngle = 3;
97         % end
98         % end
99         obj.CoordinateToolTip = tt;
100         obj.ClearanceAngle = 3;
101         obj.RakeAngle = 6;
102         obj.frame = Frame(index).f;
103         obj = obj.toolContour();
104         obj = obj.findLineTool();
105         obj = obj.chipContour();
106         obj = obj.findLineChip();
107         obj = obj.pointsRFandCF();
108         obj = obj.TempTT();
109         obj = obj.meanTemperatureTool();
110         obj = obj.maxTemperatureTool();
111         obj = obj.maximumTemperature();
112         obj = obj.maxTemperatureChip();
113         obj = obj.calculateGradient();
114         obj = extremePointsChip(obj);
115         obj = obj.heatBalance(tuc,Vp,widthTool);
116         obj = obj.internalEnergyTool(widthTool);
117         obj = obj.shearLine();
118         obj = obj.calculatePecletNumber();
119         obj = obj.forcesValues(Fp,Fq,widthTool,tuc);
120         obj.TotalPowerBalance = 0.97*(obj.CuttingVelocity*(obj.
CuttingForcePowerDirection*(1-A) + obj.CuttingForceParallelToolFace*A*obj.RatioR

```

```

121         )/60);
122         obj.HeatFluxThroughWorkpiece = obj.TotalPowerBalance - obj.
HeatCarriedAwayByChip - obj.HeatFluxAwayFromToolTip;
123         end
124         function obj = framesOverlap(obj,Frame,index)
125             cTT = obj.CoordinateToolTip;
126             alpha = (90 - obj.ClearanceAngle)*pi/180;
127             gamma = obj.RakeAngle*pi/180;
128             p1 = cTT + 67*[-cos(gamma) sin(gamma)];
129             p2 = cTT + 33*[-cos(alpha) sin(alpha)];
130             c = [cTT(1) p1(1) p2(1)];
131             r = [cTT(2) p1(2) p2(2)];
132             biTool70 = roipoly(Frame(index).e70,c,r);
133             aux = biTool70 == 1 & obj.biImageChip == 1;
134             biTool70 = biTool70 - aux;
135             biTool70andChip = biTool70 == 1 | obj.biImageChip == 1;
136             biFrame85 = ones(size(Frame(index).e85)) - biTool70andChip;
137             obj.frame = biTool70andChip.*Frame(index).e70 + biFrame85.*Frame(index).
e85;
138         end
139         function obj = toolContour(obj)
140             A = round(obj.CoordinateToolTip);
141             m = size(obj.frame,1);
142             xt = A(1);
143             yt = A(2);
144             y1 = round(yt + (xt - 1)*tan(obj.RakeAngle*pi/180));
145             x2 = round(xt - (m - yt)*tan(pi/2 - (90 - obj.ClearanceAngle)*pi/180));
146             c = [xt 0 0 x2];
147             r = [yt y1 m m];
148             B = roipoly(obj.frame,c,r);
149             obj.biImageTool = B;
150         end
151         function obj = chipContour(obj)
152             c = obj.line200(1,:);
153             r = obj.line200(2,:);
154             B = roipoly(obj.frame,c,r);
155             obj.biImageChip = B;
156             B2 = obj.biImageTool == 1 & B == 1;
157             B = B - B2;
158             obj.biImageChip = B;
159         end
160         function obj = maximumTemperature(obj)
161             obj.MaximumTemperatureCuttingZone = max(max(obj.frame));
162             [~,lin] = max(obj.frame);
163             [~,col] = max(max(obj.frame));
164             lin = lin(col);
165             obj.xyMaxTemp = [col lin];
166         end
167         function l = contactLength(obj)
168             imagesc(obj.frame)
169             imdistline%Help to measure the amount of pixels on the contact length
170             v = input('What is the value of the contact length for this frame? ');
171             close all
172             l = 15*10^-6*v;
173         end
174         function obj = maxTemperatureTool(obj)
175             C = obj.biImageTool;
176             Frame = C.*obj.frame;
177             T = max(max(Frame));
178             obj.MaximumTemperatureTool = T;
179         end
180         function obj = maxTemperatureChip(obj)
181             Frame = obj.biImageChip.*obj.frame;

```



