Team Number:	2208185
Problem Chosen:	A

2022 APMCM summary sheet

Mould protection slag plays the role of heat preservation and oxidation prevention in continuous casting process. Therefore, the distribution of mould flux has been the focus of research in the metallurgical industry. In this paper, graphical processing methods and regression analysis are used to discuss the melting crystallization process of mould protection slag, which provides useful guidance for metallurgical industry to improve technology.

For Task 1, we design a high-precision text recognition model for the manually collected label graph of smelting key nodes to improve the efficiency of extracting image information. Firstly, the image is preprocessed, and then the **DenseNet model**is trained again based on **the CNOCR recognition** library. Finally, the accuracy of the model for the extraction of two groups of thermocouple temperature data is improved to **more than 98.8%**. In addition, the reason for the large measurement deviation of the original image is inferred by mean analysis of the extracted temperature data.

For task 2, in order to comprehensively analyze the changing characteristics of image features over time, we extract the feature vectors of the images in Appendix 1 from five aspects: **gray feature, color feature,IBP feature, frequency feature and wavelet**. Then the within-group feature significance and between-group feature significance were analyzed using principal component analysis and box plots. The experimental results show that all the five features show strong significance. Based on this, the general Gaussian regression was used to fit the above mathematical model, and the melting and crystallization process of protection slag was analyzed and studied.

For task 3, in order to further analyze the relationship between the temporal characteristics and the melting crystallization thermodynamics of the protection slag, a **Gaussian process regression model** for the temperature and time of the #1 and #2 thermocouple was established based on the feature vectors of the existing keyframe images. Using a regression model as a prediction operator, we performed numerical simulations and fortunately found a polynomial relationship between temperature and melting period and die. Finally, polynomial regression was used to obtain the functional relationship between temperature and time changes and the melting and crystallization process of protective slag.

Finally, by searching the literature, the above models are reasonably evaluated and suggestions for the development of follow-up research are put forward.

Keywords: CNOCR DenseNet Image processing PCA Gaussian process regression

Contents

1. Introduction **1**
1.1 Background 1
1.2 Restatement of the Problem 1
1.3 Our Work 2
2. Problem analysis **2**
3. Assumption **3**
4. Symbol Description **3**
5. Model of Task 1 . . . . . **3**
5.1 Dataset pre-processing 3
5.2 Model Building——CNOCR 4
5.3 Model solving process 5
5.4 Handbook of CNOCR Model 6
6. Model of Task 2 . . . . . **8**
6.1 Image Rough analysis 8
6.2 Model Building——5-domains feature extractor 8
6.3 Model solving process 14
7. Model of Task 3 . . . . . **15**
7.1 Rough analysis of relationship between temperature and time 15
7.2 Model Building——Gaussian process Regression 16
7.3 Model solving process 17
8. Strengths and Weakness. **19**
9. Future Work **20**
10. Conclusions **21**
10.1 Conclusions of the problem 21
10.2 Methods used in our models 22
10.3 Applications of our models 22
11. Appendix **23**
Appendix **23**

I. Introduction

1.1 Background

Mould flux plays the role of thermal insulation and preventing oxidation in the continuous casting process. The function of mould flux is mainly determined by the melting rate and crystallization rate of the flux under the temperature control curve. Therefore, the distribution of mould flux has always been a key research direction in metallurgical industry.

During smelting, the mould flux is continuously added to the liquid steel on the top of the container and accumulates in the form of powder on the surface of the liquid steel, which can prevent the crust of the liquid steel layer due to too fast cooling.

After crystallization, it will eventually form three layers of slag film structure: glass layer, crystal layer and liquid slag layer. When the mould flux is completely fused, a layer of liquid slag will be formed on the surface of the molten steel. With the decrease of the longitudinal temperature of the mould liquid slag, the slag film attached to the wall of the copper plating mold was quenched and solidified to form a glass layer. At the end of smelting, the slag film appeared regional crystallization and formed a crystalline layer when the mould was forced to cool.

As the phase transition of the mould flux is not easy to observe,with SHTT melting crystallization temperature tester, the researcher recorded the 562 crystallization behavior images of the mould flux from the 110^{th} second to 674^{th} second , see Attachment 1 and Figure 1-6, and the artificial identification information was recorded in the upper left corner of the key node image. In order to improve the recognition efficiency and facilitate scientific research, it is urgent to realize automatic feature extraction and model abstraction of images.

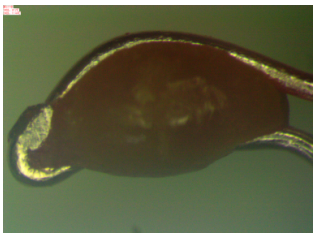

Figure 1 The sample

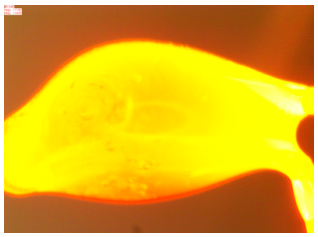

Figure 2 Begins to melt at 110s

Figure 3 Completely melted at 141s

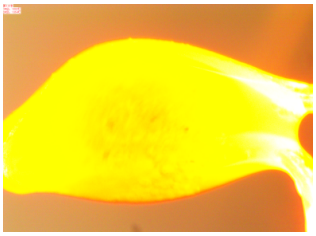

Figure 4 Sees steels melting at 149s

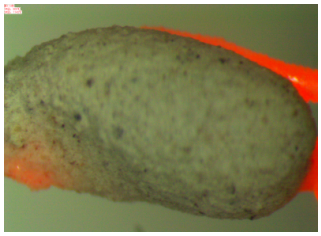

Figure 5 Crystallize begins at 150s

Figure 6 Crystallize ends at 671s

1.2 Restatement of the Problem

Task 1: Please use Image Recognition to establish a mathematical model to realize the automatic extraction and data filling of # 1 & # 2 thermocouple temperature of key node image. By the way,a

technical document is expected. In addition, make a temperatture-time curve diagram , point out and explain the test source of error of 1# wire or 2 # wire results.

Task 2: According to 6 node images in Figs 1, the dynamic differences between adjacent sequence images in the process of melting and crystallization are to quantify by digital image processing . Based on Time Series Analysis on different characteristics,establish the simulation model and discuss the melting crystallization process curve.

Task 3: Please discuss relationship of function between the temperature and time evolution of the crystallization & melting process. Then, discuss the kinetics of melting & crystallization of mold fluxes with numerical simulation methods, including temperature, melting rate and crystallization rate.

1.3 Our Work

Based on the existing node image, we need to design a character recognition model to automatically collect the temperature data of the thermocouple and draw a curve to analyze the measurement error. Meanwhile, image feature extractor and time series model are also expected to be established. In addition, I should analyze the analytical relationship between temperature and time in the process and discuss kinetics of melting and crystallization.

II. Problem analysis

2. 1 Analysis of Task 1

After preliminary observation of Annex 1, in order to achieve accurate extraction of label information, it is necessary to preprocess the picture to improve the effective information density. There are many methods to realize text recognition through image processing technology, but it is necessary to select a model with high accuracy and strong and robust. After collecting the temperature data through the identification model, the program can be used to automate the filling of the form. Since the node image is acquired manually, the temperature error is inevitable due to the different measurement methods received.

2. 2 Analysis of Task 2

The node image can reflect the time series change of the sample state characteristics with the melting time. We need to extract the effective pixels of the sample in the image and establish a time series model for analysis. We consider achieving pixel-level quantization and drawing curves by computer simulation, according to which dynamic changes can be visually reflected.

2. 3 Analysis of Task 3

There are different temperatures in different stages of ore melting and crystallization. Sufficient samples are given in the attachment. Based on the temperature recognition model of task 1 and the pixel change curve simulated by task 2, the functional relationship can be analyzed by feature extraction and dimension reduction and compression.

Figure 7 Slice of the image after preprocessing

III. Assumption

- All the indexes of water vapor and dust in the production environment meet the production requirements and will not have adverse effects on smelting.
- The temperature measurement of the two groups of thermocouples of the equipment is accurate and there is no instrument error. However, there are measurement errors due to manual observation.
- The quality of the image acquired by the instrument meets the basic requirements of image processing and the resolution of the camera is high enough.

IV. Symbol Description

Symbol	Meaning	Unit
$T1_i$,$T2_i$	#1 temperature and #2 temperature at the i^{th}second	℃
μ,Std	First order color moment	-
η, λ, ξ	First,second and -more order color moment,	-
LBP	Linear Back Projection	-
X, Y, C	Input Set,Output Set, Kernel Function Matrix	-
P, p	Gaussian probability compensation function,Probability value	-

V. Model of Task 1

5.1 Dataset pre-processing

For purpose on the improvement of the extraction accuracy of the key information of the node image, it is necessary to standardize the detection image to improve the enrichment of the effective information of the target.

5.1.1 Image cropping

Task 1 is to extract the temperature data of thermocouple #1 and thermocouple #2 in the top-left corner of the dataset, so only the top-left region of the image contains valid data. If the original image is directly used for processing, due to the small area of the upper-left label in the image, direct recognition is not conducive to the convergence of the model and will produce large errors. If the proportion of the target image is too large, it will also increase the recognition error. Therefore, we consider the use of image processing techniques to crop the data map to the upper left 200×100 pixel range as the target label to make a new sample set,shown as following Figure7.

5.1.2 Target Limitation

Task 1 also requires to automatically fill in the table after extracting data. In the table in Attachment 2, the data are required to include: detection time, #1 temperature, #2 temperature, and the two target temperature variables are stipulated as $T1_i$,$T2_i$,denoting #1 temperature and #2 temperature at the i^{th}second .

In addition, according to the meaning of the question, the image of the attached data set is recorded from the 110^{th} second to the 674^{th} second. In order to facilitate the recording of the time, we have renamed the image file in batch and named it as ”110”, ”111”,... , ”674”, facilitating aspects of image management,operation and manual review.

5 . 2 Model Building——CNOCR

5.1.1 Introduction to OCR

Optical Character Recognition (OCR), it converts all kinds of printed text into image information through optical input methods such as scanning, and then extracts effective computer input information from the image by using character recognition technology.

Using OCR technology can realize the extraction of the target text, text recognition is divided into Chinese, English and so on. In this case, the label map includes Chinese, numbers, and symbols. Finally, we will consider using the CNOCR library in Python to train the recognition model. CNOCR is an OCR toolkit that supports the recognition of Chinese, English and numbers in Python3. It supports the recognition of vertical characters and supports secondary development. We consider training the character recognition model of this problem on the basis of CNOCR.

5.1.2 DenseNet in CNOCR

Convolutional Neural Network (CNN) is one of the most popular methods,and DenseNet is a dense connection based on CNN, where the preceding layers are connected to the following layers. Specifically, each layer accepts all the preceding layers as its additional input. In DenseNet, each layer is concatenated with all previous layers in the channel dimension (concat) and used as input to the next layer. For a L-layer network, DenseNet contains a total of $\frac{L(L+1)}{2}$ connections, and DenseNet directly concat feature maps from different layers, which can realize feature reuse and improve efficiency.

