Technische Universiteit Delft
Faculteit Elektrotechniek, Wiskunde en Informatica
Delft Institute of Applied Mathematics

Analysis of a streaming video feed of a pulverized coal injection system

(Analyse van een video van een verpulverd kool injectie systeem)

Verslag ten behoeve van het
Delft Institute of Applied Mathematics
als onderdeel ter verkrijging

van de graad van

BACHELOR OF SCIENCE
in
TECHNISCHE WISKUNDE

by

ELSEMIEK SMILDE

Delft, Nederland
August 2019

BSc report TECHNISCHE WISKUNDE

"Analysis of a streaming video feed of a pulverized coal injection system"
("Analyse van een video van een verpulverd kool injectie systeem")

ELSEMIEK SMILDE

Technische Universiteit Delft

Begeleider

Dr. N. V. Budko

Overige commissieleden

Dr. D. J. P. Lahaye

Drs. E. M. van Elderen

E. Tesselaar (Danieli Corus)

Augustus, 2019

Delft

Abstract

Danieli Corus supplies tailor-made solutions for the global steel industry. Cameras are used in tuyeres of blast furnaces to detect physical aspects and irregularities. No operator could monitor those videos all day, which is why video analyses are needed. A program is written to analyse a test video and give information about the temperature, the injection of pulverized coal and the movement of rocks. The mathematical background of these calculations is presented, which depend on the transport of light from the blast furnace to the camera through the tuyere. Also the implementation of the calculations can be found. The results are interpreted and discussed and further improvements are suggested.

Preface

This report is written as part of the bachelor program Applied Mathematics at the TU Delft. The project really sparked my interest of extracting information from a video and how this can be used and I will absolutely try to continue this interest during my Master Applied Mathematics. I would like to thank Neil Budko for all the help and ideas during our weekly meetings and Domenico Lahaye and Ewout Tesselaar for the feedback later in the process.

I hope my enthusiasm is found back in the report and I hope you enjoy reading it.

Elsemiek Smilde
Delft, 20 augustus 2019

Contents

Introduction and Danieli Corus

Danieli Corus supplies tailor-made solutions for the global steel industry. One of these solutions are Pulverized Coal Injection Systems. These systems involve blowing large volumes of fine coal granulates into blast furnaces. This speeds up the production of metallic iron, reducing the need for coke production. As a result energy use and emissions can be reduced (Danieli-Corus, 2019).

Cameras are installed in these blast furnaces to keep an eye on this process. The goal of this project is to find out what usefull information can be abstracted from these videos. In Figure 1 there is a screenshot from the video that is used during this project.

Danieli Corus already made up seven focus points that they want to obtain from the videos. These focus points are taken into account, but are not the main focus of this project.

1. Blackness of the image at coal injection
2. Motion detection
3. Tilting tuyeres detection
4. Broken coal lance detection / wear coal lance detection
5. Water leakage inside tuyere detection
6. Sharp white light (natural gas blockage)
7. Light intensity as temperature indication

Figure 1: Screenshot from the Danieli Corus video feed

Chapter 1

Literature review

In this chapter more information about the cameras and existing techniques to extract information from similar videos will be provided.

1.1 Tuyere cameras

The tuyeres where the cameras are placed are part of a bigger system in which iron is melted in the blast furnace. This process is shown in Figure 1.1. The coal enters the blast furnace together with hot air through *tuyeres*, which are cylinder shaped passages. Inside the blast furnace the layers of iron ore and coke are moving down while the furnace is active. Because of the air flow these layers do not come (too) close to the tuyere. The cameras from which the test videos are originating are placed just behind these tuyeres, recording the coal flow going in through a pipe and the movement in the background.

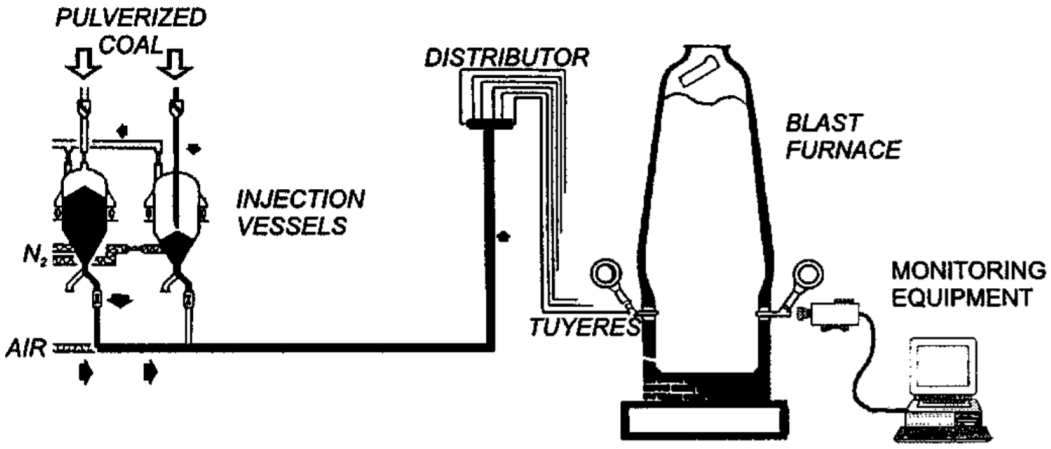
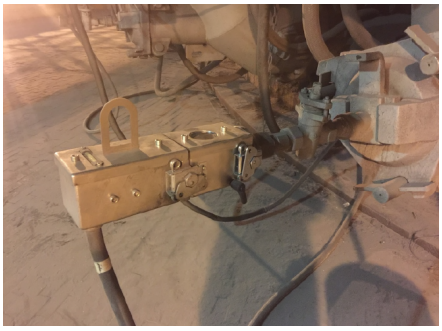
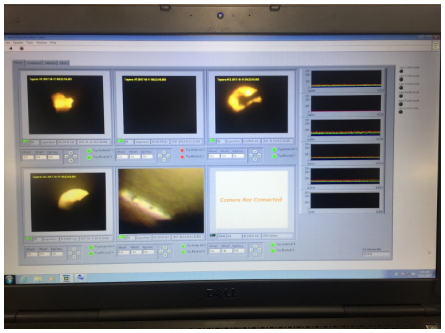


Figure 1.1: Overview of the blast furnace, the pulverid coal injection system and tuyere cameras (*Source*: Birk et al. (2002))

The cameras are placed at the elbow of the tuyere and are housed in a protecting housing. This is shown in Figure 1.2a. Through the hole in the topside of the housing, controllers can check the camera without removing the housing. There are usually around 30 tuyeres in one blast furnace. The result of the cameras in the control room are shown in Figure 1.2b.



(a) A housed camera placed at a tuyere



(b) Output of camera on screen in control room

Figure 1.2: a) Camera and b) output (*Source*: Danieli Corus presentation - Tuyere Camera)

1.2 Existing possibilities

In the past time Paul Wurth - active in technological solutions for the primary stage of integrated steelmaking - has also developed monitoring technology for the tuyere cameras, called the *Tuyere Phenomena Detection System* (Paul-Wurth, 2014). These technologies detect relevant phenomena, which is connected to additional emergency measures (for instance, immediate shut-off of powder coal injection). In this way, the technology contributes to an optimal operation of the blast furnace (Simoes et al., 2012).

The phenomena that can be detected by this system are the following (Simoes et al., 2012):

- Blockage of the tuyere
- Tuyere movement
- Injection lance movement or bending
- Injection lance back burning or breaking
- Injection status
- Ignition state of oxy-coal lances
- Ash or slag deposits at the tuyere tip

Before the calculation of these phenomena, a reference procedure is run, supplying the system with reference data. During this procedure, the areas in the frame are determined where the phenomena take place. These areas are called Regions of Interest (ROI). The circle that forms the tuyere tip - where the light is coming from - is found by looking at points with a high gradient in light intensity. A circle is fitted through these points. The pipe that is visible in the frame is also found in a comparable way. By looking at the pixel values and applying treatments and algorithms in the determined Regions of Interest the phenomena in the list can be calculated.

The calculations of the phenomena are carried out using pictures that are send to the detection service from each tuyere camera. This happens at fixed intervals. By comparing the obtained results with previous results and the reference data the accuracy of the calculations is improved. The data is also compared to other measurements made in the blast furnace, which also contributes to the reliability of the detection.

Stelco, a canadian steel company, also uses tuyere camera technology, produced by Shenwang Pioneer Technology Corporation. The purpose of the cameras and the technology, which were installed in 2011, was originally to detect blockage in the blast furnace. Since then the possibilities grew and now it can be used for multiple purposes in monitoring and controlling the furnace

operation (Ward et al., 2017). The image analysis software provides output of a parameter called the burn ratio and the temperature. The burn ratio is the result of the amount of dark area in the raceway region. The raceway region is the area where the light from the blast furnace is coming from. A higher percentage of dark area will result in a higher value for burn ratio. The temperature is calculated with the help of infrared images. All information is also stored and can be used in further research regarding the process in the blast furnace. This helps for example in finding the relationship between key operating parameters.

1.3 Theoretical approaches

1.3.1 Localize regions of interest

Birk et al. (2002) suggest a method to find the area in which the coal particles are moving by looking at the areas where the frames are moving. In other words to compute an image $D = \sum_\tau |F(\tau) - F(\tau - 1)|$, where τ is a discrete time-parameter. After this the areas with the highest values in D can be taken into account. They call this area R_τ. In this way the furnace wall and the tuyere will be excluded from the analysed area.

To find the area where the coal particle flow is visible Birk et al. look at the grayscale histograms of the area R_τ. They found two well seperated 'hills', which represent the bright background and the dark coal powder flow. After choosing a treshold value, the area where the powder is visible is found and called P_τ.

Figure 1.3: Background for comparison (*Source*: (Wang et al., 2015))

Wang et al. (2015) suggest a method in which the coal powder area is detected by comparing the frames to a precomposed background area, shown in Figure 1.3.

1.3.2 Estimation of physical quantities

To calculate the mass flow of coal particles the following formula can be used.

$$m(t) = \sum_{(i,j)\in P_\tau} f(R_\tau, i, j, \tau) \tag{1.1}$$

where the optimal form of $f(\cdot)$ is still under discussion (Birk et al., 2002). The most straightforward choice is $f(\cdot) = A$, where A is a constant, i. e. a value directly proportional to the total area covered by the pixels in P is used as relative estimate. A more sophisticated choice is $f(\cdot) = f_\alpha(\cdot) = A_\alpha[R(i, j, \tau) + \epsilon]^{-\alpha}$ where A_α, α, and ϵ are constants greater than zero. The reason for this choice is to give coordinates in P_τ with a low grayscale value a higher weight, since dark pixels suggest regions with higher coal powder density.