```

188     obj.MaximumTemperatureChip = max(max(Frame));
189 end
190
191 function obj = meanTemperatureTool(obj)
192     B = obj.biImageTool;
193     Frame = B.*obj.frame;
194     B = Frame > obj.validTemperature;
195     Frame = B.*Frame;
196     s = sum(sum(Frame));
197     n = sum(sum(B));
198     meanT = s/n;
199     obj.MeanTemperatureTool = meanT;
200 end
201
202 function obj = displayBinary(obj)
203     imshow(obj.BW);
204     hold on
205     plot(obj.coordRF(:,1),obj.coordRF(:,2),'bx')
206     plot(obj.coordCF(:,1),obj.coordCF(:,2),'yx')
207     plot(obj.CoordinateToolTip(1),obj.CoordinateToolTip(2),'xm')
208     hold off
209 end
210
211 function obj = TempTT(obj)
212     p1 = round(obj.CoordinateToolTip + 5*[-cos(obj.RakeAngle*pi/180) sin(obj
.RakeAngle*pi/180)]);
213     p2 = round(obj.CoordinateToolTip + 5*[-cos((90 - obj.ClearanceAngle)*pi
/180) sin((90 - obj.ClearanceAngle)*pi/180)]);
214     p3 = round(obj.CoordinateToolTip + 5*[-(cos(obj.RakeAngle*pi/180)+cos
((90 - obj.ClearanceAngle)*pi/180)) (sin(obj.RakeAngle*pi/180)+sin((90 - obj
.ClearanceAngle)*pi/180))]);
215     T1 = obj.frame(p1(2),p1(1));
216     T2 = obj.frame(p2(2),p2(1));
217     T3 = obj.frame(p3(2),p3(1));
218     TT = obj.frame(round(obj.CoordinateToolTip(2)),round(obj
.CoordinateToolTip(1)));
219     T = [T1 T2 T3 TT];
220     obj.TemperatureToolTip = mean(T);
221 end
222
223 function obj = calculateCoordinates(obj)
224     obj.BW = edge(obj.frame,'sobel');
225     %-----Finding the clearance face-----
226     [H, THETA, RHO] = hough(obj.BW,'Theta',2:5);%Hough transformation
227     P = houghpeaks(H, 10);
228     obj.lines = houghlines(obj.BW, THETA, RHO, P, 'FillGap', 15,'MinLength'
,10);%Here we can find the lines of cutting edge and afterwards find the
coordinate of the tool tip
229     l = length(obj.lines);
230     obj.coordCF = [];
231     for i=1:l
232         Theta = obj.lines(i).theta;
233         t1 = obj.lines(i).point1;
234         t2 = obj.lines(i).point2;
235         rho = obj.lines(i).rho;
236         if rho < 204 && rho > 198
237             obj.coordCF = [t1;t2];
238             obj.ClearanceAngle = Theta;
239         end
240     end
241     %-----Finding the rake face-----
242     [H, THETA, RHO] = hough(obj.BW,'Theta',81:85);%Hough transformation
243     P = houghpeaks(H, 10);
244     obj.lines = houghlines(obj.BW, THETA, RHO, P, 'FillGap', 15,'MinLength'
,10);%Here we can find the lines of cutting edge and afterwards find the
coordinate of the tool tip
245     l = length(obj.lines);
246     obj.coordRF = [];
247     for i=1:l
248         Theta=obj.lines(i).theta;

```