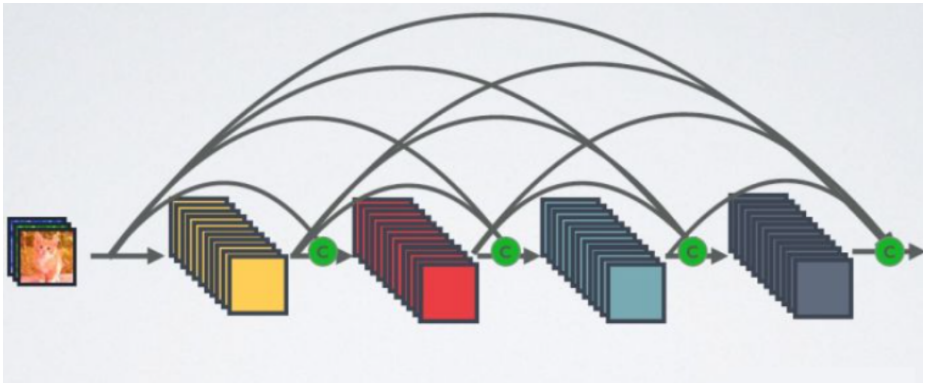

Figure 8 Dense connectivity mechanism for the DenseNet network

We adopt the CNOCR framework to implement DenseNet. CNOCR has built-in DenseNet and Densenet-Lite models. Here is a brief introduction to the training process:

Figure 9 The forward process of DenseNet

1)Input Layer

In DenseNet, all previous layers are concatenated as input:

$$X_k = H_k([x_0, x_1, \cdots, x_{k-1}]) \tag{5-2-1}$$

The H_k stands for non-liear transformation, which is a composition operation that may include a series of BN, ReLU, Pooling and Conv operations.

2)Output Feature

Filled boxes are tensors allocated to memory, while translucent boxes are Pointers. Full arrows represent computations and dashed arrows represent memory Pointers. The efficient implementation stores the output of the join layer, batch normalization layer, and reread layer in a temporary storage buffer, while the original implementation allocates new memory.

The Transition module connects two adjacent Denseblocks and reduces the feature map size by Pooling. Figure 9 shows the network structure of DenseNet, which contains a total of 4 Denseblocks connected by transitions.

Features from input Layers are integrated and standardized by concat. After multiple convolutions, the input data is reduced to one dimension. The text pixel information on the picture is propagated forward for many times to finally compress the target features and extract the text.

5.3 Model solving process

5.3.1 CNOCR processing

We use the DenseNet model in CNOCR to train the set of images obtained by preprocessing. We divide the dataset into training set, test set and validation set. Since the model is relatively mature, we will use the first 100 groups of images for training to find a more suitable parameter configuration, and then use 50 groups of data for testing. After testing, the accurate recognition rate of the model can reach **98.5%**.

After that, we used the trained network to recognize all 564 groups of images, and used pandas to automatically write the obtained text data into the table in Annex 2, as shown in Table 1 below. Due to space reasons, only the first 16 groups of text recognition results are taken:

5.3.2 Temperature change curve and anomaly analysis

Based on the temperature information obtained by the character recognition model, the temperature change curves of #1 thermocouple and #2 thermocouple with respect to time were drawn by MATLAB program, as shown in Figure 10:

It can be seen from the image that the thermocouple #1 shows a steady warming trend during the heating stage and a uniform decrease in temperature during the crystallization, but there is a

Table 1 Thermocouple temperature identification results

NO	Time	1# Temperature	2# Temperature	NO	Time	1# Temperature	2# Temperature
1	110	900	1142	9	118	935	1148
2	111	904	1146	10	119	938	1146
3	112	910	1146	11	120	942	1146
4	113	914	1144	12	121	947	1153
5	114	918	1144	13	122	951	1267
6	115	923	1146	14	123	953	1418
7	116	927	1146	15	124	959	1377
8	117	932	1146	16	125	965	1218

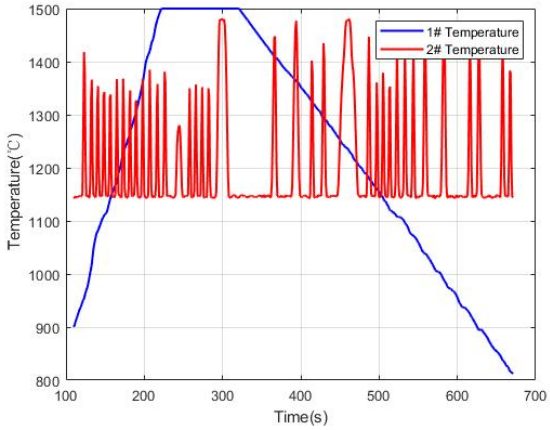

Figure 10 1# wire temperature-2# wire temperature-time diagram

constant temperature interval between the two states.

Referring to the relevant literature and the manual of SHTT experiment instrument, we analyzed and inferred the possible reasons as follows:

1) After the melting reaches the melting point continue to absorb heat will not rise.

2) SHTT experimental instrument has a thermal insulation function between the melting and crystallization stages of the sample.

There is a certain correlation between the test data of the two groups of thermocouples. According to the requirements of the topic, we calculated the average temperature of the thermocouple at each time, and drew the average temperature curve of the thermocouple #1 & #2 as shown in Figure:

5.4 Handbook of CNOCR Model

Environment:

The temperature information text recognition model of task 1 is developed based on Python3, and depends on CNOCR library for secondary training. Before using this recognition software, you should ensure that the Python environment of version 3.7 or above is installed on the computer,

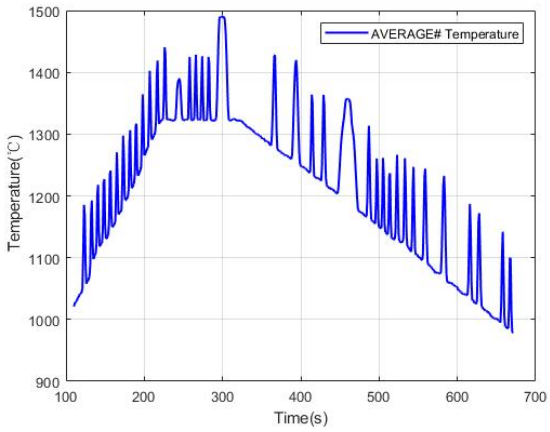

Figure 11 1#wire average temperature-2#wire average temperature-time diagram

and the third-party libraries referenced by this program are installed according to the following instructions:

Install Dependence:

```
pip install cnocr
pip install pandas
pip install re
pip install Image
```

If the installation speed is slow, you can specify the domestic installation source, such as using Douban source:

```
pip install cnocr -i https://pypi.doubanio.com/simple
```

Warning:

If you have never installed PyTorch, OpenCV python packages on your computer, you may encounter problems with the initial installation, but they are generally common problems that can be solved by Baidu /Google.

Steps:

Step.1: Place the collected original picture in the path D disk path ”D:
Attachment 1”.

Step.2: Place the table in Appendix 2 in the same directory as the python program task1.py.

Step.3: Use Powershell or Pycharm and other IDE software to start the python program, pay attention to use ”administrator mode” to ensure that the program has read and write the target file permissions.

Step.4: Finally, the temperature information will be collected and written to the Attachment 2.xlsx file in the directory

VI. Model of Task 2

6.1 Image Rough analysis

6.1.1 Selection of extracted features

The sample feature map of ore is included in the question stem. In order to comprehensively reflect the change situation of each characteristic index of ore in the smelting and crystallization process, it is necessary to increase the coverage of the number of indicators as much as possible when designing characteristic indexes, so as to improve the comprehensivity of feature extraction.

When selecting features, it can be cut from multiple dimensions, such as hue, color, pixel, etc., usually including the extraction of color features and texture features, and each sub-divided into a number of features can be used.

6.1.2 Ore smelting stage

Observing the image in the melting stage, we find that the color brightness of the image during melting is high, and the color concentration is dense, and the contrast is obvious. The corresponding image features can be extracted by pixel compression. At this time, the cosolvent is still in play, the image is bright, and the sample boundary is clear.

6.1.3 Ore crystallizing stage

In the crystallization stage of the image, we found that: with the advance of time, the sample temperature gradually decreased, and eventually the image would crystallizate into low-brightness objects, and the sample center approached to form the slag film first, and the coverage of the slag film would gradually increase the cooling rate. When the slag film layer was completely covered, the glass layer would be formed when the temperature continued to cool.

6.2 Model Building——5-domains feature extractor

6.2.1 Grayscale feature Extractors

Co-occurrence gray matrix can reflect the comprehensive information of image gray level about adjacent interval, direction and amplitude. GLCM is generally determined by statistics of specific gray levels of the image and the state between pixels, and then seven statistics based on GLCM are calculated, including mean,sid angular second moment, contrast, correlation, dissimilarity, entropy, homogeneity and energy. The gray-level features of the image are explained:

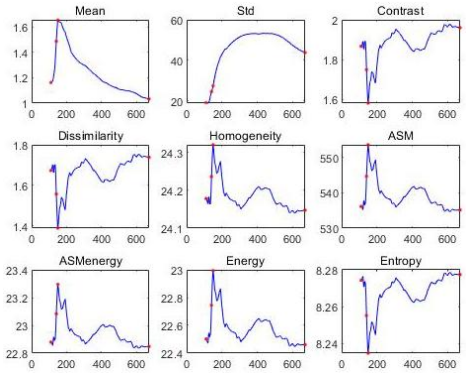

Figure 12 Curves of the 9 items of Grayscale features over time

Definition formula of 9 gray feature indexes is shown as below

$$
\begin{aligned}
Mean &= \frac{1}{i \cdot j} \sum_i \sum_j G(i,j) \\
Std &= \sum_i \sum_j (i - Mean)^2 \cdot G(i,j) \\
ASM &= \sum_i \sum_j G(i,j)^2 \\
Contrast &= \sum_i \sum_j (i-j)^2 G(i,j) \\
Correlation &= \sum_i \sum_j \frac{(i-\mu_x)(j-\mu_x)}{\sigma_x \sigma_y} \\
Dissimilarity &= \sum_i \sum_j |i-j|^k G(i,j) \\
Entropy &= -\sum_i \sum_j G(i-j) log(G(i,j)) \\
Homogeneity &= \sum_i \sum_j \frac{G(i,j)}{1+|i-j|^2} \\
Energy &= \sqrt{\sum_i \sum_j G(i,j)}
\end{aligned}
\tag{6-2-1}
$$

Where G(i, j) is the element value of the gray level co-occurrence matrix：

$$
\begin{aligned}
\mu_x &= \sum_i \sum_j i \cdot G(i,j) \\
\mu_y &= \sum_i \sum_j j \cdot G(i,j) \\
\sigma_x &= \sum_i \sum_j (i-\mu_x)^2 \cdot G(i,j) \\
\sigma_x &= \sum_i \sum_j (i-\mu_y)^2 \cdot G(i,j)
\end{aligned}
\tag{6-2-2}
$$