Wang et al. (2015) propose a different method to assign weights to pixel points, which uses the minimum and maximum value of the grayscale pixels as reference and applies linear weights. The linear weight function will be: $f(R) = kR + b$ for grayscale pixelvalues of R.

Chapter 2

Properties of the video

In this chapter properties of the video will be discussed. First we will take a look at what exactly can be seen in the video and how the correct regions of interest (ROI's) can be found for the calculation of physical quantities. Afterwards, the number of frames and the number of pixels will be discussed.

2.1 Localization of regions of interest

In the video a Pulverized Coal Injection System is shown (see Figure 1). The pipe that transports the coal is visible, ass well as the coal coming out of it. In the background we see light coming from the working blast furnace and moving rocks are visible. The inside of the cylinder in which the coal is injected is also visible. In Figure 2.1 the situation is simplified and visualized.

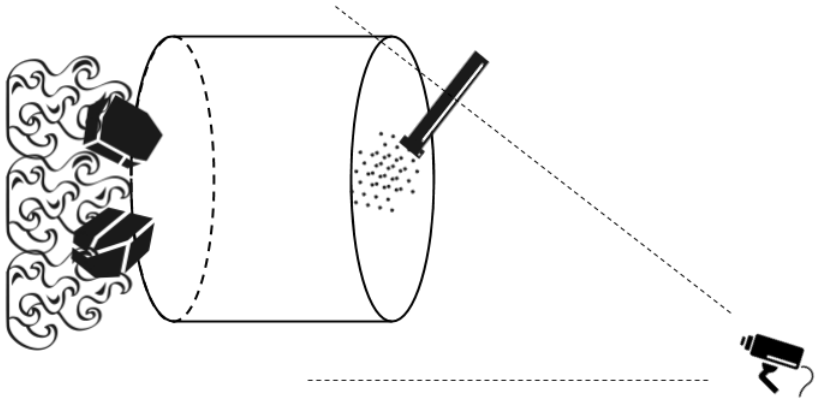


Figure 2.1: Simplified overview of situation

To determine the interesting areas for the calculations we will first try to find the ellipses that correspond to the partially visible lower end upper ends of the tuyere's truncated cone

2.1.1 Localization of ellipses

To determine the ellipses in the frame a method is used, where points on the ellipses - visible in the video - can be assigned and the best possible ellipse fit will be generated based on those points.

General form of an ellipse

The general form of an ellipse is as follows:

$$\frac{(x-x_c)^2}{a^2} + \frac{(y-y_c)^2}{b^2} = 1 \tag{2.1}$$

where x_c, y_c are the coördinates of the center of the ellipse and a and b are the lengths of the axes. This is shown in Figure 2.2. An ellipse can also be rotated by an angle, which influences the equation. If the ellipse is rotated by an angle θ the equation becomes:

$$\frac{((x-x_c)\cos\theta + (y-y_c)\sin\theta)^2}{a^2} + \frac{((x-x_c)\sin\theta - (y-y_c)\cos\theta)^2}{b^2} = 1 \tag{2.2}$$

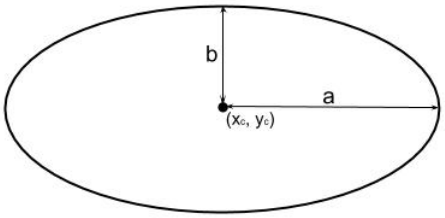


Figure 2.2: General form of an ellipse

The ellipse of the cylinder that is closest to the camera is shown most clearly in the video. For that reason this ellipse is first approximated. Based on the first ellipse there is already information about the second ellipses. It is known that the ellipses are actually the same shape. The ellipse that is closer to the blast furnace is portrayed smaller and in a different position, due to the position of the camera. The ratio between a and b is still the same. For this reason, the equation for the lower ellipse is as follows:

$$\frac{(x-x_c)^2}{a^2} + \frac{(y-y_c)^2}{(ka)^2} = 1 \tag{2.3}$$

where k is the ratio $\frac{b}{a}$ from the determined upper ellipse.

In the next section we show how the ellipses can be found based on chosen points.

Least squares fitting

To find the best fitting ellipses from chosen points the least squares fitting method is used, proposed by Halir and Flusser (1998).

An ellipse can be described by

$$F(x,y) = ax^2 + bxy + cy^2 + dx + ey + f = 0 \tag{2.4}$$

with the constraint that $b^2 - 4ac > 0$. Here, (x, y) are coordinates of points on the ellipse and a, b, c, d, e, f are coefficients of the ellipse. $F(x, y)$ is the algebraic distance of the point (x, y) and the ellipse.

We can write equation (2.4) alternatively:

$$F(x, y) = \mathbf{D} \cdot \mathbf{a} = 0 \tag{2.5}$$

where

$$\mathbf{D} = (x^2, xy, y^2, x, y, 1)$$
$$\mathbf{a} = (a, b, c, d, e, f)$$

The best ellipse through a data-set of N points (x_i, y_i) with $i = 1, 2, ..., N$ may be approached by finding the coefficients $\mathbf{a}$ that minimize the sum of the squared distances between the points and the corresponding ellipse.

$$\min_{\mathbf{a}} \sum_{i=1}^{N} (F(x_i, y_i))^2 = \min_{\mathbf{a}} \sum_{i=1}^{N} (\mathbf{D} \cdot \mathbf{a})^2 \tag{2.6}$$

To make sure that the resulting $\mathbf{a}$ produces an ellipse equation the constraint $b^2 - 4ac > 0$ should be taken into account. We consider $b^2 - 4ac = 1$. The problem can now be reformulated as

$$\min_{\mathbf{a}} \sum_{i=1}^{N} (\mathbf{D} \cdot \mathbf{a})^2 \text{ subject to } \mathbf{a}^T \mathbf{C} \mathbf{a}.$$

where

$$\mathbf{D} = \begin{pmatrix} x_1^2 & x_1 y_1 & y_1^2 & x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_i^2 & x_i y_i & y_i^2 & x_i & y_i & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_N^2 & x_N y_N & y_N^2 & x_N & y_N & 1 \end{pmatrix} \text{ and } \mathbf{C} = \begin{pmatrix} 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.7}$$

Applying Lagrange multipliers we get

$$\mathbf{S}\mathbf{a} = \lambda \mathbf{C}\mathbf{a} \tag{2.8}$$
$$\mathbf{a}^{\mathrm{T}} \mathbf{C} \mathbf{a} = 1 \tag{2.9}$$

where $\mathbf{S} = \mathbf{D}^{\mathrm{T}} \mathbf{D}$.

We are looking for the eigenvector $\mathbf{a}_k$ corresponding to the minimal positive eigenvalue λ_k. Then the resulting $\mathbf{a}_k$ should be scaled to make sure that $\mathbf{a}_k^{\mathrm{T}} \mathbf{C} \mathbf{a}_k = 1$. With this $\mathbf{a}_k = (a_k, b_k, c_k, d_k, e_k, f_k)$ the properties of the ellipse can be calculated as follows (with $\mathbf{a}_k = (a, b, c, d, e, f)$):

$$
\begin{aligned}
x_{center} &= \frac{c\frac{d}{2} - \frac{b}{2}\frac{f}{2}}{\frac{b}{2}\frac{b}{2} - ac} \\
y_{center} &= \frac{a\frac{f}{2} - \frac{b}{2}\frac{d}{2}}{\frac{b}{2}\frac{b}{2} - ac} \\
\text{length of axis 1 } (a) &= \sqrt{\frac{2(a\frac{f}{2}\frac{f}{2} + c\frac{d}{2}\frac{d}{2} + g\frac{b}{2}\frac{b}{2} - 2\frac{b}{2}\frac{d}{2}\frac{f}{2} - acg)}{(\frac{b}{2}\frac{b}{2} - ac)((c-a)\sqrt{1 + 4\frac{b}{2}\frac{b}{2}/((a-c)(a-c))} - (c+a))}} \\
\text{length of axis 2 } (b) &= \sqrt{\frac{2(a\frac{f}{2}f + c\frac{d}{2}\frac{d}{2} + g\frac{b}{2}\frac{b}{2} - 2\frac{b}{2}\frac{d}{2}\frac{f}{2} - acg)}{(\frac{b}{2}\frac{b}{2} - ac)((a-c)\sqrt{1 + 4\frac{b}{2}\frac{b}{2}/((a-c)(a-c))} - (c+a))}} \\
\text{angle } (\theta) &= \frac{\arctan(\frac{2\frac{b}{2}}{a-c})}{2}
\end{aligned}
\tag{2.10}
$$

Improved least squares fitting

The method described up untill now is theoretically correct, but because of the singularity of the matrix $\mathbf{C}$ and the near-singularity of the matrix $\mathbf{S}$ the computation of eigenvalues can yield wrong results. Both matrices have special structures which allow a simplification of the problem of finding the eigenvalues. The matrices $\mathbf{D}$, $\mathbf{C}$ and $\mathbf{S}$ are split up as follows:

$$
\mathbf{D} = (\mathbf{D}_1 | \mathbf{D}_2) \tag{2.11}
$$

where

$$
\mathbf{D}_1 = \begin{pmatrix} x_1^2 & x_1 y_1 & y_1^2 \\ \vdots & \vdots & \vdots \\ x_i^2 & x_i y_i & y_i^2 \\ \vdots & \vdots & \vdots \\ x_N^2 & x_N y_N & y_N^2 \end{pmatrix} \quad \text{and} \quad \mathbf{D}_2 = \begin{pmatrix} x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots \\ x_i & y_i & 1 \\ \vdots & \vdots & \vdots \\ x_N & y_N & 1 \end{pmatrix} \tag{2.12}
$$

$$
\mathbf{S} = \left(\begin{array}{c|c} \mathbf{S}_1 & \mathbf{S}_2 \\ \hline \mathbf{S}_2^{\mathrm{T}} & \mathbf{S}_3 \end{array}\right) \quad \text{where} \quad \begin{aligned} \mathbf{S}_1 &= D_1^{\mathrm{T}} D_1 \\ \mathbf{S}_2 &= D_1^{\mathrm{T}} D_2 \\ \mathbf{S}_3 &= D_2^{\mathrm{T}} D_2 \end{aligned} \tag{2.13}
$$

$$
\mathbf{C} = \left(\begin{array}{c|c} \mathbf{C}_1 & 0 \\ \hline 0 & 0 \end{array}\right) \quad \text{where} \quad \mathbf{C}_1 = \begin{pmatrix} 0 & 0 & 2 \\ 0 & -1 & 0 \\ 2 & 0 & 0 \end{pmatrix} \tag{2.14}
$$

$$
\mathbf{a} = \left(\frac{\mathbf{a}_1}{\mathbf{a}_2}\right) \quad \text{where} \quad \mathbf{a}_1 = \begin{pmatrix} a \\ b \\ c \end{pmatrix} \quad \text{and} \quad \mathbf{a}_2 = \begin{pmatrix} d \\ e \\ f \end{pmatrix} \tag{2.15}
$$