```

249         t1 = obj.lines(i).point1;
250         t2 = obj.lines(i).point2;
251         rho = obj.lines(i).rho;
252         if rho < 103 && rho > 98
253             obj.coordRF = [t1;t2];
254             obj.RakeAngle = 90 - Theta;
255         end
256     end
257 end
258
259 function obj = coordinateToolTip(obj)
260     a = (obj.coordRF(1,2)-obj.coordRF(2,2))/(obj.coordRF(1,1)-obj.coordRF
(2,1));%The slope of the rake face hardly will be Inf(Infinite) or NaN(Not-a-
number),
261     %because we took for this face a slope smaller than 45
262     b = obj.coordRF(1,2)-a*obj.coordRF(1,1);
263     m = (obj.coordCF(1,2)-obj.coordCF(2,2))/(obj.coordCF(1,1)-obj.coordCF
(2,1));%Slope of the cf, in some cases may be Inf(inclination of 90?, for
example)
264     h = @(x) (a*x+b);%line of the clearance face represented by f
265     if m == Inf || m == -Inf%if the slope of the cf is 90? or -90?(Inf or -Inf
)
266         xi = obj.coordCF(1,1);%xi represents the coordinate x of the
intersection(tool tip)
267     else
268         n = obj.coordCF(1,2)-m*obj.coordCF(1,1);
269         xi = (n-b)/(a-m);
270     end
271     yi = h(xi);
272     obj.CoordinateToolTip = [xi yi];
273 end
274
275 function obj = displayImageAndToolTip(obj)
276     figure
277     imagesc(obj.frame);
278     hold on
279     plot(obj.CoordinateToolTip(1),obj.CoordinateToolTip(2),'xm')
280     hold off
281 end
282
283 function obj = pointsRFandCF(obj)
284     alpha = (90 - obj.ClearanceAngle)*pi/180;
285     gamma = obj.RakeAngle*pi/180;
286     obj.pointRF = obj.CoordinateToolTip + 90*[-cos(gamma) sin(gamma)];
287     obj.pointCF = obj.CoordinateToolTip + 90*[-cos(alpha) sin(alpha)];
288     obj.pointM = obj.CoordinateToolTip + 40*[-2*cos(alpha)-cos(gamma) 2*sin(
alpha)+sin(gamma)];
289 end
290
291 function vT = temperatureRFandCF(obj)
292     pixelpitch = 15*10^-3;% mm/pixel
293     extCF = obj.pointCF;% final point on the clearance face
294     extRF = obj.pointRF;% final point on the rake face
295     extM = obj.pointM;
296     l1 = round(abs(obj.CoordinateToolTip(1)-extRF(1)));%length in pixels
rake line
297     l2 = round(abs(obj.CoordinateToolTip(2)-extCF(2)));%length in pixels
clearance line
298     l3 = max(round(abs(obj.CoordinateToolTip-extM)));
299     vRFx = round(linspace(obj.CoordinateToolTip(1),extRF(1),l1));%
coordinates x of the rake line
300     vRFy = round(linspace(obj.CoordinateToolTip(2),extRF(2),l1));%
coordinates y of the rake line
301     vCFx = round(linspace(obj.CoordinateToolTip(1),extCF(1),l2));%
coordinates x of the clearance line
302     vCFy = round(linspace(obj.CoordinateToolTip(2),extCF(2),l2));%
coordinates y of the clearance line
303     vMx = round(linspace(obj.CoordinateToolTip(1),extM(1),l3));
304     vMy = round(linspace(obj.CoordinateToolTip(2),extM(2),l3));

```