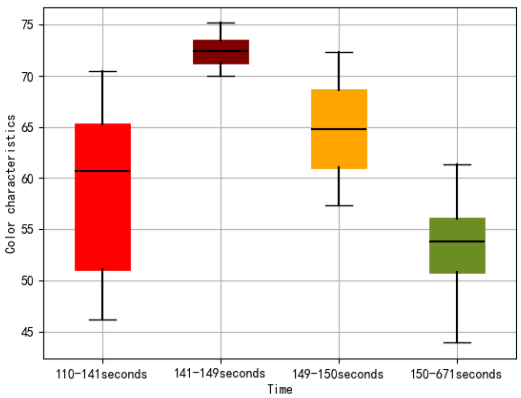

Figure 13 Grayscale features of adjacent sequence images of melting crystallization process

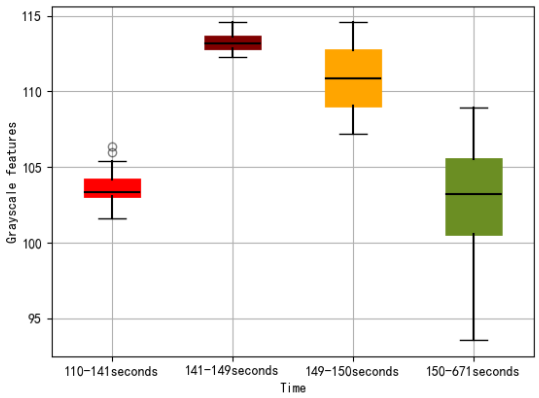

Figure 14 Color features of adjacent sequence images of melting crystallization process

6.2.2 Color Feature Extractor

Color is one of the main features of an image. Color moment is a simple and effective color feature representation method, including first order moment, second order moment and third order moment. In this paper, the first, second and third order moments of three color components R, G and B are used to represent the color distribution of the molten crystallization process image of protection slag. The lower order moments can reflect the color distribution more comprehensively, and need not quantify the space. The three color moments are defined as follows.

$$\begin{aligned} \eta_i' &= \frac{1}{N}\sum_{j=1}^{N} P_{i,j} \\ \lambda_i' &= \sqrt{\frac{1}{N}\sum_{j=1}^{N}(P_{i,j}-\eta_i)^2} \\ \xi_i' &= \sqrt[3]{\frac{1}{N}\sum_{j=1}^{N}(P_{i,j}-\eta_i)^2} \end{aligned} \tag{6-2-3}$$

The first three order color moments of the three components of the image form a 9-dimensional vector, that is, nine color features are obtained.

6.2.3 LBP Feature Extractor

LBP refers to Local Binary Pattern, which is an operator used to describe the local features of images. LBP features have significant advantages such as gray invariance and rotation invariance.

The original LBP operator is defined in a 3×3 window and the center pixel of the window is used as the threshold. Relative to the gray value of the 8 adjacent pixels, the position is marked as 1 if the value of the surrounding pixel is greater than the value of the middle pixel. Otherwise, it will be marked as 0.

The principle of the traditional LBP model is as follows:

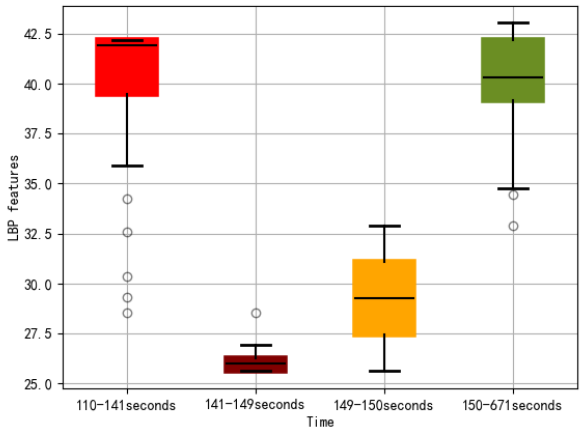

Figure 15 LBP features of adjacent sequence images of melting crystallization process

$$LBP(x_c, y_c) = \sum_{p=1}^{8} s(I(p) - I(c)) * 2^p$$
$$s(x) = \begin{cases} 1, x \geq 0 \\ 0, other \end{cases} \tag{6-2-4}$$

In order to improve the uniqueness of image features, this paper uses the improved feature model of LBP – circular LBP. Ojala et al. improved the LBP operator by extending the 3×3 neighborhood to any neighborhood and replacing the square neighborhood with a circular neighborhood to improve the circular LBP. The principle of circular LBP is basically consistent with the traditional LBP, and the sampling points P and the radius R of the sampling circle field are introduced.

$$LBP_{P,R}(x_c, y_c) = \sum_{p=1}^{P} s(I(p) - I(c)) * 2^p$$
$$\begin{cases} x_p & = x_c + R \cdot cos(\frac{2\pi p}{P}) \\ y_p & = y_c - R \cdot sin(\frac{2\pi p}{P}) \end{cases}$$
$$s(x) = \begin{cases} 1, x \geq 0 \\ 0, other \end{cases} \tag{6-2-5}$$

According to the basic principle of LBP, we extract the original LBP features, rotation invariant LBP features, equivalent pattern LBP features and rotation invariant equivalent pattern LBP features of the image region, and draw the feature time series curve, as shown in the figure:

6.2.4 Frequency Feature Extractor

We extracted the spectral frequency features of image pixels using fast Fourier transform (FFT), which is a discrete Fourier transform algorithm that is done in time of$O(n \log n)$

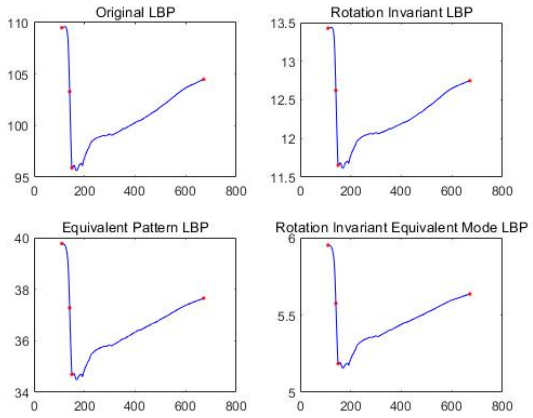

Figure 16 4 kinds of LBP feature time series curves

To expand the target expression, we apply a polynomial transformation to obtain:

$$\begin{aligned} f(x) &= (a_0 + a_2x^2 + \cdots + a_{n-2}^{n-2}) + (a_1^1 + a_3^3 + \cdots + a_{n-1}^{n-1}) \\ Then: f(x)_1 &= a_0 + a_2x^2 + \cdots + a_{n-2}x^{\frac{n}{2}-1} \\ f(x)_2 &= x(a_1 + a_3x + \cdots + a_{n-1}x^{\frac{n}{2}-1}) \end{aligned} \tag{6-2-6}$$

With $x = \omega_n^k$ and $x = \omega_n^{k+\frac{n}{2}}$ relatively,then we get:

$$\begin{aligned} f(\omega_n^k) &= f(\omega_{\frac{n}{2}}^k)_1 + \omega_n^k f(\omega_{\frac{n}{2}}^k)_2 \\ f(\omega_n^{k+\frac{n}{2}}) &= f(\omega_{\frac{n}{2}}^k)_1 - \omega_n^k f(\omega_{\frac{n}{2}}^k)_2 \end{aligned} \tag{6-2-7}$$

Finally, we can convert to the transform target of FFT as below:

$$fun = fsolve(f(\omega_1^k) + f(\omega_2^k) = 1) \tag{6-2-8}$$

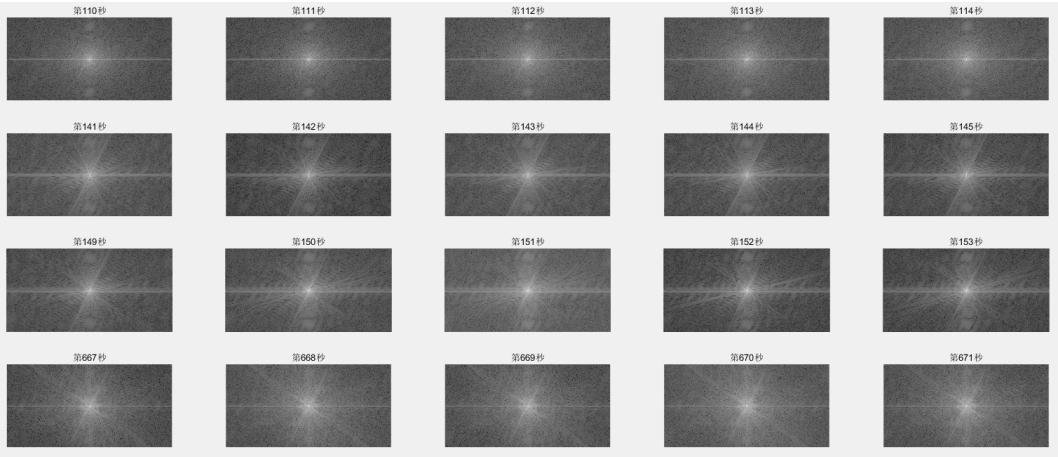

Figure 17 Image spectral frequency feature map

We selected 20 representative samples from the sample, and used fast Fourier transform to extract the frequency characteristics of the samples respectively:

6.2.5 Wavelet Feature Extractor

(1)Dobbercy Wavelet transform decomposition

The Coiflet wavelet is a discrete wavelet designed by Ingrid Daubechies. It can simultaneously have high vanishing momentum and its waveform is nearly symmetric, which is often used in digital signal processing. For image digital signal processing, it has strong extraction ability.

In order to improve the efficiency of the model, the use of discretization processing algorithm can greatly reduce the space complexity, and dyadic wavelet is commonly used to deconstruct two-dimensional image information:

2-Dimensional continuous wavelet transform:

$$W_f(a, b_x, b_y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) \Psi_{a_x b_x b_y}(x, y) = dxdy \tag{6-2-9}$$

Inverse 2-Dimensional continuous wavelet transform

$$f(x, y) = \frac{1}{C_\Psi} \int_0^{\infty} \int_{-\infty}^{\infty} \int_{-\infty}^{0} W_f(a, b_x, b_y) \Psi_{a_x b_x b_y}(x, y) db_x db_y \frac{da}{a^{3s}} \tag{6-2-10}$$

Two-dimensional discrete wavelet transform is the extension of one-dimensional discrete wavelet

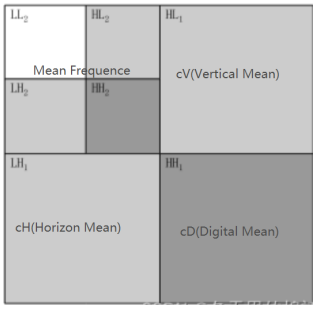

Figure 18 Component structure diagram of the Dobbercy wavelet decomposition

transform, which decomposes the signal in different scales, so as to obtain the myopic value and detail value of the original signal. Since the signal is two-dimensional, the decomposition is also two-dimensional; The results of the decomposition are: approximate components; Horizontal detail component; Vertical detail component; Diagonal detail component, in this paper, the image information is considered as the low-frequency input excitation as below:

1) Mean low frequency(cA)

2) Horizontal high frequency mean(cH)

3) Vertical high frequency mean(cV)

4) Diagonal high-frequency mean(cD)

Using the Dobbercy wavelet, we extracted wavelet features for the detection image in problem 1, as shown in the figure is the wavelet feature curve during melting and crystallization.