Now equation (2.8) can be rewritten as

$$
\left(\begin{array}{c|c} \mathbf{S}_1 & \mathbf{S}_2 \\ \hline \mathbf{S}_2^{\mathrm{T}} & \mathbf{S}_3 \end{array}\right) \cdot \left(\frac{\mathbf{a}_1}{\mathbf{a}_2}\right) = \lambda \cdot \left(\begin{array}{c|c} \mathbf{C}_1 & 0 \\ \hline 0 & 0 \end{array}\right) \cdot \left(\frac{\mathbf{a}_1}{\mathbf{a}_2}\right) \tag{2.16}
$$

which is equivalent to

$$\mathbf{S}_1\mathbf{a}_1 + \mathbf{S}_2\mathbf{a}_2 = \lambda\mathbf{C}_1\mathbf{a}_1 \tag{2.17}$$

$$\mathbf{S}_2^{\mathrm{T}}\mathbf{a}_1 + \mathbf{S}_3\mathbf{a}_2 = 0 \tag{2.18}$$

The matrix $\mathbf{S}_3$ is not singular, except when the points all lay on a line, in which case it is not useful to fit an ellipse through the points. Now $\mathbf{a}_2$ can be expressed from equation (2.18) as

$$\mathbf{a}_2 = -\mathbf{S}_3^{-1}\mathbf{S}_2^{\mathrm{T}}\mathbf{a}_1 \tag{2.19}$$

Then (2.17) gives

$$\left(\mathbf{S}_1 - \mathbf{S}_2\mathbf{S}_3^{-1}\mathbf{S}_2^{\mathrm{T}}\right)\mathbf{a}_1 = \lambda\mathbf{C}_1\mathbf{a}_1 \tag{2.20}$$

which can be rewritten to

$$\mathbf{C}_1^{-1}\left(\mathbf{S}_1 - \mathbf{S}_2\mathbf{S}_3^{-1}\mathbf{S}_2^{\mathrm{T}}\right)\mathbf{a}_1 = \lambda\mathbf{a}_1 \tag{2.21}$$

The condition in equation (2.9) can also be rewritten with the help of decomposition.

$$\mathbf{a}_1^{\mathrm{T}}\mathbf{C}_1\mathbf{a}_1 = 1 \tag{2.22}$$

This finally results in the following set of equations:

$$\begin{aligned} \mathbf{M}\mathbf{a}_1 &= \lambda\mathbf{a}_1 \\ \mathbf{a}_1^{\mathrm{T}}\mathbf{C}_1\mathbf{a}_1 &= 1 \\ \mathbf{a}_2 &= -\mathbf{S}_3^{-1}\mathbf{S}_2^{\mathrm{T}}\mathbf{a}_1 \\ \mathbf{a} &= \begin{pmatrix}\mathbf{a}_1 \\ \mathbf{a}_2\end{pmatrix} \end{aligned} \tag{2.23}$$

where

$$\mathbf{M} = \mathbf{C}_1^{-1}\left(\mathbf{S}_1 - \mathbf{S}_2\mathbf{S}_3^{-1}\mathbf{S}_2^{\mathrm{T}}\right) \tag{2.24}$$

Now the appropriate eigenvector $\mathbf{a}$ of the matrix $\mathbf{M}$ must be found in the same way as before and the properties of the ellipse can be derived from this $\mathbf{a}$.

2.1.2 Defining the regions of interest

After the localization of the ellipses some other shapes in the frame have to be determined, after which the regions of interest can be set up.

Pipe

With the intersection of the two ellipses we don't have the region of interest. The reason for this is the presence of the pipe in the frames. The pipe is approximated by a quadrangle, formed by the four corner points. The region of interest can now be described by the intersection of the two ellipses and the area outside the pipe, see Figure 2.3.

Background and coal areas

In Figure 1 it is clearly visible that the pulverized coal is injected on the right side of the cylinder. It is observed that this holds for every frame in the video, and the region of interest is split up again. The left side is referred to as the *background* and the right side is referred to as the *coal*

area. These areas are shown in Figure 2.3. The blue ellipse is further referenced to as E_1, the red ellipse as E_2 and the green shape as P.

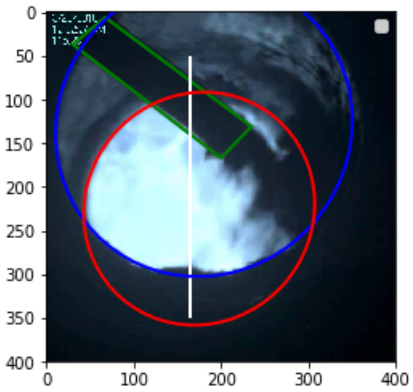
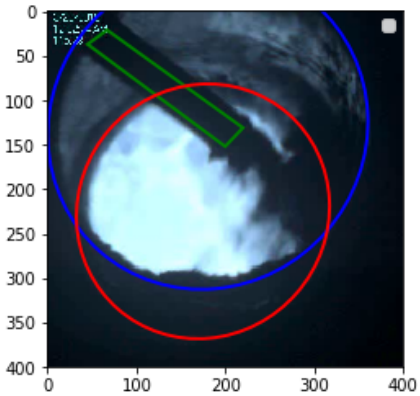


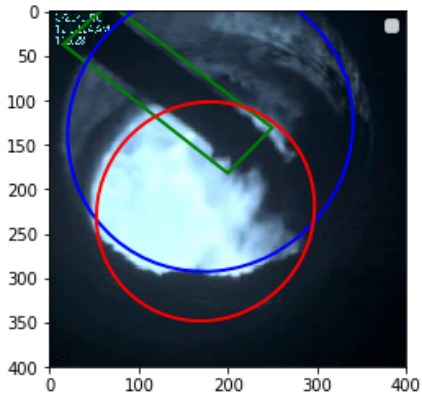
Figure 2.3: Founded area's displayed in video

Boundaries

The high temperature in the blast furnace causes the air to vibrate, which causes transient wave-like deformations moving across the video frame. These heat waves make the boundaries of the ROI's oscillate around their average location. To inspect these heat-induced oscillations first the boundary neighborhoods are introduced. By inspection the width of these neighbourhoods are set to twenty pixels. The boundary neighborhoods are created by extending and shrinking the original ROI's and taking the geometrical difference. The extended and reduced ROI's are shown in Figure 2.4. The extended ellipses and pipe are referred to as E_1^+, E_2^+ and P^+, the reduced ellipses and pipe are referred to as E_1^-, E_2^- and P^-.



(a) Extended area - bigger ellipses but smaller pipe



(b) Reduced area - smaller ellipses but bigger pipe

Figure 2.4: a) Extended and b) reduced area

Since heat waves move in different directions, the boundaries are split in four area's, which are shown in Figure 2.5 in different colours. These heat-induced oscillations are further inspected in Chapter 4.4

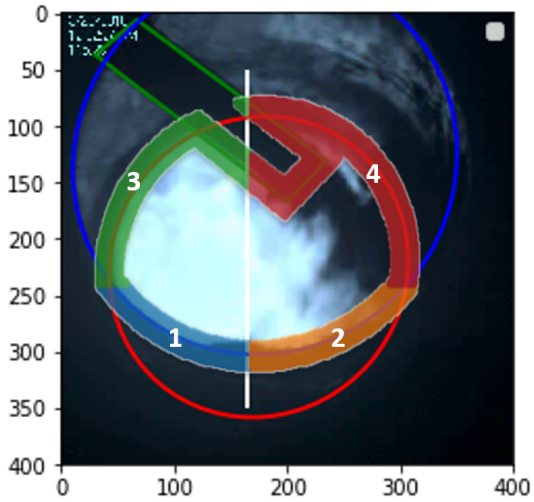


Figure 2.5: Boundaries of interest area

Summary of regions of interest

In the following table the sets of points belonging to the areas are summed up to provide an overview.

$A_{background}$	$(i,j) \in E_1 \cap E_2 \cap P^c$ with $i < x_{line}$
$A_{coal-area}$	$(i,j) \in E_1 \cap E_2 \cap P^c$ with $i > x_{line}$
$A_{boundary}$	$(i,j) \in E_1^+ \cap E_2^+ \cap (P^-)^c \cap (E_1^- \cap E_2^- \cap (P^+)^c)^c$

2.2 Number of frames and pixels

The number of frames per time unit and the number of pixels per frame have an influence on the program performance and should thus be considered.

Number of frames

Next to the regions of interest there should also be decided how many frames will be analysed. In other words, how many time there will be between each frame. The analysing time for each frame plays an important role in this decision. Ward et al. (2017) suggest that one frame per second is used in similar operating systems. To make this work we must make sure that the analysing time of a frame will be less than one second. If more frames could be analysed this could improve the calculations even more.

Number of pixels

The last thing to note about the videos is the number of pixels in the frame. In the example video this amount is equal to 400x400, which means 160000 pixels. For frames with more pixels, the processing time per frame could increase.

Chapter 3

Theory

After the determination of the regions of interest (see Section 2.1) a series of calculations can be performed with the goal to try to estimate the temperature, the amount of pulverized coal and the movement of rocks in the video. These calculations all depend on the amount of light from the blast furnace that reaches the camera and that the camera can detect.

3.1 Light transport model

The process in the blast furnace emits a lot of light. This light comes from the burning layers of ore and cokes and reaches the camera through the opening of the tuyere, as already shown in Figure 2.1. The quantitative analysis of the physical phenomena makes use of the amount of light that reaches the camera. Some of the light will be blocked by the pipe, by coal particles or by rocks. We apply the Beer-Lambert law to model the light transport between the hot light-emitting surface inside the oven and the camera. The law describes the relationship between the transmittance of light through a material, the attenuation cross section, the length of the optical path through the material and the number density of the species along the path. The path where the light goes through is shown in Figure 3.1.