```

305     T_RF = zeros(1,11);%temperature for each pixel (each coordinate pair) -
rake line
306     T_CF = zeros(1,12);%temperature for each pixel (each coordinate pair) -
clearance line
307     T_M = zeros(1,13);
308     for t=1:11
309         T_RF(t) = obj.frame(vRFy(t),vRFx(t));%Building the temperature
vector - rake line
310     end
311     for t=1:12
312         T_CF(t) = obj.frame(vCFy(t),vCFx(t));%Building the temperature
vector - clearance line
313     end
314     for t=1:13
315         T_M(t) = obj.frame(vMy(t),vMx(t));%Building the temperature vector -
clearance line
316     end
317     d1 = zeros(1,11);%distance for each pixel along the line
318     d2 = zeros(1,12);
319     d3 = zeros(1,13);
320     for t=1:11 - 1
321         d1(t+1)=(((vRFx(t+1)-vRFx(1))^2)+((vRFy(t+1)-vRFy(1))^2))^(1/2);
322     end
323     for t=1:12 - 1
324         d2(t+1)=(((vCFx(t+1)-vCFx(1))^2)+((vCFy(t+1)-vCFy(1))^2))^(1/2);
325     end
326     for t=1:13 - 1
327         d3(t+1)=(((vMx(t+1)-vMx(1))^2)+((vMy(t+1)-vMy(1))^2))^(1/2);
328     end
329     d1 = d1*pixelpitch;
330     d2 = d2*pixelpitch;
331     d3 = d3*pixelpitch;
332     figure
333     hold on
334     plot(d1,T_RF)
335     plot(d2,T_CF)
336     plot(d3,T_M)
337     xlabel('Distance from the tool tip (mm)')
338     ylabel('Temperature (?C)')
339     legend('Rake face','Clearance face','Middle vector')
340     hold off
341     figure
342     imagesc(obj.frame)
343     colormap jet
344     hold on
345     plot(vRFx,vRFy,'k','LineWidth',1)
346     plot(vCFx,vCFy,'k','LineWidth',1)
347     plot(vMx,vMy,'k','LineWidth',1)
348     hold off
349     m = min([l1 l2 l3]);
350     vT = [d1(1:m)' T_RF(1:m)' d2(1:m)' T_CF(1:m)' d3(1:m)' T_M(1:m)'];
351 end
352
353 function obj = extremePointsChip(obj)
354     [y,x] = find(obj.lineChip);
355     obj.extPtosLineChip = [x(1) y(1);x(end) y(end)];
356 end
357
358 function obj = displayIsotherms(obj)
359     tRF = obj.RakeAngle*pi/180;
360     tCF = (90 - obj.ClearanceAngle)*pi/180;
361     vRF = [-cos(tRF) sin(tRF)];
362     vCF = [-cos(tCF) sin(tCF)];
363     %p1 RF direction
364     t = (obj.CoordinateToolTip(1) - 1)/vRF(1);
365     p1 = obj.CoordinateToolTip - t*vRF;
366     %p2 CF direction
367     t = (256 - obj.CoordinateToolTip(2))/vCF(2);
368     p2 = obj.CoordinateToolTip + t*vCF;
369     %auxiliar to plot

```