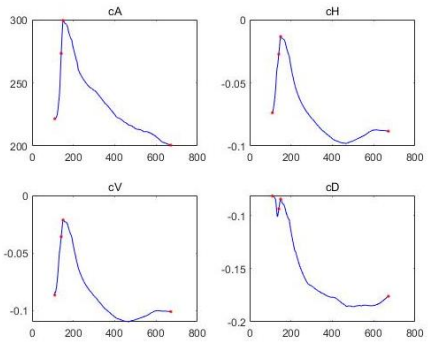

Figure 19 Graph of wavelet characteristic during melting and crystallization

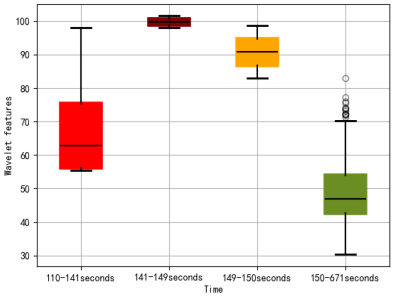

Figure 20 Wavelet features of adjacent sequence images of melting crystallization process

6.3 Model solving process

Based on the above research model, we successively extract five main types of image features as feature indicators in the smelting process. Due to the large number of selected indicators, it is necessary to screen the quality of indicators. We assume that the principal component analysis (PCA) is calculated between adjacent sequence images in the melting and crystallization process of protective slag by using the five groups of features extracted in time series:

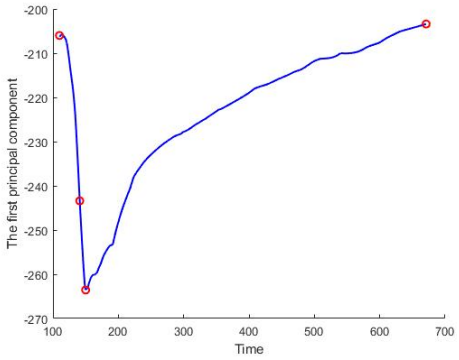

Figure 21 PCA feature quantity dimensionality reduction curve

After the verification of principal component analysis, the extracted image features in the five fields have strong relationships with the smelting and crystallization process of ore samples. We fit the model by using the Gaussian process linear model with five groups of feature data, and the model parameters are as shown in Table 2:

$$f(x) = a_1 e^{\frac{-(x-b_1)^2}{c_1}} + a^2 e^{\frac{-(x-b_2)^2}{c_2}} \tag{6-3-1}$$

According to the results of the model fitting, we plotted the melting and crystallization process curves of the protective slag in the melting and crystalline states:

The melting and crystallization process curves of the protection slag in, in the melting stage:, the dynamic differences between adjacent sequence images do not change significantly, and the image characteristics of key nodes are similar, which are all highlighted. In the crystallization stage: Since the sample in the figure is rapidly cooled and crystallized, the characteristics of the crystallization stage change rapidly, and the change rate gradually slows down.

Table 2 Generic Gaussian model fitting parameters

Parameter	Min	Average	Max
a_1	0.92	1.16	1.40
b_1	723.50	810.90	898.30
c_1	97.56	304.20	510.70
a_2	-0.19	0.20	0.60
b_2	316.10	429.20	542.40
c_2	151.90	213.10	274.40

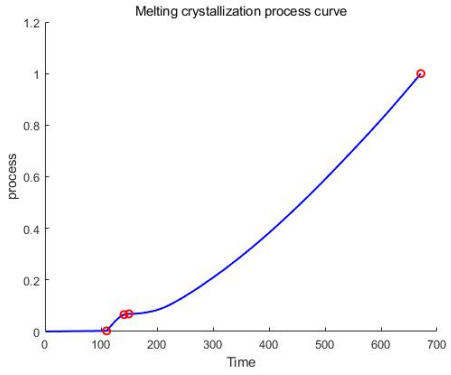

Figure 22 Melting and crystallization process curves

VII. Model of Task 3

7.1 Rough analysis of relationship between temperature and time

Based on the above extraction model of melting crystallization image features and the temperature data in Annex 2, we begin to analyze the temperature timing relationship.

The data in Annex 2 are extracted from the image set in Annex 1, which is limited to the accuracy of OCR. Within the allowable range of error, the regression method can be used to analyze the relationship between time and temperature.

We project the data extracted from task 1 into a three-bit space with time as the axis, and obtain the surface of the temperature of the two couples with respect to time, as shown in Figure 22:

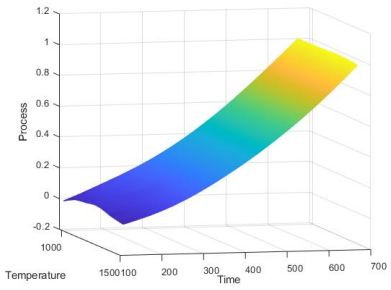

Figure 23 The surface of the thermocouple temperature with respect to time

From the figure, it can be found that the surface is smooth without singularities and has strong linear correlation.Since there is a linear relationship between the variables, we consider analyzing the expression of the temperature using a polynomial regression model-Gaussian process regression is the typical polynomial regression model.

7.2 Model Building——Gaussian process Regression

7.2.1 Gaussian process In statistics, a Gaussian process refers to a probabilistic model in which observations occur in a continuous domain (Figure 23). In a Gaussian process, a finite set of any random variables is required to satisfy a normal distribution, and continuous points in the input space are correlated with normally distributed random variables. Gaussian process is commonly used in machine learning to measure the similarity of kernel function in sample space. Based on a large amount of training data, Gaussian process can be used to predict points distributed at the edge of probability space. The Gaussian process model is highly portable and suitable for regression prediction of nonlinear models.

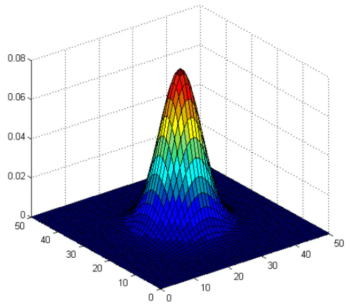

Figure 24 The surface of 2-dimensional Gaussian surface distribution

7.2.2 Principle of multivariate Gaussian distribution

Multivariate Gaussian distribution extends Gaussian distribution from one-dimensional variable to high latitude space, which makes Gaussian process regression model have stronger applicability. Multivariate Gaussian distributions in Gaussian probability compensation functions are derived as Equation 7-2-1 below:

$$\begin{aligned} P(x_1, x_2 \cdots, x_n) &= \prod_{i=1}^{n} p(x_i) \\ &= \frac{1}{\prod_{j=1}^{n} \sigma_j \sqrt{(2\pi^{\pi})}} exp(-\frac{1}{2} \sum_{i=1}^{n} \frac{(x_i - \bar{x}_i)^2}{2\sigma_i^2}) \end{aligned} \tag{7-2-1}$$

7.2.3 Gaussian process regression Model

Step 1. Determine the sampling point:

$$\begin{aligned} X &= [x_1, x_2, \cdots, x_n] \\ Y &= [y_1, y_2, \cdots, y_n] \\ X &\sim N(\bar{x}(t), C(t, t')) \end{aligned} \tag{7-2-2}$$

Step 2. Identify the sample centers:

In this model, in order to facilitate model derivation, the mean value $\bar{x_i}$is taken here so that the Gaussian distribution center is located at the origin

Step 3. Squared Exponential Kernel function of Gaussian regression:

$$c(x, x'|\theta) = \sigma_f^2 e^{\frac{-(x_i - x_j)^T (x_i - x_j)}{2\sigma_l^2}} \tag{7-2-3}$$

$$C(x, x') = \begin{bmatrix} c(x_1, x_1') & c(x_1, x_2') & \cdots & c(x_1, x_n') \\ c(x_2, x_1') & c(x_2, x_2') & \cdots & c(x_2, x_n') \\ \vdots & \vdots & \ddots & \vdots \\ c(x_n, x_1') & c(x_n, x_2') & \cdots & c(x_n, x_n') \end{bmatrix} \tag{7-2-4}$$

After debugging the model, the optimal parameters of Gaussian kernel function are as follows.

$\sigma_f = 0.245369 \qquad \sigma_l = 0.013485$

Step 4. Calculate the posterior probabilities:

According to Bayes theorem, based on the data set of observations (X, Y), the posterior probability distribution of the Gaussian process can be calculated as follows:

$$p(f \mid X, Y) \propto p(Y \mid X, f)p(f) \tag{7-2-5}$$

Step 5. Regression fitting was performed:

Based on the posterior probability distribution, time series samples in the data set can be collected and the selected feature indicators can be analyzed in turn:

$$\begin{bmatrix} y \\ y^* \end{bmatrix} \sim N\left(0, \begin{bmatrix} C & C_*^T \\ C_* & C_{**} \end{bmatrix}\right) \tag{7-2-6}$$

The matrix representation of covariance matrix, $C_*^T = C_*$,is predicted data collection observation covariance matrix between the data sets, C_*said prediction covariance matrix of the data set; The resulting predicted values satisfy the following probability constraints:

$$p(y^* \mid x^*, X, Y) = \int p(y^* \mid f^*)p(f^* \mid x^*, X, Y)df^* \tag{7-2-7}$$

7.3 Model solving process

7.3.1 Gaussian regression predicts thermocouple temperature

According to the multivariate Gaussian process regression model established above, we regressed the # 1 and # 2 thermocouple temperatures extracted in Annex 2 on the image time respectively, so as to predict the corresponding thermocouple temperature at a certain time.

7.3.2 Numerical simulation and fitting

We used Gaussian process regression as the generator of the numerical simulation, and plotted the scatter plot using multiple sets of time prediction thermocouple plots. By observing the scatter plot of melting and crystallization phases, we found that the scatter plot of temperature change rate in melting phase showed a quadratic function distribution, and the scatter plot of temperature change rate in crystallization phase showed a first-order function distribution.