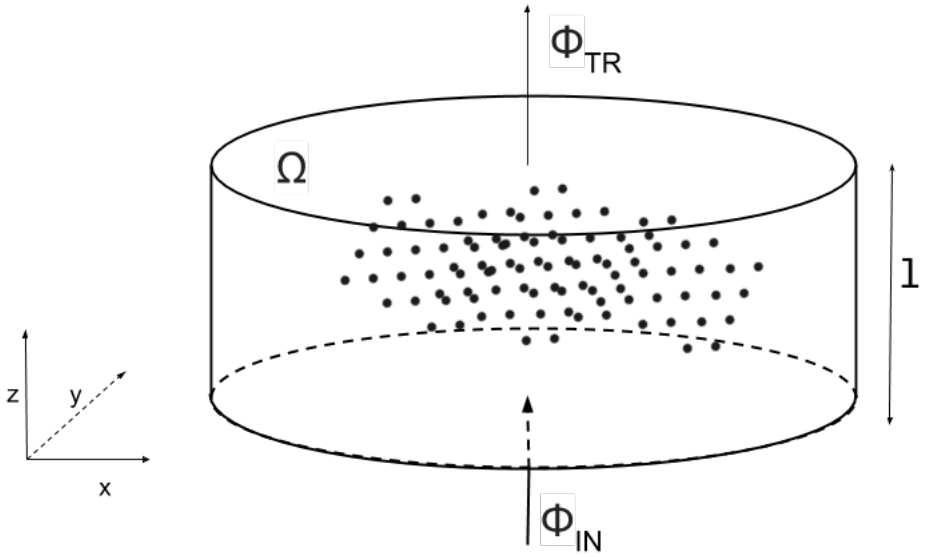


Figure 3.1: Illustration of the law of Beer-Lambert through the cylinder

The law is as follows:

$$\frac{\Phi_{TR}}{\Phi_{IN}} = e^{-\sum_{i=1}^{N} \sigma_i \int_0^l n_i(z)\mathrm{d}z} \tag{3.1}$$

where:
Φ_{IN} is the incoming light,
Φ_{TR} is the detected light,
N is the number of attenuating species in the material sample,
σ_i is the attenuation cross section of the attenuating species i in the material sample,
l is the length of the path through the material,
n_i is the number density of the attenuating species i in the material sample.

3.2 Pulverized coal mass estimation

Multiple ways to estimate the amount of coal in the frame have already been tried, as described in Section 1.3.2. None of these approaches takes into account what the amount of light is that is coming from the furnace in the background, while this obviously has an influence on the amount of light that will (not) reach the tuyere camera. Therefore our approach is a little different in this respect.

To estimate the amount of pulverized coal in the video at a given moment, one has to compare the amount of light that was emitted by a hot surface in the furnace to the amount of light that has reached the camera. The light that will not reach the camera will be blocked by coal. This process is described in the law of Beer-Lambert.

In this case only one species, pulverized coal, is considered, which makes it possible to simplify the equation to

$$\frac{\Phi_{TR}}{\Phi_{IN}} = e^{-\sigma_{pc} \int_0^l n_{pc}(z)\mathrm{d}z} \tag{3.2}$$

where:
σ_{pc} is the attenuation cross section of pulverized coal,
n_{pc} is the number density of pulverized coal.

This can be rewritten as

$$\begin{aligned} \Phi_{TR} &= \Phi_{IN}\, e^{-\sigma_{pc} \int_0^l n_{pc}(z)\mathrm{d}z} \\ \ln\left(\Phi_{TR}\right) &= \ln\left(\Phi_{IN}\right) - \sigma_{pc} \int_0^l n_{pc}(z)\mathrm{d}z \\ \int_0^l n_{pc}(z)\mathrm{d}z &= \frac{\ln\left(\Phi_{IN}\right) - \ln\left(\Phi_{TR}\right)}{\sigma_{pc}} \end{aligned} \tag{3.3}$$

At the same time, we know that

$$\begin{aligned} m(t) &= \iiint_V m_{pc}\, n(\underline{x}, t)\mathrm{d}V \\ &= \iint_\Omega \int_o^l m_{pc}\, n(\underline{x}, t)\mathrm{d}z\mathrm{d}A \end{aligned} \tag{3.4}$$

where:
$m(t)$ is the mass of pulverized coals inside the cylinder at moment t,
V is the volume of the cylinder,
m_{pc} is the molar mass of coal,
$n(\underline{x}, t)$ is the number density of pulverized coal particles at place $\underline{x}$ at time t.

Inserting the last equation (3.3) into the last equation (3.4) the following is obtained:

$$m(t) = \frac{m_{pc}}{\sigma} \iint_\Omega \ln \frac{\Phi_{IN}(x, y, t)}{\Phi_{TR}(x, y, t)}\mathrm{d}A \tag{3.5}$$

Assuming that $\frac{m_{pc}}{\sigma}$ is constant, we find that

$$m(t) \sim \iint_\Omega \ln \frac{\Phi_{IN}(x, y, t)}{\Phi_{TR}(x, y, t)}\mathrm{d}A \tag{3.6}$$

In the processing of the video, Φ_{TR} can be defined as the light that reaches the camera. To find the value of Φ_{IN} we look at two different areas: *background* and *coal area*, as seen in Figure 2.3. We define $S_{background}$ as the number of pixels $(i, j) \in A_{background}$. We define $S_{coal-area}$ as the number of pixels $(i, j) \in A_{coal-area}$. Assuming pulverized coal particles are the only disturbance of light, we find for $A_{background}$ that $\Phi_{TR}(x, y, t) = \Phi_{IN}(x, y, t)$, since there are no coal particles in the way from the *background* to the camera, so $\frac{\Phi_{TR}(x,y,t)}{\Phi_{IN}(x,y,t)} = 1$ and $\ln \frac{\Phi_{TR}(x,y,t)}{\Phi_{IN}(x,y,t)} = 0$.

Furthermore, we assume the spatial average of $\Phi_{IN}(x, y, t)$ to be equal in $A_{background}$ and $A_{coal-area}$, since the light comes from statistically the same fire. This means

$$\frac{1}{S_{background}} \iint_{A_{background}} \ln \Phi_{IN}(x, y, t)\mathrm{d}A = \frac{1}{S_{coal-area}} \iint_{A_{coal-area}} \ln \Phi_{IN}(x, y, t)\mathrm{d}A \tag{3.7}$$

and so

$$\begin{aligned} \iint_{A_{coal-area}} \ln \Phi_{IN}(x, y, t)\mathrm{d}A &= \frac{S_{coal-area}}{S_{background}} \iint_{A_{background}} \ln \Phi_{IN}(x, y, t)\mathrm{d}A \\ &= \frac{S_{coal-area}}{S_{background}} \iint_{A_{background}} \ln \Phi_{TR}(x, y, t)\mathrm{d}A \end{aligned} \tag{3.8}$$

Equation (3.6) can be rewritten as follows:

$$
\begin{aligned}
m(t) &\sim \iint_{A_{background}} \ln \frac{\Phi_{IN}(x,y,t)}{\Phi_{TR}(x,y,t)} \mathrm{d}A + \iint_{A_{coal-area}} \ln \frac{\Phi_{IN}(x,y,t)}{\Phi_{TR}(x,y,t)} \mathrm{d}A \\
&= \iint_{A_{coal-area}} \ln \frac{\Phi_{IN}(x,y,t)}{\Phi_{TR}(x,y,t)} \mathrm{d}A \\
&= \frac{S_{coal-area}}{S_{background}} \iint_{A_{background}} \ln \Phi_{TR}(x,y,t) \mathrm{d}A - \iint_{A_{coal-area}} \ln \Phi_{TR}(x,y,t) \mathrm{d}A
\end{aligned} \tag{3.9}
$$

In discretized form this results in

$$
m(t) \sim \frac{S_{coal-area}}{S_{background}} \sum_{(i,j) \in A_{background}} \ln \Phi_{i,j}(t) - \sum_{(i,j) \in A_{coal-area}} \ln \Phi_{i,j}(t) \tag{3.10}
$$

3.3 Temperature estimation

Here two ways to estimate the temperature in the blast furnace using the video frames are proposed. The first uses the amount of light in the background and the other uses the frequency of the heat-induced oscillations.

3.3.1 Background light intensity

It is assumed that when the temperature in the blast furnace increases, the amount of light coming from it also increases. The area where this is measured should not be influenced by pulverized coal, which disturbs the transport of light as described in Section 3.2. Therefore, the *background* area is taken for this calculation. First the total light intensity that reaches the camera at time t from the *background* area is determined by taking the integral of the intensity over the area:

$$
\text{Light intensity(t)} \sim \int_{A_{background}} \Phi(x,y,t) \tag{3.11}
$$

where $\Phi(x,y)$ is the light intensity at place (x,y)

In discretized form the formula is the following, with $\Phi_{i,j}$ the pixel value of the pixel i,j.

$$
\text{Light intensity(t)} \sim \sum_{(i,j) \in A_{background}} \Phi_{i,j}(t) \tag{3.12}
$$

After computing the total light intensity coming from the background, the temperature can be estimated. Here it should be taken into account that the camera can only detect light that has a wavelength between 380 nm and 780 nm (the visual spectrum). The relationship between the light intensity and temperature is described in the law of Stefan-Bolzmann:

$$
j^\star = \sigma T^4 \tag{3.13}
$$

where
$j^\star$ is the total energy radiated per unit surface area of a black body across all wavelengths per unit time,

σ is the StefanBoltzmann constant: $5.670373 \times 10^{-8}\,\mathrm{W\,m^{-2}K^{-4}}$ and
T is the thermodynamic temperature.

A normal (not infrared) camera is used, which can not detect all wavelengths. The distribution of the wavelengths of the emitted light is influenced by the temperature. This is described in Planck's law:

$$B_\lambda(\lambda, T) = \frac{2hc^2}{\lambda^5} \frac{1}{e^{\frac{hc}{\lambda k_\mathrm{B} T}} - 1} \tag{3.14}$$

where
B_λ is the spectral radiance,
λ is the wavelength,
T is the absolute temperature,
k_B is the Boltzmann constant,
h is the Planck constant,
and c is the speed of light in the medium.

With this law the amount of energy that is emitted at different wavelengths at a given temperature can be determined. Using this information, a better estimation of the temperature could be computed from the light intensity from the background. For now, we use equation (3.13) and estimate j^* with the energy radiated in the visual spectrum, which is measured as the light intensity in the background. This gives us the following formula:

$$\begin{aligned} \sigma T^4 &= j^\star_{visualspectrum} \\ T^4 &= \frac{1}{\sigma} j^\star_{visualspectrum} \\ T &= \sqrt[4]{\frac{1}{\sigma}} \sqrt[4]{j^\star_{visualspectrum}} \end{aligned} \tag{3.15}$$

This gives us:

$$T \sim \sqrt[4]{j^\star_{visualspectrum}} \tag{3.16}$$

$$T \sim \sqrt[4]{\sum_{(i,j) \in A_{background}} \Phi_{i,j}(t)} \tag{3.17}$$

3.3.2 Heat-induced oscillations

As seen in Chapter 2.1.2, for the oscillations the neighbourhood of the boundaries are considered to see if there is a connection between oscillations and temperature. The boundaries are shown in Figure 2.5. The four area's are called $A_{boundary-i}$ with $i \in \{1, 2, 3, 4\}$. For the four area's the sum of the pixel values is calculated at each time step:

$$\text{Boundary-}i \text{ light intensity } (t) = \sum_{(i,j) \in A_{boundary-i}} \Phi_{i,j}(t) \tag{3.18}$$

The results will be analysed by a spectogram, which will show if there is a clear frequency and where in the video it is high or low.