```

370     auxX = [p1(1) obj.CoordinateToolTip(1) p2(1)]';
371     auxY = [p1(2) obj.CoordinateToolTip(2) p2(2)]';
372     Tmax = max(max(obj.biImageTool.*obj.frame));
373     Tv = obj.validTemperature;
374     v = round(Tv:40:Tmax);
375     %Display tool and isotherms-----
376     lc = obj.extPtosLineChip;
377     figure
378     imagesc(obj.frame)
379     colormap jet
380     hold on
381     plot(auxX,auxY,'k')
382     plot(lc(:,1),lc(:,2),'k--','LineWidth',1)
383     [C,h] = contour(obj.frame,v);
384     h.LineColor = [0.247 0.247 0.247];
385     clabel(C,h,'manual','FontSize',10);
386     x = obj.CoordinateToolTip(1);
387     y = obj.CoordinateToolTip(2);
388     axis([x-180 x+15 y-60 y+130])
389     cb = colorbar('vert');
390     zlab = get(cb,'ylabel');
391     set(zlab,'String','Temperature (?C)');
392     cb.Limits = [0 450];
393     cb.FontSize = 10;
394     zlab.FontSize = 10;
395     daspect([1,1,1])
396     ax = gca;
397     v = [0.2 0.6 1.0 1.4 1.8 2.2 2.6];
398     vt = v/0.015;
399     vx = x + 15 - vt;
400     ax.XTick = fliplr(vx);
401     ax.XTickLabel = fliplr(v);
402     ax.XAxisLocation = 'top';
403     vy = y - 60 + vt;
404     ax.YTick = vy;
405     ax.YTickLabel = v;
406     ax.YAxisLocation = 'right';
407     xlabel('millimeters')
408     ylabel('millimeters')
409     hold off
410 end
411
412 function obj = calculateGradient(obj)
413     pp = 15*10^-6;
414     tx = zeros(size(obj.frame));
415     ty = zeros(size(obj.frame));
416     k = 0;
417     for j = 1:5
418         [auxx,auxy]=gradaux_v2(obj.frame,j);
419         tx = tx + auxx;
420         ty = ty + auxy;
421         k = k + 1;
422     end
423     obj.Tx = tx/(k*pp);
424     obj.Ty = ty/(k*pp);
425 end
426
427 function obj = displayGradient(obj)
428     auxx = [obj.pointCF(1) obj.CoordinateToolTip(1) obj.pointRF(1)];
429     auxy = [obj.pointCF(2) obj.CoordinateToolTip(2) obj.pointRF(2)];
430     k = 75.4;
431     qx = -k*obj.Tx;
432     qy = -k*obj.Ty;
433     figure
434     quiver(qx,qy)
435     hold on
436     plot(auxx,auxy,'k')
437     xmin = obj.CoordinateToolTip(1) - 10;
438     xmax = obj.CoordinateToolTip(1) + 5;
439     ymin = obj.CoordinateToolTip(2) - 5;

```

```

440     ymax = obj.CoordinateToolTip(2) + 10;
441     axis([xmin xmax ymin ymax])
442     title('Tool Tip')
443     daspect([1,1,1])
444     figure
445     quiver(qx,qy)
446     hold on
447     plot(auxx,auxy,'k')
448     xmin = obj.CoordinateToolTip(1) - 30;
449     xmax = obj.CoordinateToolTip(1) - 10;
450     ymin = obj.CoordinateToolTip(2) - 5;
451     ymax = obj.CoordinateToolTip(2) + 15;
452     axis([xmin xmax ymin ymax])
453     title('Rake Face')
454     daspect([1,1,1])
455     figure
456     quiver(qx,qy)
457     hold on
458     plot(auxx,auxy,'k')
459     xmin = obj.CoordinateToolTip(1) - 10;
460     xmax = obj.CoordinateToolTip(1) + 10;
461     ymin = obj.CoordinateToolTip(2) + 10;
462     ymax = obj.CoordinateToolTip(2) + 20;
463     axis([xmin xmax ymin ymax])
464     title('Clearance Face')
465     daspect([1,1,1])
466 end
467
468 function obj = displayGradientContour(obj)
469     auxx = [obj.pointCF(1) obj.CoordinateToolTip(1) obj.pointRF(1)];
470     auxy = [obj.pointCF(2) obj.CoordinateToolTip(2) obj.pointRF(2)];
471     k = 75.4;
472     qx = -k*obj.Tx;
473     qy = -k*obj.Ty;
474     figure
475     quiver(qx,qy)
476     hold on
477     plot(auxx,auxy,'k')
478     contour(obj.frame,10)
479     xmin = obj.CoordinateToolTip(1) - 20;
480     xmax = obj.CoordinateToolTip(1) + 5;
481     ymin = obj.CoordinateToolTip(2) - 5;
482     ymax = obj.CoordinateToolTip(2) + 20;
483     axis([xmin xmax ymin ymax])
484     daspect([1,1,1])
485 end
486
487 function obj = findLineChip(obj)
488     [m,n] = size(obj.frame);
489     o = obj.RakeAngle*pi/180;
490     l = obj.ContactLength/(15*10^-6);
491     c = obj.CoordinateToolTip + l*[-cos(o) sin(o)];
492     xm = c(1);
493     ym = c(2);
494     x1 = xm - tan(o)*(ym - 1);
495     x2 = x1 + tan(o)*(m - 1);
496     vx = round(linspace(x1,x2,m));
497     vy = linspace(1,m,m);
498     B1 = zeros(m,n);
499     for i = 1:m
500         B1(vy(i),vx(i)) = 1;
501     end
502     B2 = B1 == 1 & obj.biImageChip == 1;
503     obj.lineChip = B2;
504 end
505
506 function obj = findLineTool(obj)
507     [m,n] = size(obj.frame);
508     Tmax = max(max(obj.biImageTool.*obj.frame));
509     Tv = obj.validTemperature;

```