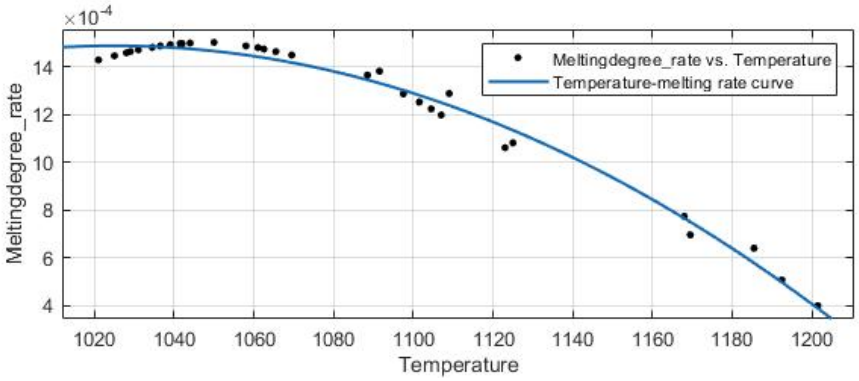

Figure 25 Temperature-melting rate curve

After predicting the temperature rate scatter through computer simulation, we used a quadratic linear regression model to regressively fit the temperature change rate during melting, and obtained the thermodynamic relationship curve of the melting stage:

$$f(x) = p_1 x^2 + p_2 x + p_3 \tag{7-3-1}$$

Table 3 Model parameters of temperature and time during melting

Parameter	Min	Average	Max
p_1	-4.08E-08	-3.53E-08	-2.97E-08
p_2	6.01E-05	7.23E-05	8.46E-05
p_2	-0.04233	-0.03557	-0.02881

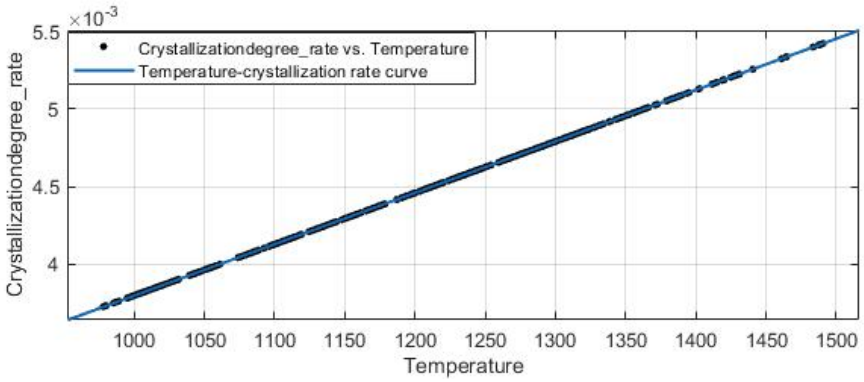

Figure 26 Temperature-time quadratic regression curve of the melting process

Similarly, we used computer simulation to predict the thermodynamic parameters of the mold for regression, and used a linear fitting model to regression the thermodynamic relationship curve of the crystallization stage. The relationship between temperature, melting rate and crystallization rate can be shown in the curve.

$$f(x) = p_1 x + p_2 \tag{7-3-2}$$

Table 4 Model parameters of temperature and time during crystallization

Parameter	Min	Average	Max
p_1	3.31E-06	3.31E-06	3.31E-06
p_2	4.84E-04	4.84E-04	4.84E-04

VIII. Strengths and Weakness

8.1 Evaluation of Model 1

8.1.1 Strengths of CNOCR

CNOCR uses a variety of network models, the effect of region detection and content recognition is high, and the program size is small, and the code is not difficult to write. It can re-train the text recognition model based on the built-in model, which has excellent portability.

8.1.2 Weakness of CNOCR

CNOCR is mainly aimed at typesetting simple printed text pictures, such as screenshots, scans, etc. The current built-in text detection and line layout modules cannot handle complex text typesetting positioning. If it is used for scene text image recognition, it needs to be used in combination with other scene text detection engines, such as cnstd, which is also based on MXNet. At the same time, due to the limited CNOCR users, the code lacks maintenance and updates are slow.

8.2 Evaluation of Model 2

8.2.1 Strengths of 5-domains feature extractor

(1) Grayscale feature

It is very simple, can store stable grayscale data, the computer grayscale image processing response quickly, in line with the law of binary operation.

(2) Color feature

Using color moments to shave color features does not need color space quantization, and the dimension of feature vector is low.

(3) LBP feature

LBP algorithm has low computational complexity, no training and illumination invariance, and is easy to be implemented in engineering.

(4) Frequency feature

The spectrum of the image is center symmetric, and the fast Fourier transform is used to expand. The model has universal applicability and can explain the time domain.

(5) WaveLet feature

Wavelet transform can not only reflect the distribution of frequency energy, but also retain the spatial distribution of image features. The Dobbercy wavelet transform has good properties, and it can retain the original information after multiple compression.

8.2.2 Weakness of 5-domains feature extractor

(1)Grayscale feature

The time complexity of processing grayscale images is large. Due to the small amount of information after binarization, it may not be normal to judge and recognize in complex scenes.

(2)Color feature ‘

But the retrieval efficiency of color moments is low. In practical applications, color moments are often used to filter images to narrow the retrieval range.

(3)LBP feature

The disadvantage of LBP features is that it is less robust over planar image regions. In the planar image region, the intensity difference is small and highly affected by image noise.

(4)Frequency feature

The conditions for existence are harsh and the time-frequency resolution cannot be satisfied simultaneously.

(5)WaveLet feature

Using wavelet transform to extract features, inevitably compressed image features, image distortion.

8.3 Evaluation of Model 3

8.3.1 Strengths of Gaussian process regression

The Gaussian process regression model is analyzed by the observation results using a kernel function, which reflects the characteristics of the input set. The output data set finally presents Gaussian distribution, which can be adjusted by adjusting the confidence interval and so on, and the model can be generalized by adjusting the kernel function.

10.3.2 Weakness of Gaussian process regression

The Gaussian process regression model is easy to fall into the curse of dimensionality for high latitude feature cases, and the prediction is invalid, the error is too large, and the noise is difficult to eliminate. GPR is a probabilistic analysis method based on random process, but it also has fatal shortcomings: high space complexity and large amount of calculation. When the amount of data reaches a certain scale, the model is difficult to converge and easy to fall into the trap of ”overfitting”.

IX. Future Work

9.1 Improvement of Model 1

Check the image quality to ensure the identifiability of the original image. A better OCR algorithm is designed by combining the training set of the problem. The algorithm model is used to iteratively train the image size that is most suitable for identifying temperature data, improve the effective ppi, and eliminate image noise.

9.2 Improvement of Model 2

(1) Grayscale feature

An improved method of gray feature extraction using co-occurrence gray matrix is mentioned in the literature[1].The improved algorithm reduces the computational complexity and greatly reduces the time of image texture feature extraction.

(2) Color feature ‘

A color feature extraction method based on opportunistic gamut component is presented in the literature[2], which has lower time complexity and wider applicability than using color moments.

(3) LBP feature

In the literature[3],they propose propose the adjacency- and reinforced adjacency-based variants of LBP for complex real-world background modeling and model learning for object detection.which improves the quality and efficiency of LBP feature extraction.

(4) Frequency feature & WaveLet feature

In the literature[4], researchers have cleverly applied wavelet transform to extract frequency features. The combination of the two types of features greatly improves the efficiency of feature extraction. In this paper, a spatial frequency ratio image fusion method based on nonsampled lifting wavelet transform is proposed for low frequency components. Compared with the traditional wavelet transform, image fusion can preserve the edge details of the source images more effectively and obtain better fused images.

9.3 Improvement of Model 3

(1) Reduce denoising misjudgment

For the error problem of Gaussian regression: more accurate and sensitive denoising can be achieved by introducing filters. Literature[5] describes an optimization scheme of Gaussian process regression denoising using the ”UKF improved square root” method, which can effectively reduce the prediction error and denoising misjudgment.

(2) Reduce the complexity

If the complexity of the model is too large, we can consider reducing the dimensionality of the prediction set and the observation set. The commonly used dimensionality reduction methods can be divided into three categories: data subset methods, reduced-rank approximation methods, and sparse pseudo-input methods. Literature[6] provides a novel dimensionality reduction method, which can use the ”thin plate spline hidden variable” to realize the dimensionality reduction of Gaussian process, and then perform supervised gradient learning, which can better reduce the complexity of regression model.

X. Conclusions

10.1 Conclusions of the problem

- In this paper, the time variation law of slag layer structure in the process of ore smelting is studied. Firstly, we recognize the training images, and conduct a secondary development based on CNOCR to train the text recognition model for the temperature of the thermocouple. Experiments show that the recognition accuracy of CNOCR model is 98.5
- Then, in order to analyze the change state of the samples in each stage during the melting and crystallization, we respectively extracted the features of gray feature, color feature, IBP feature, frequency feature and wavelet feature from the key frame image. The box plots can reflect that the five features are obvious. Then we performed principal component analysis on the extracted eigenvectors, and plotted the melting crystallization process curve according to the results of PCA.

- In addition, by establishing a Gaussian process regression model, we analyze the relationship between temperature and time. By analyzing the numerical simulation results based on Gaussian process regression prediction, we find that the scatter plot of temperature change rate in melting period shows a quadratic function distribution, and the scatter plot of temperature change rate in crystallization period shows a first-order function distribution. The fitting curves for the two states are plotted accordingly.

10.2 Methods used in our models

- This paper uses the graph feature extraction algorithm commonly used in computer graphics processing, and uses CNOCR as a framework for secondary development
- This paper uses a variety of regression models, including general Gaussian regression, Gaussian process regression and polynomial fitting model, using principal component analysis and other methods.

10.3 Applications of our models

- The research results of this paper may be applied to Chinese character recognition, computer graphics analysis and image processing .
- The relevant conclusions can also guide metal smelting application scenarios such as ore smelting process improvement and blast furnace device improvement.

References

[1] GONG J, LI X. Improved texture feature extraction algorithm based on co-occurrence matrix [J]. Computer Engineering and Design, 2011, 32(6): 2068-2071.

[2] CHEN W T, LIU W C, CHEN M S. Adaptive color feature extraction based on image color distributions[J]. IEEE Transactions on image processing, 2010, 19(8): 2005-2016.

[3] ACHARYA S, NANDA P K. Adjacent lbp and ltp based background modeling with mixed-mode learning for foreground detection[J]. Pattern Analysis and Applications, 2021, 24(3): 1047-1074.

[4] ZHENGJUN Y, HUAFENG L, RUIJING S, et al. Improving spatial frequency ratio of lifting wavelet transform for image fusion[D]. 2009.

[5] LI P, SONG S, DUAN G. Improved square-root ukf and its application in rendezvous and docking[J]. Electric Machines And Control, 2010, 14(11): 5.

[6] JIANG X. Research on dimensionality reduction method based on gaussian process[D]. Huazhong University of Science and Technology, 2012.