To create a spectogram the signal is first transformed using a Discrete Fourier Transform (Nunez-Iglesias et al., 2017). This is a technique used to convert temporal or spatial data into *frequency*

domain data. The fast Fourier transform (FFT) is a fast algorithm for computing the DFT. The Fourier transform takes a signal in the time domain and converts it into a spectrum, a set of frequencies and corresponding values. This spectrum does not contain any information about when the frequencies are occurring in time. For the oscillations in the video it is interesting when the frequencies are occuring, so this is not enough. The *short time Fourier transform* is applied, which means that the signal is split into overlapping slices and the Fourier transform is applied to each slice. This is shown in Figure 3.2. In this way it can be determinded at which moment which frequency can be detected.

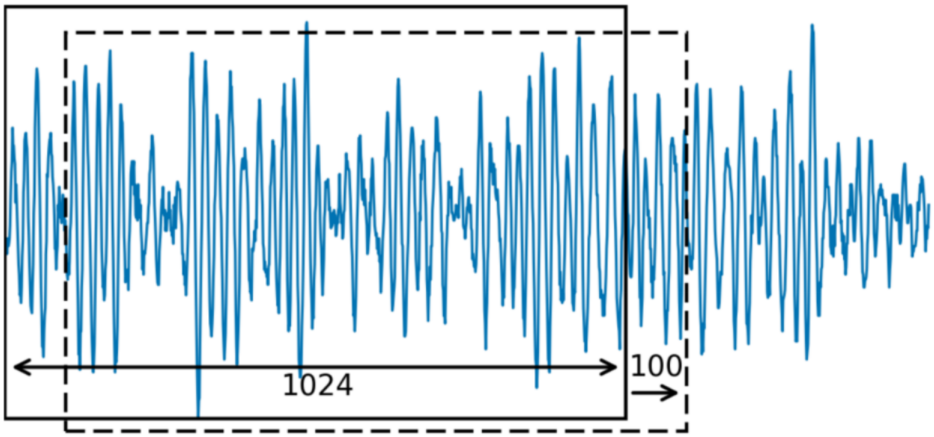


Figure 3.2: Illustration of the slices used in the FFT

3.4 Detection of rocks

The presence and movement of rocks are both approximated. When the blast furnace is working, rocks should always be a little bit visible in the frame. When rocks stop moving, the blast furnace could experience problems like blockage.

Presence of rocks

To analyse the presence of rocks in the camera vision it is necessary to look at the distribution of the pixel values in the background area. The pixels corresponding to the visible colder surfaces of the rocks are darker than the pixels between the rocks or on the surfaces of hotter underlying rocks in the same frame. If there are big differences between the pixel values, this could be an indication of more rocks.

To calculate whether there are big differences in pixel value, we subtract every pixel value by the spatial mean of the pixel values and we sum up the absolute value of the results. It should also be taken into account that the background light is not always the same. In some frames, the background is very bright, indicating a higher temperature. In these frames the difference between the stone pixels and the non-stone pixels is lower than in the darker frames (with a lower temperature). To correct for this difference, we divide the result by the spatial mean of the pixel values. The formula for the presence of rocks at time t is therefore as follows:

$$\text{Presence of rocks } (t) \sim \sum_{(i,j)\in A_{background}} \frac{|\Phi_{i,j}(t) - \overline{\Phi}(t)|}{\overline{\Phi}(t)} \tag{3.19}$$

Movement of rocks

To calculate the movement of rocks, the change in the presence of rocks and thus the change in the outcome of formula (3.19) should be determined. To do this we look at the change of the difference between the pixel value and the mean pixel values in the background in each pixel individually, and take the norm of this change. The formula for this is as follows:

$$\text{Movement of rocks at time } t \sim \left\| \frac{\Phi_{i,j}(t) - \overline{\Phi}(t)}{\overline{\Phi}(t)} - \frac{\Phi_{i,j}(t-\Delta t) - \overline{\Phi}(t-\Delta t)}{\overline{\Phi}(t-\Delta t)} \right\|_2 \text{ for } (i,j) \in A_{background} \tag{3.20}$$

3.5 Rescaling

In the formulas in this chapter information from the frames is transformed to information about physical aspects of the blast furnace. In these formulas the results are not the exact value of these physical quantities but a closely related quantity that can be used to compare different frames with each other. Therefore, the results of these calculations should be rescaled to values that are easier to understand. To do this, the results should be considered first. It should also be determined at which point the situation is critical - for example when the coal injection stops - so that a feedback program can be implemented and/or an alarm could go off.

Chapter 4

Results

In the following sections the methods proposed above will be applied to the sample video provided by the Danieli Corus company, starting with the extraction and analysis of physical quantities and associated processes, followed by an overview of the program performance.

4.1 Analysis of the sample video

In the following section the results of the calculations of physical quantities will be presented. To compare these results with the video, there are nine moments in the video displayed in Figure 4.1. The screenshots are displayed in gray-scale, since the calculations are also executed on the gray-scaled images.

The number above a screenshot indicates the number of the frame. The test video is split into 10000 frames, with 24 frames per second. In these frames it can be seen that in the beginning of the video pulverized coal is injected, from frame 400 this is not visible anymore and the temperature rises. From frame 6000, there are more rocks visible that are moving less. It seems that they are blocking the entry to the blast furnace.

In the presentation of estimates for the physical quantities the moving average is also shown (sometimes abbreviated to mva). This is the average of a number of previous measurements. This is done to make it easier to see trends in the results. The number of previous frames used in the calculation of the moving average is set to 50, which corresponds to 2.08 seconds.

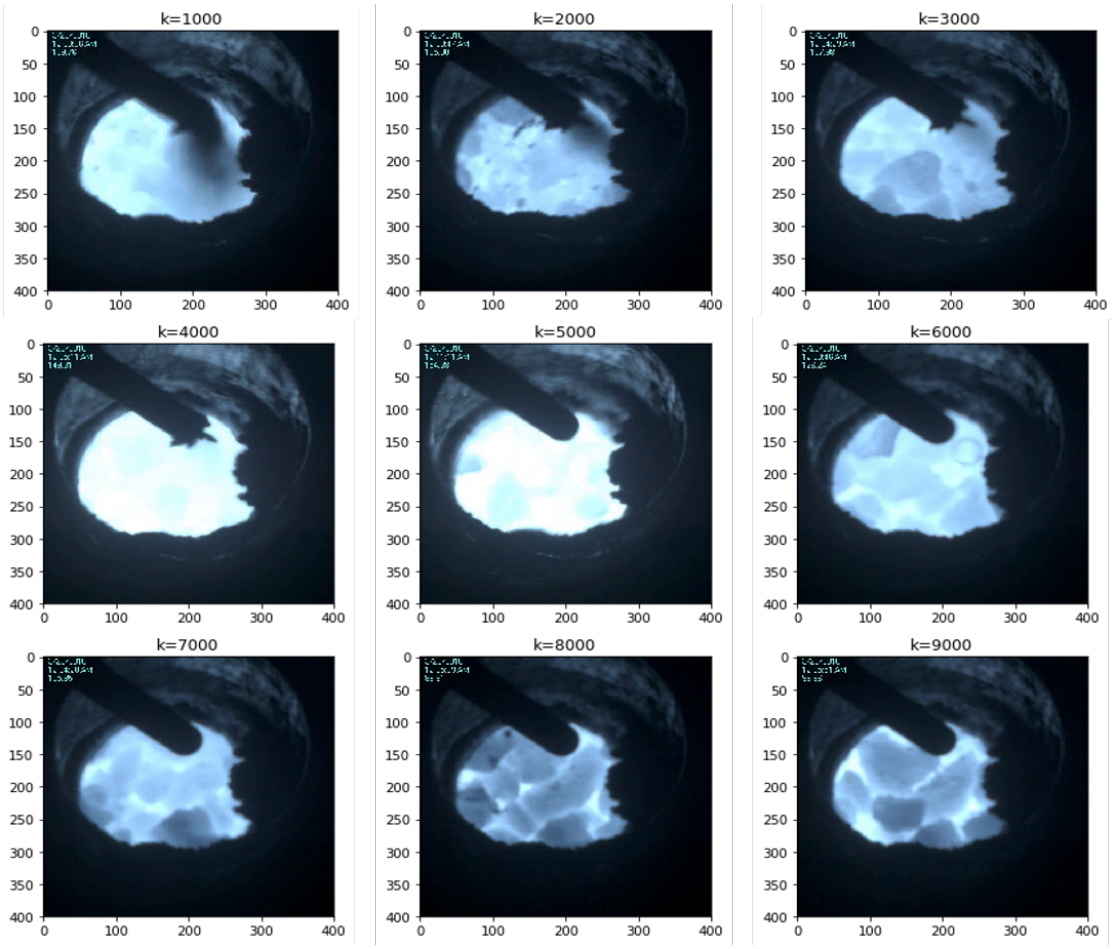


Figure 4.1: Nine screenshots from test video

4.1.1 Pulverized coal mass variation

For the amount of pulverized coal the results are shown in Figure 4.2. It can be seen that the most coal is injected at the beginning of the video and this becomes less untill frame 3000. Then the injection rate stays low, but is still fluctuating a lot. In Figure 4.1 we see that after frame 3000 the coal injection stops for the rest of the video. The fluctuating graph could be the result of differences in rocks in the background and the coal area. An improvement to this could be to look at a smaller area for the coal injection, as described by Birk et al. (2002).

