How to Use the Software

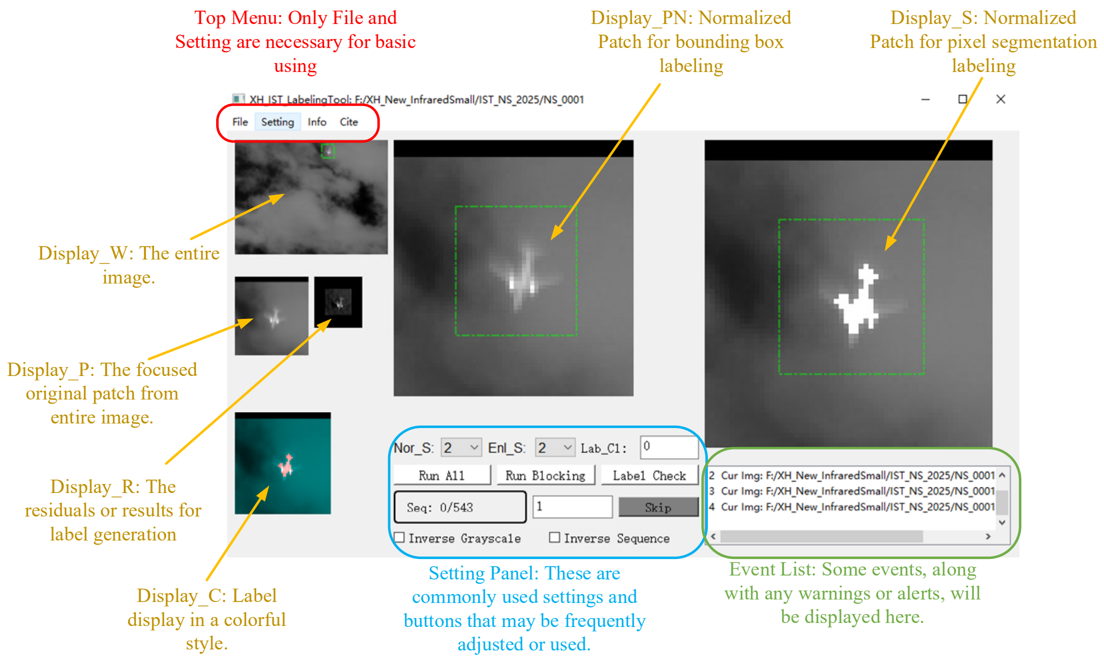
1. Basic Information

1.1 Version Information

This software is designed for annotating small infrared targets (ISTs), particularly suitable for targets not larger than 16×16 pixels. The software is released as an executable file (.exe) and can be used in a Windows environment. It is recommended to use it with a screen resolution of at least 1080P for optimal performance.

This software will streamline and accelerate infrared small target labeling, with improved accuracy under conditions of local zoom and stretching. We name it “XH_CLSBF_Label”. In this software, some assistant models will help generate a local bounding box and pixel-level segmentation labels in advance. Although these are not always perfectly accurate, they help reduce the labeling effort. The software can be obtained at: https://github.com/SeaHifly/CLSBF_software.git

1.2 Interface Overview


Top Menu: Only File and Setting are necessary for basic using
Display_PN: Normalized Patch for bounding box labeling
Display_S: Normalized Patch for pixel segmentation labeling
XH_IST_LabelingTool: F:/XH_New_InfraredSmall/IST_NS_2025/NS_0001
File
Setting
Info
Cite
Display_W: The entire image.
Display_P: The focused original patch from entire image.
Display_R: The residuals or results for label generation
Display_C: Label display in a colorful style.
Setting Panel: These are commonly used settings and buttons that may be frequently adjusted or used.
Event List: Some events, along with any warnings or alerts, will be displayed here.
Run All
Run Blocking
Label Check
Skip

Nor_S: 2 Enl_S: 2 Lab_C1: 0

Seq: 0/543 1

☐ Inverse Grayscale ☐ Inverse Sequence

```
2 Cur Img: F:/XH_New_InfraredSmall/IST_NS_2025/NS_0001
3 Cur Img: F:/XH_New_InfraredSmall/IST_NS_2025/NS_0001
4 Cur Img: F:/XH_New_InfraredSmall/IST_NS_2025/NS_0001
```