XI. Appendix

Annex A: Attachment List

Table 5 List of supporting materials in Attachment

Folder Name	File Content	File Description
task1		Accessory Package Of Task 1
	task1.py	Code Of Task 1 CNOCR
	task1.m	Code Of Task 1 Regression
	data.mat	Dataset Of Task 1
	Attachment 2.xlsx	Target Attachment2 datatable
task2		Accessory Package Of Task 2
	Features	Code Package Of Feature Extractor
	PCA	Code Of Principal Component Analysis
	task2(matlab)	Code Of Gaussian Regression
task3		Accessory Package Of Task 3
	task3.m	Code Of Gaussian Processing Regression
	wc.m	Code Of Wireless change
	data.mat	Matlab Dataset File
	traindata.mat	trainning Dataset
	Ave.jpg	trainning Dataset
	Temperature-crystallization rate curve.jpg	Graph of crystallization curve
	Temperature-melting rate curve.jpg	Graph of melting rate curve

Annex B: Code of Mathematical Model

Task 1– Code of CNOCR&Curve

task1.m

```
from cnocr import CnOcr
import pandas as pd
import re
ocr=CnOcr()
import pandas as pd
from PIL import Image
import os
class ReadIMG():
    def __init__(self,root_dir):
        self.rootdir=root_dir
        self.listdir=os.listdir(root_dir)
        self.lenth=len(self.listdir)
    def __getitem__(self, item):
        path_oldname=self.listdir[item]
        path=os.path.join(self.rootdir,path_oldname)
        img=Image.open(path)
        name=path_oldname.split('.')[0]
        return img,name
    def __len__(self):
        return self.lenth
rootdir=r'D:\Attachment 1'
rootimgs=ReadIMG(rootdir)

text_list=[]
name_list=[]
for img,name in rootimgs:
    text=[]
    name_list.append(name)
    res=ocr.ocr(img.crop((0,0,200,200)))
    for words in res:
        text.append(words['text'])
    text_list.append(text)

for i in range(len(text_list)):
    a=text_list[i]

    for j in range(len(a)):
        b=a[j]
        if ''==b:
           continue
        text_list[i][j]=re.findall('\d+',b)[-1]
print(text_list)
for i in range(len(text_list)):
    a=text_list[i]
    true_list=[]
```

```
    for j in a:
        if j == '':
            continue
        elif eval(j)<100 :
            continue
        elif eval(j)>2000 :
            true_list.append(str(eval(j)/10))
        else:
            true_list.append(j)
    text_list[i]=true_list

temp1=[]
temp2=[]
for temp in text_list:
    temp1.append(temp[1])
    temp2.append(temp[2])

df=pd.DataFrame({'NO':range(1,563),
                 'Time':range(110,672),
                 '1# Temperature':temp1,
                 '2# Temperature':temp2})
df.to_excel('Attachment 2.xlsx',index=False)
```

task1.py

```
load data.mat
%Original data drawing
figure
plot(data(:,1),data(:,2),'b-','linewidth',1.5)
hold on
grid on
plot(data(:,1),data(:,3),'r-','linewidth',1.5)
legend('1# Temperature','2# Temperature')
xlabel('Time(s)')
ylabel(' Temperature(℃)')

%average drawing
figure
plot(data(:,1),(data(:,2)+data(:,3))/2,'b-','linewidth',1.5)
hold on
grid on
legend('AVERAGE# Temperature')
xlabel('Time(s)')
ylabel(' Temperature(℃)')

%Moving average drawing
new_data=data;
new_data(:,2)=new_data(:,2)+new_data(:,3)/2;
new_data(:,2)=cumsum(new_data(:,2))./(1:size(new_data,1))';
figure
```

```
plot(new_data(:,1),new_data(:,2),'b-','linewidth',1.5)
hold on
grid on
legend('Moving average# Temperature')
xlabel('Time')
ylabel(' Temperature')
```

Task 2– Code of Feature extractor&Regression

task2.m

```
%% task2
clear;clc
%plot-GLCM_feature-imgs
load GLCM_data.mat
glcm_data=table2array(GLCM_data);
[m,n]=size(glcm_data);
GLCM_name=GLCM_data.Properties.VariableNames;
for i =1:n

    subplot(3,3,i);
    glcm_data(:,i)=cumsum(glcm_data(:,i))./(1:size(glcm_data,1))';%needsave
    %Moving average smoothing
    plot((1:size(glcm_data,1))+109,glcm_data(:,i),'b-')
    hold on
    plot([110,141,150,671],glcm_data([110,141,150,671]-109,i),'r.')
    title(GLCM_name{i})
    hold off

end
load Color_data.mat
color_data=table2array(Color_data);
[m,n]=size(color_data);
Color_name=Color_data.Properties.VariableNames;
for i =1:n

    subplot(3,3,i);
    color_data(:,i)=cumsum(color_data(:,i))./(1:size(color_data,1))';%needsave
    %Moving average smoothing
    plot((1:size(color_data,1))+109,color_data(:,i),'b-')
    hold on
    plot([110,141,150,671],color_data([110,141,150,671]-109,i),'r.')
    title(Color_name{i})
    hold off
end


load LBP_data.mat
lbp_data=table2array(LBP_data);
[m,n]=size(lbp_data);
```

```
LBP_name=LBP_data.Properties.VariableNames;
for i =1:n

    subplot(2,2,i);
    lbp_data(:,i)=cumsum(lbp_data(:,i))./(1:size(lbp_data,1))';%needsave
    %Moving average smoothing
    plot((1:size(lbp_data,1))+109,lbp_data(:,i),'b-')
    hold on
    plot([110,141,150,671],lbp_data([110,141,150,671]-109,i),'r.')
    title(LBP_name{i})
    hold off
end



load DB_data.mat
db_data=table2array(DB_data);
[m,n]=size(db_data);
DB_name=DB_data.Properties.VariableNames;
for i =1:n

    subplot(2,2,i);
    db_data(:,i)=cumsum(db_data(:,i))./(1:size(db_data,1))';%needsave
    %Moving average smoothing
    plot((1:size(db_data,1))+109,db_data(:,i),'b-')
    hold on
    plot([110,141,150,671],db_data([110,141,150,671]-109,i),'r.')
    title(DB_name{i})
    hold off
end


clc;clear
%Frequency domain characteristics
Frequency_data=zeros(562,1);
t=0;
location1=(558:758);
location2=(696:1096);
for i=1:562
    Img=double(rgb2gray(imread(['D:\Attachment 1\0' num2str(109+i) '.bmp'])));
    g = fft2(Img); % 2D Fast Fourier Transform
    g = fftshift(g); % 中心化处理
    return_img = log(1+abs(g));
    return_img = uint8(mat2gray(return_img(location1,location2)) * 255);
    Frequency_data(i)=mean(mean(return_img));
    if find(i+109==[110:114 141:145 149 150:153 667:671])
        t=t+1;
        subplot(5,5,t)
        imshow(uint8(return_img));
```

```
        title(['第' num2str(i+109) '秒'])
    end
end
figure()
Frequency_data=cumsum(Frequency_data)./(1:562)';
plot((1:562)+109,Frequency_data,'b-')
hold on
plot([110,141,150,671],Frequency_data([110,141,150,671]-109,1),'r.')
title('Frequency-feature')
```

code.m

```
%% Principal component analysis PCA
clc;clear
load x.mat;
[m,n]=size(x);%M samples, n indicators
% Z-score Standardized treatment
X=zscore(x);
% Calculate the covariance matrix of the standardized matrix.
R=cov(X);
disp('Correlation coefficient matrix:')
disp(R)
% Calculate the eigenvector and eigenvalue of covariance matrix
[V,D]=eig(R);%V corresponds to the eigenvector, and D is the corresponding eigenvalue
    of diagonal matrix.
V=rot90(V)';
% Calculate the contribution rate of principal component and cumulative contribution
    rate.
cv=diag(D);%Get the diagonal element of D, that is, the eigenvalue.
cv=cv(end:-1:1);
p=cv/n;%Contribution rate of each index
sum_p=cumsum(p);%Get cumulative contribution rate
disp('Contribution rate:')
disp(p')
disp('Cumulative contribution rate')
disp(sum_p')
num=input('The number of principal components according to the cumulative contribution
    rate (reference value > 0.8)');
disp('The load of the obtained principal component is')
disp(V(:,1:num))
PCA_data=x*V(:,1:num);

figure()
hold on
plot((1:562)+109,PCA_data(:,1),'b-','linewidth',1.5)
plot([110,141,150,671],PCA_data([110,141,150,671]-109,1),'ro','linewidth',1.5)
xlabel('Time')
ylabel('The first principal component')

process=zeros(671,2);
```

```
process(:,1)=1:671;
process_data=(PCA_data(:,1)-min(PCA_data(:,1)))/(max(PCA_data(:,1))-min(PCA_data(:,1)));
PCAsum=sum(process_data(:,1));
PCA_cumsum=cumsum(process_data(:,1));
process(110:end,2)=PCA_cumsum./PCAsum;
process(1:110,2)=linspace(0,process(110,2),110);
figure()
hold on
plot(process(:,1),process(:,2),'b-','linewidth',1.5)
%plot(process(110:end,1),process(110:end,2)+randn(562,1)*0.02,'k--','linewidth',1)
plot([110,141,150,671],process([110,141,150,671],2),'ro','linewidth',1.5)
xlabel('Time')
ylabel('process')
title('Melting crystallization process curve')
```

plot.py

```
# Import the relevant libraries
import pandas as pd
from matplotlib import pyplot as plt
# -----------------------------

def outlier(data):
    d = data.copy(deep=True)
    columns = d.columns
    out_index_ = []
    for col in columns: # Each column is identified with a box plot
        Q1 = d[col].quantile(q=0.27) # Lower quartile
        Q2 = d[col].quantile(q=0.7) # Upper quartile
        low_whisker = Q1 - 1.5 * (Q2 - Q1) # Lower edge section
        up_whisker = Q2 + 1.5 * (Q2 - Q1) # Upper edge section

        # Look for abnormal value points, obtain outlier subscripts, and delete data
        r = (d[col] > up_whisker) | (d[col] < low_whisker)
        out = d[col].index[r]
        out_index_ += out.tolist()
    d.drop(out_index_, inplace=True)
    return d
# Import data
dir = "D:\\python\\AEBP\\DG\\1.xlsx"
data_frame = pd.read_excel(dir, sheet_name='Sheet1', usecols='A:D')
columns = data_frame.columns
data_need = data_frame[columns]
df = pd.DataFrame(data_need)
df = outlier(df) # Handle exception data
# -----------------------------
# Drawing
f = df.boxplot(patch_artist=True,return_type='dict')
color = ['#FF0000','#800000','#FFA500','#6B8E23'] # Set the corresponding color
# ----------------------------------
```

```
for box, c in zip(f['boxes'],color):
    # Border color
    box.set(color=c, linewidth=4)
    # The color inside the box
    box.set(facecolor=c)
# Set the box properties
for whisker in f['whiskers']:
    whisker.set(color='black', linewidth=1.5)
for cap in f['caps']:
    cap.set(color='black', linewidth=2)
for median in f['medians']:
    median.set(color='black', linewidth=1.5)
for flier in f['fliers']:
    flier.set(marker='o', color='y', alpha=0.5)
# ----------------------------------------
# Set the x,y label
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.ylabel("Wavelet features",labelpad=2)
plt.xlabel("Time")
plt.show()
```