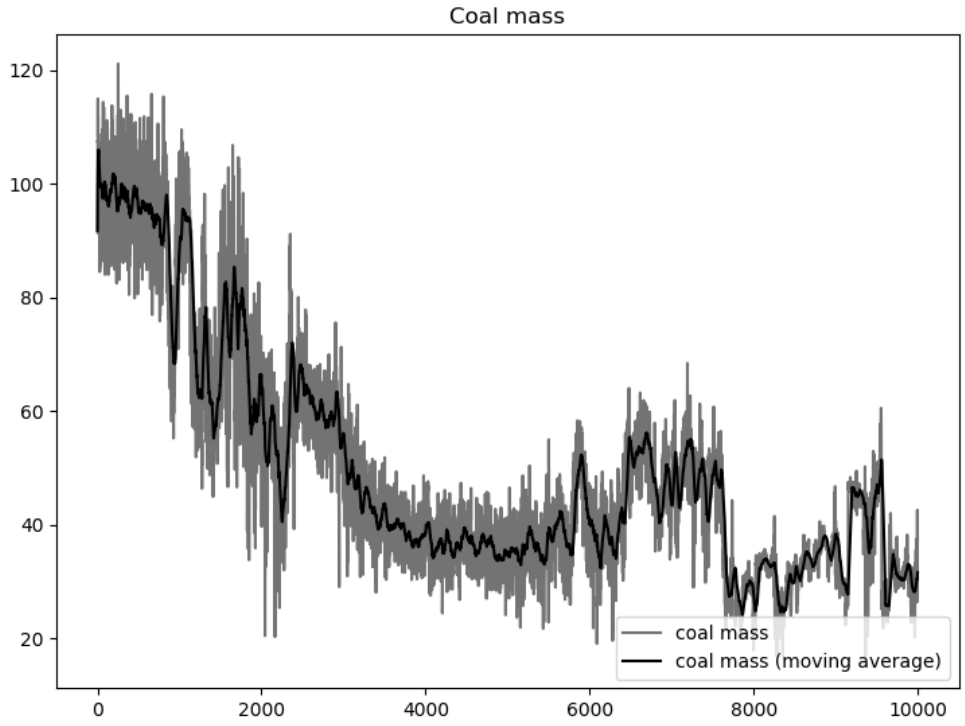


Figure 4.2: Amount of pulverized coal

4.1.2 Temperature variation

Both the temperature calculation based on the background light as well as based on the heat-induced oscillations are presented.

Background light intensity

In Figure 4.3 the calculated light intensity in the background is shown in a graph. It is clear that there are two moments in the video where the light intensity is relatively high. The first moment is from the beginning untill ± frame 1500. The second moment is from ± frame 3500 untill ± frame 5000. Is we look at the screenshots in Figure 4.1 we see that this could be correct, since frame 1000, 4000 and 5000 show the brightest background light. Frame 8000 shows the darkest background - also due to the presence of blocking rocks - and in the graph this is also visible, since the temperature based on light intensity drops here as well.

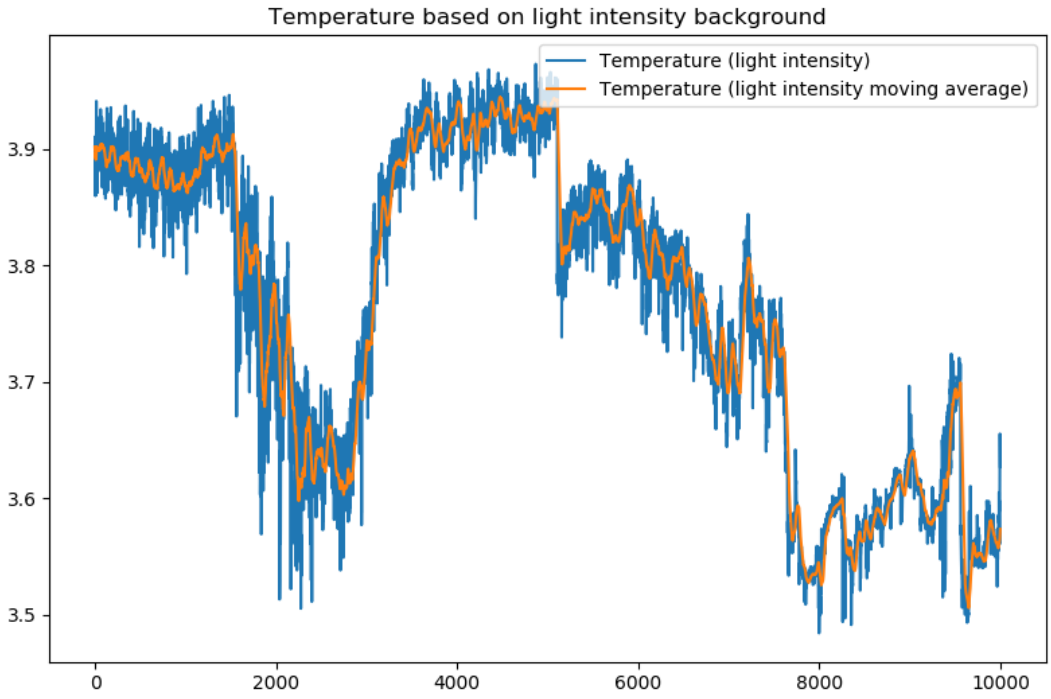
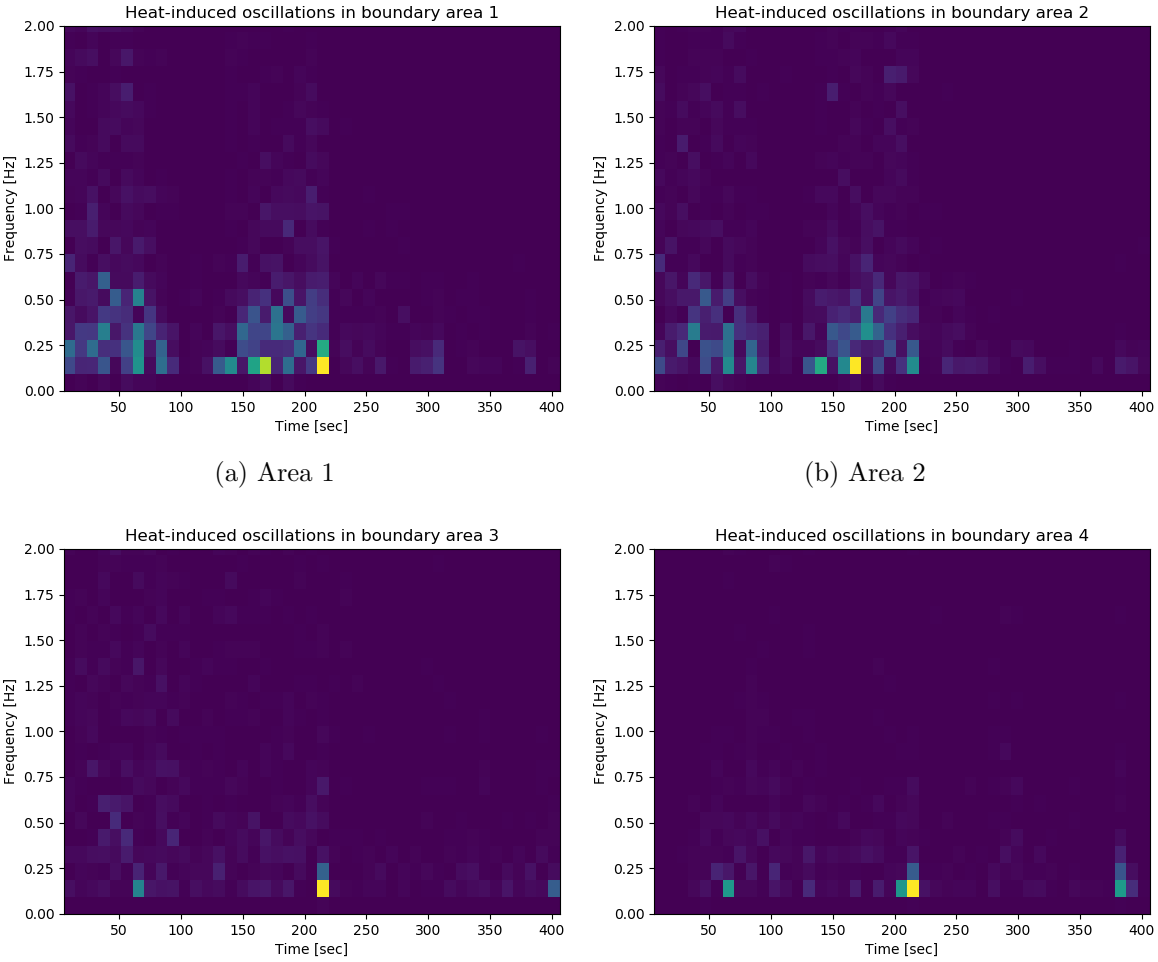


Figure 4.3: Temperature, based on light intensity

Heat-induced oscillations

To analyse the oscillations in the boundary four different spectograms are generated, based on the four parts of the boundary. It should be taken into account that each second 24 frames were analysed, so the time axis goes from 0 second to 416 seconds. In Figure 4.4 it can be seen that in the Areas 1 and 2 the spectrum of oscillations is more pronounced than in the Areas 3 and 4. If we focus on the Areas 1 and 2 we see the highest frequencies at ± 60 seconds and between ± 150 and 220 seconds. This corresponds to frame 1440 and the frames between 3600 and 5280. This would indicate high temperatures at these moments, which gives a very similar result as the temperature based on light intensity.



(a) Area 1

(b) Area 2

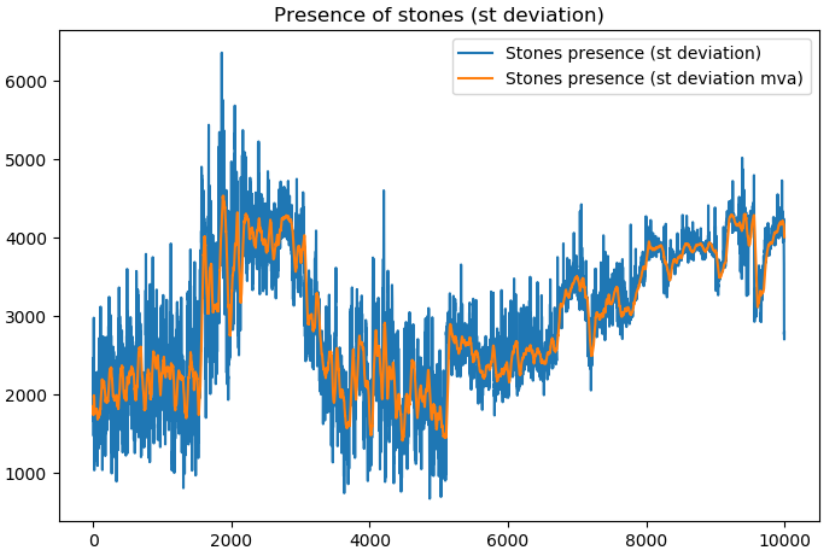
(c) Area 3

(d) Area 4

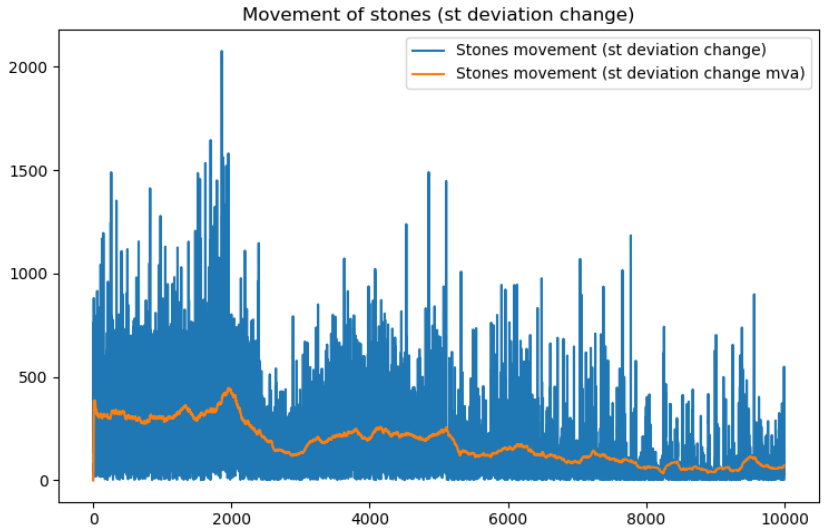
Figure 4.4: Temperature, based on heat-induced oscillations in boundary areas

4.1.3 Detection of rocks

The presence and movement of rocks, as calculated by the method described in Section 3.4, is shown in Figure 4.5. For the presence Figure 4.5a shows an increase in the beginning of the video untill frame ± 2000. Then it decreases again and starting frame 5000 it rises untill the end of the video. This pattern can be confirmed by the visual inspection of the video, see Figure 4.1. The movement of rocks is more or less constant untill frame 2000, after which it it slows down untill the end of the video. Although it is hard to directly compare this with the video, in the end the rocks are clearly moving less than in the beginning, which also follows from the analysis (see the moving average in Figure 4.5b.