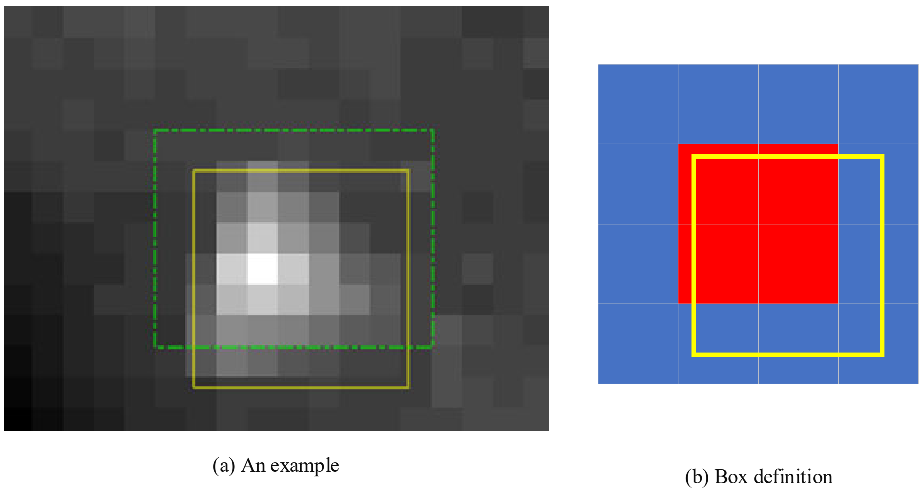
Figure 1 Basic introduction of the software

After double-clicking the executable and waiting for a few seconds or minutes, the user will be presented with the interface shown in Figure 1. The software includes six image display areas, a settings panel, and an event list.

1.2.1 Display introduction

Display_W: The entire image is shown here. The user needs to roughly outline the target area here with the mouse, especially when the auxiliary target detection model has not preset a bounding box or has made an incorrect prediction. The green box in Figure 1 represents the roughly outlined target area. The target's local area can be redefined repeatedly, but more detailed adjustments should be made elsewhere. In most cases, all displayed green boxes are identical/consistent, except that some displays show zoomed-in views of the local target area.

Display_PN: This is the target's bounding box marking area, where the user needs to accurately define the target's bounding box. Some details of the marking process are shown in Figure 2. When drawing the bounding box with the mouse, the top-left corner is included inside the box, while the bottom-right corner is excluded.


(a) An example

(b) Box definition

Figure 2 The definition of the bounding box labeling. The left-top corner is inside while the bottom-right corner is not.

Display_S: After drawing the target's bounding box in Display_PN, the user can mark the pixel-level segmentation of the target here. Areas outside the bounding box cannot be marked or labeled. Clicking the mouse will modify the segmentation label: if the clicked pixel is already marked as part of the target, it will be changed to the background; otherwise, it will be marked as the target pixel. By holding down the Ctrl key while pressing the mouse button, the user can continuously mark the target's segmented pixels. If a large area of incorrect labeling is found, especially in cases where the automatic label generation makes errors, the user can use a method similar to drawing a regression box to erase all pixel labels within the box quickly.

Display_P: This display shows the same patch as the ones shown in Display_PN, but the patch here is not normalized. No operation is allowed here, this area is for display and reference only.

Display_R: This display shows the residuals or results before the hard segmentation for label generation. No operation is allowed here, this area is for display.

Display_C: This display shows the segmentation label in a colorful style. If the pixels are not labeled, the pixels tend to be light/sky blue. If the pixels are labeled as target pixels in Display_S, the pixels tend to be red. Some examples are presented in Figure 3.

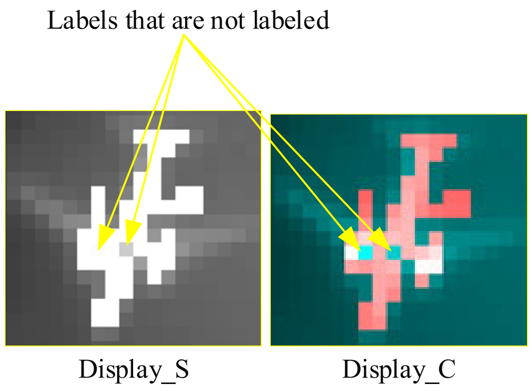


Figure 3 Some target pixels have high grayscale values, which may be missed during labeling, and this issue is particularly noticeable in Display_C.

1.2.2 Setting Panel

These are settings that may be used frequently.

Nor_S (normalized size coefficient): This coefficient refers to the statistical range of the stretching threshold in Display_PN. In Display_PN, the target's local image is stretched using the minimum and maximum values, with the statistical range defined by Nor_S. When set to 2, the width and height of the statistical area for the minimum and maximum values will be twice the size of the corresponding bounding box (the target's approximate or precise bounding box, as the presented green box in Figure 1).