LBP feature functions

```
# Import the relevant libraries
import pandas as pd
from PIL import Image
from pylab import *
import os

# Define the LBP instance class
class LBP:
    def __init__(self):
        # 旋转不变模式的36种特征值从小到大进行序列化编号
        self.revolve_map = {0: 0, 1: 1, 3: 2, 5: 3, 7: 4, 9: 5, 11: 6, 13: 7, 15: 8,
            17: 9, 19: 10, 21: 11, 23: 12,
                        25: 13, 27: 14, 29: 15, 31: 16, 37: 17, 39: 18, 43: 19, 45:
                            20, 47: 21, 51: 22, 53: 23,
                        55: 24,
                        59: 25, 61: 26, 63: 27, 85: 28, 87: 29, 91: 30, 95: 31, 111:
                            32, 119: 33, 127: 34, 255: 35}
        # 等价模式的58种特征值从小到大进行序列化编号
        self.uniform_map = {0: 0, 1: 1, 2: 2, 3: 3, 4: 4, 6: 5, 7: 6, 8: 7, 12: 8,
                        14: 9, 15: 10, 16: 11, 24: 12, 28: 13, 30: 14, 31: 15, 32: 16,
                        48: 17, 56: 18, 60: 19, 62: 20, 63: 21, 64: 22, 96: 23, 112:
                            24,
                        120: 25, 124: 26, 126: 27, 127: 28, 128: 29, 129: 30, 131: 31,
                            135: 32,
                        143: 33, 159: 34, 191: 35, 192: 36, 193: 37, 195: 38, 199: 39,
                            207: 40,
                        223: 41, 224: 42, 225: 43, 227: 44, 231: 45, 239: 46, 240: 47,
```

```
                        241: 48,
                    243: 49, 247: 50, 248: 51, 249: 52, 251: 53, 252: 54, 253: 55,
                        254: 56,
                    255: 57}

# Convert to grayscale map to obtain the pixel information of grayscale map
def describe(self, image):
    image_array = np.array(Image.open(image).convert('L'))
    image_array = np.array(image_array).astype(np.float32)
    return image_array
# --------------------------------------------------------------
# LBP raw feature calculation algorithm: Compare the pixels at the specified
    location of the image with the surrounding 8 pixels
# Points larger than the center pixel are assigned 1, and points smaller than the
    center pixel are assigned 0, returning the resulting binary sequence
# --------------------------------------------------------------
def calute_basic_lbp(self, d_array, i, j):
    sum = []
    if d_array[i - 1, j - 1] > d_array[i, j]:
        sum.append(1)
    else:
        sum.append(0)
    if d_array[i - 1, j] > d_array[i, j]:
        sum.append(1)
    else:
        sum.append(0)
    if d_array[i - 1, j + 1] > d_array[i, j]:
        sum.append(1)
    else:
        sum.append(0)
    if d_array[i, j - 1] > d_array[i, j]:
        sum.append(1)
    else:
        sum.append(0)
    if d_array[i, j + 1] > d_array[i, j]:
        sum.append(1)
    else:
        sum.append(0)
    if d_array[i + 1, j - 1] > d_array[i, j]:
        sum.append(1)
    else:
        sum.append(0)
    if d_array[i + 1, j] > d_array[i, j]:
        sum.append(1)
    else:
        sum.append(0)
    if d_array[i + 1, j + 1] > d_array[i, j]:
        sum.append(1)
    else:
```

```
            sum.append(0)
        return sum
    # ------------------------------------------------------------
    # The binary sequence rotates in a continuous loop to obtain the minimum decimal
        value of the new binary sequence
    def get_min_for_revolve(self, arr):
        values = []
        circle = arr
        circle.extend(arr)
        for i in range(0, 8):
            j = 0
            sum = 0
            bit_num = 0
            while j < 8:
                sum += circle[i + j] << bit_num
                bit_num += 1
                j += 1
            values.append(sum)
        return min(values)

    # Gets the number of bits in the binary of the value r
    def calc_sum(self, r):
        num = 0
        while (r):
            r &= (r - 1)
            num += 1
        return num

    # Acquire the LBP raw mode features of the image
    def lbp_basic(self, image_array):
        basic_array = np.zeros(image_array.shape, np.uint8)
        width = image_array.shape[0]
        height = image_array.shape[1]
        for i in range(1, width - 1):
            for j in range(1, height - 1):
                sum = self.calute_basic_lbp(image_array, i, j)
                bit_num = 0
                result = 0
                for s in sum:
                    result += s << bit_num
                    bit_num += 1
                basic_array[i, j] = result
        return basic_array

    # Acquire the LBP rotation invariant mode feature of the image
    def lbp_revolve(self, image_array):
        revolve_array = np.zeros(image_array.shape, np.uint8)
        width = image_array.shape[0]
        height = image_array.shape[1]
```

```
        for i in range(1, width - 1):
            for j in range(1, height - 1):
                sum = self.calute_basic_lbp(image_array, i, j)
                revolve_key = self.get_min_for_revolve(sum)
                revolve_array[i, j] = self.revolve_map[revolve_key]
        return revolve_array

    # Obtain the LBP equivalent mode features of the image
    def lbp_uniform(self, image_array):
        uniform_array = np.zeros(image_array.shape, np.uint8)
        basic_array = self.lbp_basic(image_array)
        width = image_array.shape[0]
        height = image_array.shape[1]
        for i in range(1, width - 1):
            for j in range(1, height - 1):
                k = basic_array[i, j] << 1
                if k > 255:
                    k = k - 255
                xor = basic_array[i, j] ^ k
                num = self.calc_sum(xor)
                if num <= 2:
                    uniform_array[i, j] = self.uniform_map[basic_array[i, j]]
                else:
                    uniform_array[i, j] = 58
        return uniform_array

    # Acquire the LBP rotation invariant equivalent mode feature of the image
    def lbp_revolve_uniform(self, d_array):
        uniform_revolve_array = np.zeros(d_array.shape, np.uint8)
        basic_array = self.lbp_basic(d_array)
        width = d_array.shape[0]
        height = d_array.shape[1]
        for i in range(1, width - 1):
            for j in range(1, height - 1):
                k = basic_array[i, j] << 1
                if k > 255:
                    k = k - 255
                xor = basic_array[i, j] ^ k
                num = self.calc_sum(xor)
                if num <= 2:
                    uniform_revolve_array[i, j] = self.calc_sum(basic_array[i, j])
                else:
                    uniform_revolve_array[i, j] = 9
        return uniform_revolve_array

# Function calls
# ------------------------------------------------------
if __name__ == '__main__':
    image = "D:\\2022 APMCM Problems\\2022 APMCM Problem A\\Attachment 1"
```

```
    image_path = os.listdir(image)
    s = []
    for i in range(len(image_path)):
        image0 = image + '\\' + image_path[i]
        lbp = LBP()
        p = []
        image_array = lbp.describe(image=image0)
        # Original LBP features
        basic_array = lbp.lbp_basic(image_array)
        p.append(basic_array.mean())
        # Rotation does not change the LBP feature
        revolve_array = lbp.lbp_revolve(image_array)
        p.append(revolve_array.mean())
        # Equivalent mode LBP features
        uniform_array = lbp.lbp_uniform(image_array)
        p.append(uniform_array.mean())
        # Rotation invariant equivalence mode LBP feature
        resolve_uniform_array = lbp.lbp_revolve_uniform(image_array)
        p.append(resolve_uniform_array.mean())
        s.append(p)
    # ---------------------------------------------
    # Transform the list matrix and put the data into a CSV file
    s = np.array(s)
    list1 = s[:, 0]
    list2 = s[:, 1]
    list3 = s[:, 2]
    list4 = s[:, 3]
    df = pd.DataFrame(
        {'Original LBP features': list1, 'Rotation does not change the LBP feature':
            list2, 'Equivalent mode LBP features': list3, 'Rotation invariant
            equivalence mode LBP feature': list4})
    df.to_csv('s_data_2.csv', index=False)
    # -------------------------------
```

Grayscale feature functions.py

```
# ----------------
# Import the relevant libraries
import os
import pandas as pd
from PIL import Image
import numpy as np
import cv2

#   ---------------------------------
#   Define a grayscale symbiotic matrix
def fast_glcm(img, vmin=0, vmax=255, nbit=8, kernel_size=5):
    mi, ma = vmin, vmax
    ks = kernel_size
    h, w = img.shape
```

```
    # digitize
    bins = np.linspace(mi, ma + 1, nbit + 1)
    gl1 = np.digitize(img, bins) - 1
    gl2 = np.append(gl1[:, 1:], gl1[:, -1:], axis=1)

    # make glcm
    glcm = np.zeros((nbit, nbit, h, w), dtype=np.uint8)
    for i in range(nbit):
        for j in range(nbit):
            mask = ((gl1 == i) & (gl2 == j))
            glcm[i, j, mask] = 1
    kernel = np.ones((ks, ks), dtype=np.uint8)
    for i in range(nbit):
        for j in range(nbit):
            glcm[i, j] = cv2.filter2D(glcm[i, j], -1, kernel)
    glcm = glcm.astype(np.float32)
    return glcm


#   ---------------------------------
#   mean
def fast_glcm_mean(img, vmin=0, vmax=255, nbit=8, ks=5):
    h, w = img.shape
    glcm = fast_glcm(img, vmin, vmax, nbit, ks)
    mean = np.zeros((h, w), dtype=np.float32)
    for i in range(nbit):
        for j in range(nbit):
            mean += glcm[i, j] * i / (nbit) ** 2
    return mean.mean()


#   ---------------------------------------
#   standard deviation
def fast_glcm_std(img, vmin=0, vmax=255, nbit=8, ks=5):
    h, w = img.shape
    glcm = fast_glcm(img, vmin, vmax, nbit, ks)
    mean = np.zeros((h, w), dtype=np.float32)
    for i in range(nbit):
        for j in range(nbit):
            mean += glcm[i, j] * i / (nbit) ** 2
    std2 = np.zeros((h, w), dtype=np.float32)
    for i in range(nbit):
        for j in range(nbit):
            std2 += (glcm[i, j] * i - mean) ** 2
    std = np.sqrt(std2)
    std = np.std(std)
    return std
```

```
#   ------------------------------------------------
#   contrast
def fast_glcm_contrast(img, vmin=0, vmax=255, nbit=8, ks=5):
    h, w = img.shape
    glcm = fast_glcm(img, vmin, vmax, nbit, ks)
    cont = np.zeros((h, w), dtype=np.float32)
    for i in range(nbit):
        for j in range(nbit):
            cont += glcm[i, j] * (i - j) ** 2
    return cont.mean()


#   ------------------------------------------------------
#   Dissimilarity
def fast_glcm_dissimilarity(img, vmin=0, vmax=255, nbit=8, ks=5):
    h, w = img.shape
    glcm = fast_glcm(img, vmin, vmax, nbit, ks)
    diss = np.zeros((h, w), dtype=np.float32)
    for i in range(nbit):
        for j in range(nbit):
            diss += glcm[i, j] * np.abs(i - j)
    return diss.mean()


#   ------------------------------------------------------------
#   Homogeneity/inverse gap
def fast_glcm_homogeneity(img, vmin=0, vmax=255, nbit=8, ks=5):
    h, w = img.shape
    glcm = fast_glcm(img, vmin, vmax, nbit, ks)
    homo = np.zeros((h, w), dtype=np.float32)
    for i in range(nbit):
        for j in range(nbit):
            homo += glcm[i, j] / (1. + (i - j) ** 2)
    return homo.mean()


#   ------------------------------------------------------
#   Angular second-order moment / energy
def fast_glcm_ASM(img, vmin=0, vmax=255, nbit=8, ks=5):
    h, w = img.shape
    glcm = fast_glcm(img, vmin, vmax, nbit, ks)
    asm = np.zeros((h, w), dtype=np.float32)
    for i in range(nbit):
        for j in range(nbit):
            asm += glcm[i, j] ** 2
    ene = np.sqrt(asm)
    ene = np.mean(ene)
    return asm.mean(), ene
```

```
#   ----------------------------------------------------
#   能量(Energy)
def fast_glcm_max(img, vmin=0, vmax=255, nbit=8, ks=5):
    glcm = fast_glcm(img, vmin, vmax, nbit, ks)
    max_ = np.max(glcm, axis=(0, 1))
    return max_.mean()


#   ----------------------------------------------------------
#   Entropy
def fast_glcm_entropy(img, vmin=0, vmax=255, nbit=8, ks=5):
    glcm = fast_glcm(img, vmin, vmax, nbit, ks)
    pnorm = glcm / np.sum(glcm, axis=(0, 1)) + 1. / ks ** 2
    ent = np.sum(-pnorm * np.log(pnorm), axis=(0, 1))
    return ent.mean()


# Function calls
if __name__ == '__main__':
    # --------------
    # Set the initial parameters
    nbit = 8
    ks = 5
    mi, ma = 0, 255
    # --------------------
    path = 'D:\\2022 APMCM Problems\\2022 APMCM Problem A\\Attachment 1'
    path1 = os.listdir(path)
    n_len = len(path1)
    key = np.zeros([n_len, 8]) # Define the receive zeros matrix
    s = []
    for i in range(n_len):
        p = []
        # print(p)
        # print(s)
        img = np.array(Image.open(path + '\\' + path1[i]).convert('L'))
        h, w = img.shape
        img = np.array(img).astype(np.float32)
        glcm_mean = fast_glcm_mean(img, mi, ma, nbit, ks) # mean
        p.append(glcm_mean)
        glcm_std = fast_glcm_std(img) # standard deviation
        p.append(glcm_std)
        glcm_cont = fast_glcm_contrast(img) # contrast
        p.append(glcm_cont)
        glcm_diss = fast_glcm_dissimilarity(img) # Dissimilarity
        p.append(glcm_diss)
        glcm_homo = fast_glcm_homogeneity(img) # Homogeneity/inverse gap
        p.append(glcm_homo)
        glcm_asm, glcm_ene = fast_glcm_ASM(img) # Angular 2nd Order Moment / Energy
```

```
            (ASM)
        p.append(glcm_asm)
        p.append(glcm_ene)
        glcm_max = fast_glcm_max(img) # 能量(Energy)
        p.append(glcm_max)
        glcm_ent = fast_glcm_entropy(img) # Entropy
        p.append(glcm_ent)
        s.append(p)
    s = np.array(s)
    list1 = s[:, 0]
    list2 = s[:, 1]
    list3 = s[:, 2]
    list4 = s[:, 3]
    list5 = s[:, 4]
    list6 = s[:, 5]
    list7 = s[:, 6]
    list8 = s[:, 7]
    list9 = s[:, 8]
    df = pd.DataFrame({'mean': list1, 'standard deviation': list2, 'contrast': list3,
                       'Dissimilarity': list4, 'Homogeneity/inverse gap': list5, 'Angular
                           second-order moment': list6,
                       'Energy(ASM)': list7, '能量(Energy)': list8, 'Entropy': list9})
    df.to_csv('s_data_1.csv', index=False)
```