```

510     v = round(Tv:40:Tmax);
511     if length(v) == 1
512         v = round([Tv Tmax]);
513     end
514     [C,~] = contour(obj.frame,v);
515     close
516     l = length(v);
517     B = zeros(m,n,l);
518     C = round(C);
519     for k = 1:l
520         [~,J] = find(C == v(k));
521         [~,p] = max(C(2,J));
522         J = J(p);
523         for z = J+1:J+C(2,J)
524             B(C(2,z),C(1,z),k) = 1;
525         end
526         if k == 1
527             obj.line200 = C(:,J+1:J+C(2,J));
528         end
529         B(:, :, k) = B(:, :, k) .* obj.biImageTool;
530     end
531     obj.lineTool = B;
532 end
533
534 function obj = heatBalance(obj,tuc,Vc,w)
535     k = 75.4;%heat conductivity
536     pp = 15*10^-6; %pixel pitch
537 %-----
538     %First part - Heat carried away by the chip
539     cp = [-4.39956806034758e-07 0.000707314520321484...
540         -0.0488770693887544 481.214007868631]; %AISI 1045
541     %Heat capacity for the workpiece
542     M = obj.lineChip.*obj.frame;
543     MH = polyval(cp,M);
544     MH(MH == cp(4)) = 0;
545     Ht = MH.*(obj.frame-22);%J/kg - 22 is the temperature of the environment
546     Ht = sum(sum(Ht));
547     n = sum(sum(obj.lineChip));
548     Hc = Ht/n; %mean entalpy on the line chip
549     % Vchip = 100*200/(60*n*15);
550     p = 7874; %kg/m^3
551     Qc = Hc*Vc*tuc*p;%Vc*tuc is the same for Vchip*tchip
552     obj.HeatCarriedAwayByChip = Qc*w;
553 %-----
554     %Second part - Heat carried away by the tool
555     dT = ((obj.Tx).^2 + (obj.Ty).^2).^(1/2);
556     Q = zeros(size(obj.lineTool,3),1);
557     for i = 1:size(obj.lineTool,3)
558         L = obj.lineTool(:, :, i);
559         Q(i) = sum(sum(L.*dT))*pp*w*k;
560     end
561     obj.heatAccumulatedPerLine = Q;
562     Qm = mean(Q(1:2));
563     obj.HeatFluxAwayFromToolTip = Qm;
564 end
565
566 function n = exceedingPoints(obj,Temperature)
567     B = obj.frame.*obj.biImageTool > Temperature;
568     n = sum(sum(B));
569 end
570
571 function obj = internalEnergyTool(obj,w)
572     pp = 15*10^-4;%in cm
573     cp = [2.50542895559373e-10 -1.99579761670655e-06 0.00274369536032376
574         3.09265830398264];%J/(K*cm3)
575     %Heat capacity for tool
576     Te = 22;
577     B = obj.frame.*obj.biImageTool > obj.validTemperature;
578     B1 = obj.frame.*B;
579     B2 = polyval(cp,B1);%Heat capacity for each pixel (J/kgK)

```