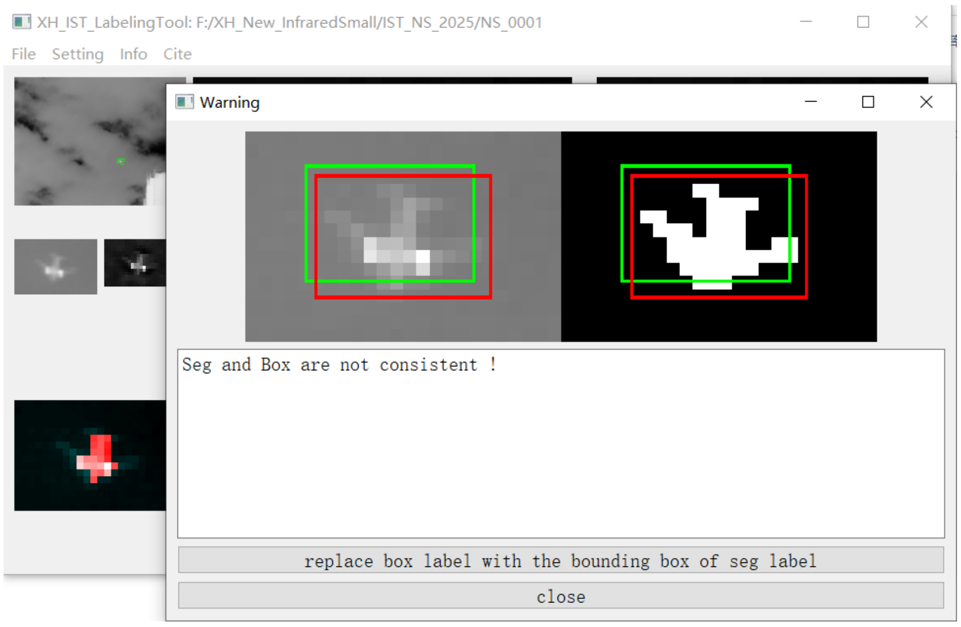
Enl_S (enlarged size coefficient): This coefficient determines the size of the local display area in Display_PN and Display_S. When set to 2, the width and height of the displayed local image will be twice the size of the green bounding box.

Lal_Cl (Label Class): The bounding box labels are stored in the TXT file in the format of (category/class, center coordinates x, center coordinates y, bounding box width, bounding box height). The category of the target can be defined here even if it is not required in the infrared small target detection task. In future work, it may be possible to recognize different target categories through sequence fusion.

Run All: This button, when clicked, invokes the pre-configured detection model to process the sequence of images and display the results in real time on Display_W. The detection results are indicated by green bounding boxes, while already marked targets are shown with red boxes. All bounding boxes are expanded by a certain number of pixels to better visualize the target's center. Using this button, the user can quickly and efficiently identify unmarked targets in the sequence by observing the continuously moving green boxes. Although the assistant detection model may have certain inaccuracies in detection and localization, some targets may be overlooked by the human eye, and it can assist the user in improving the completeness of target labeling. Considering that the detection process for the entire sequence may take a long time, the user can click this button again during execution to stop its function. When the button is green, it indicates that the detection model is processing the sequence images; when the button is white, it means the function is stopped.

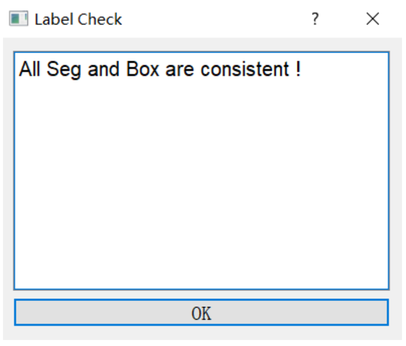
Run Blocking: This feature is yet to be implemented, and the specific plan has not been clearly defined. We are considering setting up a function here, where the software can stop at a specific frame when consecutive unmarked targets are detected, and prompt the user to mark them. This feature may be implemented in the future, and we are still in the process of exploring it.

Label Check: This button represents the label consistency check. Some researchers may require the bounding box of the target and its segmentation label to be consistent. After labeling the sequence images, the user can use this button to perform the check. If any inconsistency is found, the user will receive a warning message similar to Figure 4. If the labels are consistent, a message similar to Figure 5 will appear. Note that, in general, we define image coordinates starting from 0, which is common in platforms like PyCharm and Visual Studio. However, in MATLAB, image coordinates start from 1. Therefore, when the format is inconsistent, a warning will be prompted, as shown inFigure 4, where the green and red boxes differ by one pixel.


XH_IST_LabelingTool: F:/XH_New_InfraredSmall/IST_NS_2025/NS_0001
File
Setting
Info
Cite
Warning
replace box label with the bounding box of seg label
close

```
Seg and Box are not consistent !
```

Figure 4 Warning of the inconsistent labels. The red boxes denote bounding boxes of segmentation labels while the green ones are the boxes labeled directly by users.