Dobess wavelet transform functions.py

```
# Import the relevant libraries
import os
import pandas as pd
import pywt
import cv2
import numpy as np

if __name__ == '__main__':
    image = "D:\\2022 APMCM Problems\\2022 APMCM Problem A\\Attachment 1"
    image_path = os.listdir(image)
    s = []
    for i in range(len(image_path)):
        image0 = image + '\\' + image_path[i]
        p = []
        img = cv2.imread(image0, 0)
        cA, (cH, cV, cD) = pywt.dwt2(img, "db2")
        cA = np.mean(cA) # Low frequency mean
        p.append(cA)
        cH = np.mean(cH) # Horizontal high-frequency mean
        p.append(cH)
        cV = np.mean(cV) #Vertical high-frequency mean
        p.append(cV)
        cD = np.mean(cD) # Diagonal high-frequency mean
        p.append(cD)
```

```
        s.append(p)
        print(s)
    # Transform the list matrix and put the data into a CSV file
    s = np.array(s)
    list1 = s[:, 0]
    list2 = s[:, 1]
    list3 = s[:, 2]
    list4 = s[:, 3]
    df = pd.DataFrame(
        {'cA': list1, 'cH': list2,
         'cV': list3, 'cD': list4})
    df.to_csv('s_data_3.csv', index=False)
    # --------------------------------
```

Color characteristic functions.py

```
# Import the relevant libraries
import pandas as pd
from PIL import Image
import numpy as np
import os

"""
# The first-order color moment of the r channel
rd_1 = rd.mean()

# The second-order color moment of the r channel
rd_2 = rd.std()
"""


# A function that defines a third-order color moment
def var(x=None):
    mid = np.mean(((x - x.mean()) ** 3))
    return np.sign(mid) * abs(mid) ** (1 / 3)


# Find the color moments of each order of all pictures in a folder in batches
def GetData(path):
    file_path = os.listdir(path) # Get all the files under the path
    n_len = len(file_path)
    data = np.zeros([n_len, 9]) # Set up an empty data space
    for i in range(n_len):
        img = Image.open(path + '\\' + file_path[i]) # Open the image under the path
        r, g, b = img.split() # Get data for three channels
        rd = np.asarray(r, dtype=None, order=None)
        gd = np.asarray(g, dtype=None, order=None)
        bd = np.asarray(b, dtype=None, order=None)
        data[i, 0] = rd.mean() # Get a first-order color moment
        data[i, 1] = gd.mean()
```

```
        data[i, 2] = bd.mean()
        data[i, 3] = rd.std() # The second-order color moment is obtained
        data[i, 4] = gd.std()
        data[i, 5] = bd.std()
        data[i, 6] = var(rd) # The third-order color moment is obtained
        data[i, 7] = var(gd)
        data[i, 8] = var(bd)
    return data


if __name__ == '__main__':
    a = GetData(path="D:\\2022 APMCM Problems\\2022 APMCM Problem A\\Attachment 1")
    p = [] # Receive the list
    print(a) #View
    # ----------------------------------
    # Put the feature into the csv file
    list1 = a[:, 0]
    list2 = a[:, 3]
    list3 = a[:, 6]
    list4 = a[:, 1]
    list5 = a[:, 4]
    list6 = a[:, 7]
    list7 = a[:, 2]
    list8 = a[:, 5]
    list9 = a[:, 8]
    # Import nine-dimensional color features
    df = pd.DataFrame({'R1': list1, 'R2': list2, 'R3': list3, 'G1': list4,
                       'G2': list5, 'G3': list6, 'B1': list7, 'B2': list8, 'B3': list9})
    df.to_csv('s_data.csv', index=False)
```

Task 3– Code of Gaussian Process Regression

task3.m

```
%% task3
clc;clear;
load traindata.mat;
%trainThe first column is temperature 1, the second column is temperature 2,
%the third column is time and the fourth column is process.
x_predict=traindata; %No prediction is required, only fitting is required, so the
    prediction set is the training set.
[m,n]=size(x_predict);
input_feature=[1,3,4];
x=traindata(:,input_feature(1:2));%Training set
y=traindata(:,input_feature(end));%Training set output
temp=randperm(size(x,1));%Random scrambling of known sample data
num=round(size(x,1)*0.8);
train_x=x(temp(1:num),:);%Take m*0.8 training samples
train_y=y(temp(1:num),:);%Output of training samples
test_x=x(temp(num+1:end),:);%Test sample
```

```
test_y=y(temp(num+1:end),:);%Test sample output

% normalized dimensionless
[train_x0,P1]=mapminmax(train_x',0,1);
train_x0=train_x0';
[train_y0,P2]=mapminmax(train_y',0,1);
train_y0=train_y0';
test_x0=mapminmax('apply',test_x',P1)';
test_y0=mapminmax('apply',test_y',P2)';
    % model training
    yi=train_y0;
    gprMdl = fitrgp(train_x0,yi,'Basis','pureQuadratic',...
      'FitMethod','exact','PredictMethod','exact');
    % test set inspection
    test_predict=predict(gprMdl,test_x0);
    r0 =wc(test_predict,test_y0,n);%R_adjusted
    disp('After the training of the model is completed, the revised goodness of fit of
        the test set is:')
    disp([ 'R_adjusted=' num2str(r0)])

    % output forecast value
    out=predict(gprMdl,mapminmax('apply',x_predict(input_feature(1:2))',P1)');
    out1=mapminmax('reverse',out',P2)';
    y_predict=out1;




%% draw designs
x=x_predict(:,1);
y=x_predict(:,3);
[x,y]=meshgrid(x,y);
num=size(x,1);
z=zeros(size(x,1));
data=zeros(num*num,2);
for i=1:num
    for j=1:num
        data((i-1)*num+j,:)=[x(i,j),y(i,j)];
    end
end

new_data=predict(gprMdl,mapminmax('apply',data',P1)');
new_data=mapminmax('reverse',new_data',P2)';
for i=1:num
    for j=1:num
        z(i,j)=new_data((i-1)*num+j);
    end
```

```
end
mesh(x,y,z)
xlabel('Temperature')
ylabel('Time')
zlabel('Process')

%% computer simulation
temp_thaw=traindata(1:31,1);%It took 31 seconds for the sample to melt until it
    completely melted.
temp_crystal=traindata(40:end,1);%Crystallization from the 150th to the last.
time_thaw=traindata(1:31,3);
time_crystal=traindata(40:end,1);
data_thaw1=[temp_thaw,time_thaw];
data_thaw2=[temp_thaw,time_thaw+0.001];% The average speed for a very short time is
    equal to the current instantaneous speed.
data_crystal1=[temp_crystal,time_crystal];
data_crystal2=[temp_crystal,time_crystal+0.001];
predict_thaw_1=predict(gprMdl,mapminmax('apply',data_thaw1',P1)');
predict_thaw_2=predict(gprMdl,mapminmax('apply',data_thaw2',P1)');
predict_thaw_1=mapminmax('reverse',predict_thaw_1',P2)';
predict_thaw_2=mapminmax('reverse',predict_thaw_2',P2)';
thaw=[temp_thaw,(predict_thaw_2-predict_thaw_1)./0.001];%Temperature-melting rate
    characteristic data
predict_crystal_1=predict(gprMdl,mapminmax('apply',data_crystal1',P1)');
predict_crystal_2=predict(gprMdl,mapminmax('apply',data_crystal2',P1)');
predict_crystal_1=mapminmax('reverse',predict_crystal_1',P2)';
predict_crystal_2=mapminmax('reverse',predict_crystal_2',P2)';
crystal=[temp_crystal,(predict_crystal_2-predict_crystal_1)./0.001];%Temperature-crystallization
    rate characteristic data

figure()
plot(thaw(:,1),thaw(:,2),'bo')
figure()
plot(crystal(:,1),crystal(:,2),'bo')
%fit
Meltingdegree_rate=thaw(:,2);
Temperature=thaw(:,1);

Crystallizationdegree_rate=crystal(:,2);
Temperature=crystal(:,1);
```

wc.m

```
%% goodness of fit calculation function
% r_adjusted refers to the adjusted goodness of fit, which is the fitting degree of
    the regression line to the observed value.

function R2=wc(reverse_out,test_y,k)
R2=1-(sum((reverse_out-test_y).^2)/(size(reverse_out,1)-k-1))
/(sum((mean(test_y)-test_y).^2)/(size(reverse_out,1)-1));
```

```
end
```