(a) Presence of rocks



(b) Movement of rocks

Figure 4.5: Stones

Figure 4.6: Relative results of temperature, coal injection and movement of rocks

4.2 Cross examination of results

In Figure 4.6 the temperature, coal injection and movement of rocks are shown in one figure. The quantities are normalized for the sake of comparison. According to Danieli Corus the following two phenomena are frequently observed in their furnaces:

- After the injection of coal powder stops, the temperature rises.
- Faster rock motion is associated with higher temperatures.

Both of these phenomena are present in the provided sample video and are easy to detect in the estimated quantities extracted by our algorithms, see Figure 4.6.

4.3 Program speed

The code as seen in the Appendix takes 834.7 seconds to analyse 10000 frames. This equals 0.0835 seconds per frame. This means that in real time the program could evaluate $\frac{1}{0.0835} = 11.98$ frames per second. It should be taken into account that this also depends on the computing device, and depending on the needs of the results a faster computing system could be considered. In Figure 4.7 a screenshot from the running program is showed.

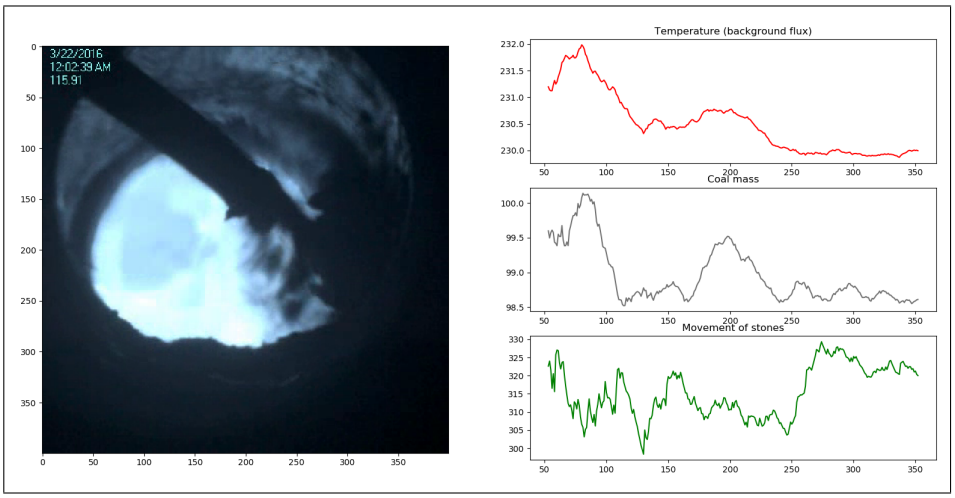


Figure 4.7: Example of the running program

Chapter 5

Conclusion, discussion and further research

Application of the methods and algorithms proposed in this Thesis to the sample tuyere camera video provided by the Danieli Corus company shows results that could be expected from the working of the blast furnace. To really test how precise the results match with the real conditions in the furnace and determine the appropriate calibration methods further research is required. For example the temperature should be measured and compared to the results of the light intensity and the oscillations. In addition to this, more test videos should be analysed to see if the results are consistent.

It would be better to analyse all tuyere cameras from one blast furnace together. In this way, when a blockage of rocks occurs, it can be estimated how big it is and in which areas the problems will be the biggest. Also the temperature distribution in the furnace can be determined in this way.

The last proposal for an improvement of the analyses is to reduce the size of the regions of interest. In particular, the coal area. This would decrease the chance that the calculated amount of coal is influenced by the amount of rocks that are accidentally taken into account.

Bibliography

Birk, W., Marklund, O., and Medvedev, A. (2002). Video monitoring of pulverized coal injection in the blast furnace. *IEEE Transactions on Industry Applications*, 38(2):571–576.

Danieli-Corus (2019). Pulverized coal injection systems. `https://www.danieli-corus.com/ironmaking/pulverized-coal-injection`. [Online; accessed 2019-july-05].

Halir, R. and Flusser, J. (1998). Numerically stable direct least squares fitting of ellipses. *Proceedings of the 6th International Conference in Central Europe on Computer Graphics and Visualization, (WSCG, Plzen, Czech Republic)*, page 125132.

Nunez-Iglesias, J., van der Walt, S., and Dashnow, H. (2017). *Elegant Scipy*. O'Reilly Media, Inc.

Paul-Wurth (2014). Market breakthrough for tuyere phenomena detection system. `http://www.paulwurth.com/News-Media/News-and-Archives/Market-Breakthrough-for-Tuyere-Phenomena-Detection-System`. [Online; accessed 2019-july-05].

Simoes, J.-P., Goedert, P., Reuter, Y., and Dorpe, P. V. (2012). Tuyere phenomena detection system. *Technical contribution to the 6th International Congress on the Science and Technology of Ironmaking ICSTI, 42nd International Meeting on Ironmaking and 13th International Symposium on Iron Ore, October 14th to 18th, 2012, Rio de Janeiro, RJ, Brazil*.

Wang, Y., Hu, Y., Yang, G., Qin, T., and Yuan, W. (2015). Feature extraction algorithms based on coal injection image of video in the blast furnace. *Procedia Engineering*, 102:265–272.

Ward, N., Klaas, M., DAlessio, J., and Badgley, P. (2017). Blast furnace process monitoring and control through the use of tuyere camera technology. *AISTech 2017 Proceedings*.

Appendix - code

Initialization

```
#import the video and initialization
cap = cv2.VideoCapture('BEPvideo.WMV')
frames = {}
i=0
n = 1000 #number of points on ellipse
partition = 165 # x-axis for seperating background and coal powder area
mva = 50 # how many frames taken into account in the moving average
f = 300 #how many timesteps to show in graph

# lists and dictionaries for saving results
framenumber = []
background_list = []
background_list_mva = []
coalarea_list = []
coalarea_list_mva = []
boundary_list = []
difference = []
difference_mva = []
background_stdeviation = []
background_stdeviation_mva = []
background_stdeviation_change = []
background_stdeviation_change_mva = [0]
A_boundary1_values = []
A_boundary2_values = []
A_boundary3_values = []
A_boundary4_values = []

#create list with pixels
pixels = []
for i in range(400):
    for j in range(400):
        pixels.append((i,j))

#read first frame and choose points on the ellipse
ret, frame = cap.read()
image1 = frame
cv2.imwrite('frame'+str(i)+'.jpg',frame)
```

```
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

init_coord, init_coord_x, init_coord_y = choosepoints(frame)
init_coord_2, init_coord_x_2, init_coord_y_2 = choosepoints(frame)
pipe, pipe_x, pipe_y = choosepoints(frame)

pipe_small = [(45,37),(200,152),(220,131),(67,21)]
pipe_small.append(pipe_small[0])
pipe_big = [(15,37),(200,182),(250,131),(67,-9)]
pipe_big.append(pipe_big[0])

#create ellipses through chosen points
ellipse1, ellipse1_small, ellipse1_big = ellipsefind(init_coord_x, init_coord_y)
ratio_width_height = ellipse1.width / ellipse1.height
ellipse2 = Ellipse(xy=[175, 225], width=260, height=260/ratio_width_height,
    angle=ellipse1.angle, edgecolor='r', fc='None', lw=2, label='Fit',
    zorder = 2)
ellipse2_small = Ellipse(xy=[175, 225], width=260-20,
    height=260/ratio_width_height-20, angle=ellipse1.angle, edgecolor='r',
    fc='None', lw=2, label='Fit', zorder = 2)
ellipse2_big = Ellipse(xy=[175, 225], width=260+20,
    height=260/ratio_width_height+20, angle=ellipse1.angle, edgecolor='r',
    fc='None', lw=2, label='Fit', zorder = 2)

#create pipe polygon through the points
polygon = Polygon(pipe, closed = True, edgecolor='g', fc='None', lw = 2)
polygon_small = Polygon(pipe_small, closed = True, edgecolor='g', fc='None',lw=2)
polygon_big = Polygon(pipe_big, closed = True, edgecolor='g', fc='None',lw=2)

#create line to seperate background from area with coal
line = Polygon ([(partition,50),(partition,350)], closed = False,
    edgecolor = 'w', lw=2)

#walk through points and save the ones that are in the right area
#(in both ellipses and not in pipe)
pixelsinfirstellipse = GetPixelsInEllipse(pixels,ellipse1)
pixelsinbothellipses = GetPixelsInEllipse(pixelsinfirstellipse,ellipse2)
A_background_coalarea = GetPixelsOutofPipe(pixelsinbothellipses,pipe)
A_background, A_coalarea = splitpixels(A_background_coalarea, partition)

#find boundary neighbourhood
    #all pixels in bigger area (bigger ellipses but smaller pipe)
pixelsinfirstellipse_extended = GetPixelsInEllipse(pixels,ellipse1_big)
pixelsinbothellipses_extended = GetPixelsInEllipse(pixelsinfirstellipse_big,
    ellipse2_big)
A_background_coalarea_extended = GetPixelsOutofPipe(pixelsinbothellipses_big,
    pipe_small)
    #find pixels in smaller area
pixelsinfirstellipse_reduced = GetPixelsInEllipse(pixels,ellipse1_small)
```

```
pixelsinbothellipses_reduced =
    GetPixelsInEllipse(pixelsinfirstellipse_reduced,ellipse2_small)
A_background_coalarea_reduced =
    GetPixelsOutofPipe(pixelsinbothellipses_reduced,pipe_big)
    #find pixels in boundary (in extended area but outside reduced area)
for (i,j) in A_background_coalarea_reduced:
    A_background_coalarea_extended.remove((i,j))
A_boundary = A_background_coalarea_reduced

#split boundary in 4 parts
centerx = 170
centery = 240

A_boundary1 = [] #lower left
A_boundary2 = [] #lower right
A_boundary3 = [] #upper left
A_boundary4 = [] #upper right
for (i,j) in A_boundary:
    if (i<=centerx and j>centery):
        A_boundary1.append((i,j))
    elif (i>centerx and j>centery):
        A_boundary2.append((i,j))
    elif (i<=centerx and j<=centery):
        A_boundary3.append((i,j))
    elif (i>centerx and j<=centery):
        A_boundary4.append((i,j))

#start the plot
plt.ion()
fig = plt.figure('output')
plt.clf()
ax1 = fig.add_subplot(1,2,1); plt.imshow(frame);
ax2 = fig.add_subplot(3,2,2)
ax3 = fig.add_subplot(3,2,4)
ax4 = fig.add_subplot(3,2,6)
```