```

579         B2(B2 == cp(4)) = 0;
580         H = B2.*(obj.frame - Te)*(pp^2)*100;%Heat Amount for each pixel(J/m)
581         Ha = sum(sum(H));%Mean value for the entire tool
582         obj.InternalEnergyTool = Ha*w;
583     end
584
585     function B = passBinaryImageTool(obj)
586         B = obj.biImageTool;
587     end
588
589     function B = passBinaryImageChip(obj)
590         B = obj.biImageChip;
591     end
592
593     function obj = shearLine(obj)
594         B = obj.biImageChip;
595         v1 = sum(B);
596         v1(v1 == 0) = [];
597         l1 = length(v1);
598         C = imcrop(B,[20 20 l1 100]);
599         [m,n] = size(C);
600         pto = zeros(1000,2);
601         count = 1;
602         for i = 2:m-1
603             for j = 2:n-1
604                 if C(i,j+1) == 1 && C(i,j-1) == 1 && C(i+1,j) == 1 && C(i-1,j)
== 1
605                     pto(count,:) = [i j];
606                     count = count + 1;
607                 end
608             end
609         end
610         for i = 1000:-1:1
611             if isequal(pto(i,:),[0 0]) == 1
612                 pto(i,:) = [];
613             end
614         end
615         l = size(pto,1);
616         for i = 1:l
617             C(pto(i,1),pto(i,2)) = 0;
618         end
619
620         [H, THETA, RHO] = hough(C,'Theta',-40:-30);%Hough transformation
621         P = houghpeaks(H, 5);
622         lin = houghlines(C, THETA, RHO, P, 'FillGap', 15,'MinLength',10);
623         l=length(lin);
624         p1 = [];
625         p2 = [];
626         for i=1:l
627             Theta=lin(i).theta;
628             t1 = lin(i).point1;
629             t2 = lin(i).point2;
630             y = abs(t1(2)-t2(2));
631             if isempty(p1) && isempty(p2) && abs(Theta + 34) < 5
632                 p1 = t1 + [19 19];
633                 p2 = t2 + [19 19];
634                 ym = y;
635                 obj.ShearAngle = abs(Theta);
636             end
637             if abs(Theta + 34) < 5 && y > ym
638                 p1 = t1 + [19 19];
639                 p2 = t2 + [19 19];
640                 obj.ShearAngle = abs(Theta);
641             end
642         end
643         if isempty(obj.ShearAngle)
644             obj.ShearAngle = 30;
645         end
646     end
647

```