Main

```
#read all frames
start_time = time.time()
k=0
while(cap.isOpened() and k<10000):
    k=k+1
    framenumber.append(k)
    ret, frame = cap.read()
    frames[k] = frame
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    #calculate temperature (background light intensity)
    background = som(gray, A_background)
    background_mean = background / len(A_background)
    background_list.append(sqrt(sqrt(background_mean)))
    background_list_mva.append(moving_average(background_list,k,mva))

    #calculate light intensity coal area
    coalarea = som(gray, A_coalarea)
    coalarea_mean = coalarea / len(A_coalarea)
    coalarea_list.append(coalarea_mean)
    coalarea_list_mva.append(moving_average(coalarea_list,k,mva))

    #calculate coal (difference between foreground and background)
    difference.append(background_mean-coalarea_mean)
    difference_mva.append(background_list_mva[k-1]-coalarea_list_mva[k-1])

    #calculate boundary in four different regions
    A_boundary1_value = som(gray, A_boundary1)
    A_boundary1_values.append(A_boundary1_value)
    A_boundary2_value = som(gray, A_boundary2)
    A_boundary2_values.append(A_boundary2_value)
    A_boundary3_value = som(gray, A_boundary3)
    A_boundary3_values.append(A_boundary3_value)
    A_boundary4_value = som(gray, A_boundary4)
    A_boundary4_values.append(A_boundary4_value)

    #calculate stones
    liststdeviation = stdeviation(gray,A_background)
    background_stdeviation.append(sum(np.abs(liststdeviation)))
    background_stdeviation_mva.append(moving_average(background_stdeviation,k,mva))
    if k==2:
        background_stdeviation_change.append
            (np.linalg.norm(background_stdeviation[k-1]-
            background_stdeviation[k-2]))
        background_stdeviation_change.append
            (np.linalg.norm(background_stdeviation[k-1]-
            background_stdeviation[k-2]))
```

```
            background_stdeviation_change_mva.append
                (moving_average(background_stdeviation_change,k,mva))

        elif k>2:
            background_stdeviation_change.append
                (np.linalg.norm(background_stdeviation[k-1]-
                background_stdeviation[k-2]))
            background_stdeviation_change_mva.append
                (moving_average(background_stdeviation_change,k,mva))

        #produce output
        ax1.cla()
        ax2.cla()
        ax3.cla()
        ax4.cla()
        ax1.imshow(frames[k])
        ax2.plot(lastframes(framenumber,f),
            lastframes(background_list_mva,f), color = 'r');
            ax2.set_title('Temperature (background light)')
        ax3.plot(lastframes(framenumber,f),
            lastframes(difference_mva,f),color = '#737373');
            ax3.set_title('Coal mass')
        ax4.plot(lastframes(framenumber,f),
            lastframes(background_stdeviation_change_mva,f),color = 'g');
            ax4.set_title('Movement of stones')

        plt.pause(0.001)
        if cv2.waitKey(5) & 0xFF == ord('q'):
            break

cap.release()
```

Definitions

```
def som(gray, pixelcoordinates):
    som = 0
    for (i,j) in pixelcoordinates:
        som = som + gray[j][i]
    return som

def lastframes(lst, number):
    q = min(len(lst), number)
    newlist = lst[len(lst)-q:len(lst)]
    return newlist

def stdeviation(gray, pixelcoordinates):
    lst = []
    liststdeviation = []
    for (i,j) in pixelcoordinates:
        lst.append(gray[j][i])
    mean = sum(lst)/len(lst)
    for i in lst:
        liststdeviation.append((i - mean) / mean)
    return liststdeviation

def stdeviation_presence_and_change(gray1, gray2, pixelcoordinates):
    lst1 = []
    lst2 = []
    for (i,j) in pixelcoordinates:
        lst1.append(gray1[j][i])
        lst2.append(gray2[j][i])
    mean1 = sum(lst1)/len(lst1)
    mean2 = sum(lst2)/len(lst2)
    stdeviation = 0
    stdeviationchange = 0
    for (i,j) in pixelcoordinates:
        stdeviation += ((gray1[j][i] - mean1) / mean1)**2
    for (i,j) in pixelcoordinates:
        stdeviationchange += (((gray1[j][i] - mean1) / mean1) -
            ((gray2[j][i] - mean2) / mean2))**2

    stdeviation = sqrt(stdeviation)
    stdeviationchange = sqrt(stdeviationchange)
    return stdeviation, stdeviationchange

def choosepoints(image):
    root = Tk()
    global my_points; my_points=[]
    global my_points_x; my_points_x=[]
    global my_points_y; my_points_y=[]
```

```
    #setting up a tkinter canvas with scrollbars
    frame = Frame(root, bd=2, relief=SUNKEN)
    frame.grid_rowconfigure(0, weight=1)
    frame.grid_columnconfigure(0, weight=1)
    xscroll = Scrollbar(frame, orient=HORIZONTAL)
    xscroll.grid(row=1, column=0, sticky=E+W)
    yscroll = Scrollbar(frame)
    yscroll.grid(row=0, column=1, sticky=N+S)
    canvas = Canvas(frame, bd=0, xscrollcommand=xscroll.set,
        yscrollcommand=yscroll.set)
    canvas.grid(row=0, column=0, sticky=N+S+E+W)
    canvas.create_line(15, 25, 200, 25)
    xscroll.config(command=canvas.xview)
    yscroll.config(command=canvas.yview)
    frame.pack(fill=BOTH,expand=1)

    #adding the image
    img = ImageTk.PhotoImage(Image.open('frame0.jpg'))
    canvas.create_image(0,0,image=img,anchor="nw")
    canvas.config(scrollregion=canvas.bbox(ALL))

    #function to be called when mouse is clicked
def printcoords(event):
        #outputting x and y coords to console
        print (event.x,event.y)
        global my_points; my_points += [(event.x,event.y)]
        global my_points_x; my_points_x += [event.x]
        global my_points_y; my_points_y += [event.y]
        canvas.create_oval(event.x-4,event.y-4,event.x+4,event.y+4,
            outline="#f11",fill="#1f1", width=2)
    #mouseclick event
    canvas.bind("<Button 1>",printcoords)
    root.mainloop()
    return my_points, my_points_x, my_points_y

def moving_average(listt, index, n):
    q = min(n, index)
    r = 0
    for i in range(q):
        r = r + listt[index-1-i]
    r = r/q
    return r


def GetPixelsInEllipse(pixels, ellipse):
    #turn ellipse into polygon
    pixels1 = pixels.copy()
    polygon = []
    polygon1 = points_on_ellipse(ellipse,n)
```

```
    for i in range(n):
        polygon.append([polygon1[0][i],polygon1[1][i]])
    #find if points are in ellipse and add them to result if they are
    path = mpltPath.Path(polygon)
    grid = path.contains_points(pixels1)
    pointsinpolygon = []
    for i in range(len(grid)):
        if (grid[i] == True):
            pointsinpolygon.append(pixels1[i])
    return pointsinpolygon

def GetPixelsOutofPipe(pixels, pipe):
    #turn ellipse into polygon
    pixels1 = pixels.copy()
    #find if points are outside pipe and add them to result if they are
    path = mpltPath.Path(pipe)
    grid = path.contains_points(pixels1)
    pointsoutpolygon = pixels1
    for i in range(len(grid)):
        if (grid[i] == True):
            pointsoutpolygon.remove(pixels[i])
    return pointsoutpolygon

def splitpixels(pixels,xpartition):
    left = []
    right = []
    for i in pixels:
        if i[0] < xpartition:
            left.append(i)
        else:
            right.append(i)
    return left, right

def ellipsefind (x,y):
    data = [np.array(x),np.array(y)]
    lsqe = el.LSqEllipse()
    lsqe.fit(data)
    center, width, height, phi = lsqe.parameters()
    ellipse = Ellipse(xy=center, width=2*width, height=2*height,
        angle=np.rad2deg(phi), edgecolor='b', fc='None', lw=2, label='Fit')
    ellipse_small = Ellipse(xy=center, width=2*width-20, height=2*height-20,
        angle=np.rad2deg(phi), edgecolor='b', fc='None', lw=2, label='Fit')
    ellipse_big = Ellipse(xy=center, width=2*width+20, height=2*height+20,
        angle=np.rad2deg(phi), edgecolor='b', fc='None', lw=2, label='Fit')
    return ellipse, ellipse_small, ellipse_big

#find n points on ellipse
def points_on_ellipse(ellipse,n):
    xarray_ellipse_new = []
```

```
yarray_ellipse_new = []

rad = 0

for i in range(n+1):
    rad = rad + 2*math.pi/n
    new_xpos = ellipse.width/2*math.cos(rad)*math.cos
        (np.deg2rad(ellipse.angle))-ellipse.height/2*math.sin(rad)*
        math.sin(np.deg2rad(ellipse.angle))+ellipse.center[0]
    new_ypos = ellipse.width/2*math.cos(rad)*math.sin
        (np.deg2rad(ellipse.angle))+ellipse.height/2*math.sin(rad)*
        math.cos(np.deg2rad(ellipse.angle))+ellipse.center[1]
    xarray_ellipse_new.append(new_xpos)
    yarray_ellipse_new.append(new_ypos)

for i in range(len(xarray_ellipse_new)):
    if xarray_ellipse_new[i] < 0:
        xarray_ellipse_new[i]=0
    elif xarray_ellipse_new[i]>399:
        xarray_ellipse_new[i]=399

for i in range(len(yarray_ellipse_new)):
    if yarray_ellipse_new[i] < 0:
        yarray_ellipse_new[i]=0
    elif yarray_ellipse_new[i]>399:
        yarray_ellipse_new[i]=399

return xarray_ellipse_new , yarray_ellipse_new
```