```

648     function obj = forcesValues(obj,Fp,Fq,w,tuc)
649         phi = obj.ShearAngle*pi/180;%shear angle
650         gamma = obj.RakeAngle*pi/180;%Rake angle
651         Fs = Fp*cos(phi) - Fq*sin(phi);%Cutting force component parallel to
shear plane
652         Ns = Fq*cos(phi) + Fp*sin(phi);%Cutting force component perpendicular to
shear plane
653         Fc = Fp*sin(gamma) + Fq*cos(gamma);%Cutting force component parallel to
tool face
654         Nc = Fp*cos(gamma) - Fq*sin(gamma);%Cutting force component
perpendicular to tool face
655         mu = Fc/Nc; % coefficient of friction
656         As = w*tuc/sin(phi);%Area shear plane
657         tau = Fs/As;%shear stress
658         sigma = Ns/As;%Normal stress
659         r = sin(phi)/cos(phi - gamma); %ratio r = t/tc = lc/l
660         ss = cos(gamma)/(sin(phi)*cos(phi-gamma));%shear strain
661         us = tau*ss;%shear energy per volume
662         uf = Fc*r/(tuc*w);%friction energy per volume
663         beta = atan(Fc/Nc);%friction angle on tool face
664         obj.CuttingForceParallelToolFace = Fc;
665         obj.CuttingForcePowerDirection = Fp;
666         obj.CuttingForceUncutChipThicknessDirection = Fq;
667         obj.CuttingForceParallelShearPlane = Fs;
668         obj.CuttingForcePerpendicularShearPlane = Ns;
669         obj.CuttingForcePerpendicularToolFace = Nc;
670         obj.CoefficientFriction = mu;
671         obj.ShearStress = tau;
672         obj.NormalStress = sigma;
673         obj.RatioR = r;
674         obj.ShearEnergyVolume = us;
675         obj.FrictionEnergyVolume = uf;
676         obj.FrictionAngle = beta*180/pi;
677     end
678
679     function obj = calculatePecletNumber(obj)
680         cp = polyval(obj.heatCapacity,obj.MaximumTemperatureCuttingZone);
681         k = 75.4;
682         d = 7.85*10^-3;
683         obj.PecletNumber = ((obj.CuttingVelocity/60)*obj.UnCutChipThickness)/(k
/(cp*d));
684     end
685
686     function vH = displayHeatCumulateperLine(obj)%Fix this function
687         d = zeros(size(obj.ptosLines,1),1);
688         d2 = zeros(size(obj.ptosLines,1)-1,1);
689         pp = 15*10^-3;%mm/pixel
690         for i = 1:size(obj.ptosLines,1)
691             d(i) = pp*((obj.CoordinateToolTip(1) - obj.ptosLines(i,1))^2 + ((obj
.CoordinateToolTip(2) - obj.ptosLines(i,2))^2))^(1/2);
692         end
693         for i = 1:size(obj.ptosLines,1)-1
694             d2(i) = pp*((obj.ptosLines(i+1,1) - obj.ptosLines(i,1))^2 + ((obj.
ptosLines(i+1,2) - obj.ptosLines(i,2))^2))^(1/2);
695         end
696         d2 = mean(d2)*10^-3;
697         figure
698         plot(d,obj.heatAccumulatedPerLine,'-x')
699         hold on
700         q = gradient(obj.heatAccumulatedPerLine,d2);
701         plot(d,q,'-r')
702         vH = [d obj.heatAccumulatedPerLine];
703     end
704 end
705 end

```


FOLHA DE REGISTRO DO DOCUMENTO

1. CLASSIFICAÇÃO/TIPO TC	2. DATA 21 de novembro de 2017	3. DOCUMENTO Nº DCTA/ITA/TC-031/2017	4. Nº DE PÁGINAS 39
5. TÍTULO E SUBTÍTULO: Analysis method of temperatures and heat flows for orthogonal cutting 1045 steel by thermal imaging			
6. AUTORA(ES): Adriana Nunes Chaves Lima			
7. INSTITUIÇÃO(ÕES)/ÓRGÃO(S) INTERNO(S)/DIVISÃO(ÕES): Instituto Tecnológico de Aeronáutica – ITA/IEM			
8. PALAVRAS-CHAVE SUGERIDAS PELA AUTORA: Manufacture; Image Processing; Matlab; Thermal Analysis; Orthogonal Cutting; Software.			
9. PALAVRAS-CHAVE RESULTANTES DE INDEXAÇÃO: Fabricação; Processamento de Imagens; Programas; Análise Térmica; Engenharia mecânica.			
10. APRESENTAÇÃO: (X) Nacional () Internacional ITA, São José dos Campos. Curso de Graduação em Engenharia Mecânica. Orientador: Anderson Vicente Borille; coorientador: Thorsten Augspurger. Publicado em 2017.			
11. RESUMO: <p>The handling of geometry files is usually associated with the usage of CAD software, which commonly have costly licenses due to the amount of sophisticated features that they offer. This expense is easily justifiable when the work involves thorough application of a number of these functions, for example, when developing a new geometry for a given project. However, for many users, the task is simple and only requires a small amount of features, for example, when generating a regular pattern of projections along a surface to obtain a set of points to guide the path of a turning or milling tool throughout a machining process. These clients usually have to spend a lot of money on licenses, only to use a small portion of what the program has to offer. This paper presents a solution which can directly read an IGES geometry file and generate a specific set of points using simple open source functions. Although it is programmed using <i>MATLABTM</i>, it is compiled to an independent executable file, thus being capable of running without any license at all.</p>			
12. GRAU DE SIGILO: <div style="display: flex; justify-content: space-around;">(X) OSTENSIVO() RESERVADO() SECRETO</div>			