All Seg and Box are consistent !

Figure 5 Consistency message.

Skip: This button allows the user to jump to any frame in the sequence. The user simply needs to enter the desired frame number in the input field on the left and click the Skip button as presented in Figure 6.

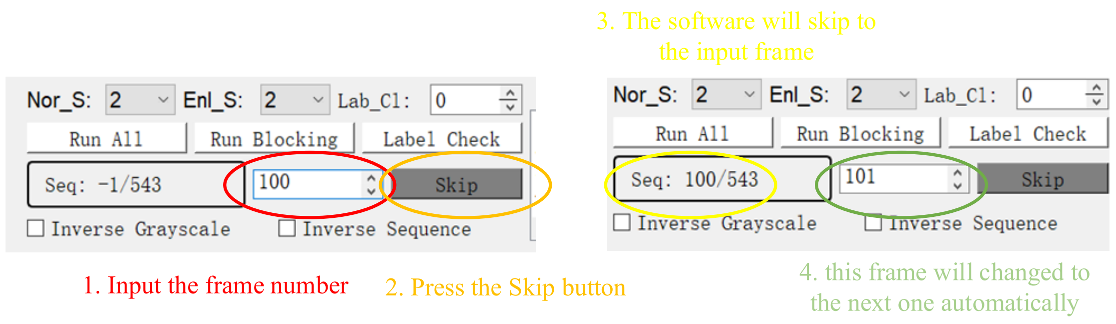


Figure 6 How to use Skip.

Inverse Grayscale: Although in typical infrared small target detection tasks, the target is usually brighter in localized areas (as is the case with our current assistant model), in scenarios such as drone-based detection over buildings or other background environments, or vehicle detection tasks on the ground, the target's brightness may be weaker than the background. Therefore, a function has been provided to invert the image's grayscale, allowing the assistant model to detect or automatically label targets based on the inverted image.

Inverse Sequence: Note that the software's target labeling strategy involves labeling the same target continuously across a sequence, rather than labeling all targets in one image and then moving to the next. After completing the target labeling for the current frame, the software will automatically move to the next frame. For example, if the current frame is 100, and the Check Box is not selected, after labeling the target, the software will automatically switch to frame 101. If the Check Box is selected, it will automatically move to frame 99. This function is essential because targets may change in brightness, becoming weaker or stronger. In most cases, human vision detects a target when it is sufficiently strong. However, with localized zoom and stretching, targets that are difficult to discern with the naked eye can still be marked continuously. With this feature, the user can start marking targets from any frame in the sequence (usually from the frame with a stronger target) and continue marking across the entire sequence.

2. Introduction and Settings

2.1 Settings in the MenuBar

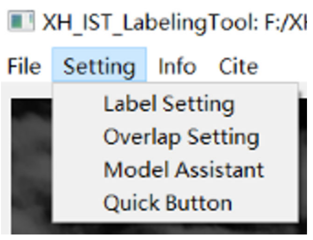
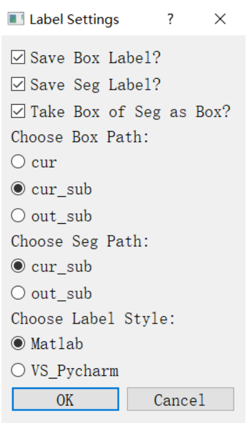


Figure 7 The four sub-settings in the menu bar.

As presented in Figure 7, there are four main settings in the menu bar, which are typically configured according to the user's preferences.

2.1.1 Label Setting

In this setting, the user will see the interface shown in Figure 8. Here, the user needs to configure the basic functionality for saving annotations. In the software, 'Segmentation' is typically abbreviated as 'Seg.' Either Box or Seg labels must be set to be saved. 'Take Box of Seg as Box' indicates whether to use the bounding box of the segmentation label as the target's bounding box label and save it. Since the bounding boxes are marked in Display_PN and the segmentation labels are marked in Display_S, the two types of labels may not align. When this checkbox is selected, the target's bounding box will be calculated based on the segmentation label. In this case, the local bounding box in Display_PN can be less precise, as long as the entire target is within the box.



Label Settings

[✓ Save Box Label?]
[✓ Save Seg Label?]
[✓ Take Box of Seg as Box?]
Choose Box Path:
cur
[✓ cur_sub]
out_sub
Choose Seg Path:
[✓ cur_sub]
out_sub
Choose Label Style:
[✓ Matlab]
VS_Pycharm

Figure 8 The interface of label setting.

Box Path and Seg Path represent the saving paths for the labels. After selecting a path through File -> Open Path (assuming the path is '*/seq01', which must be the directory containing the sequence images), the corresponding folders will be automatically created. If 'cur_sub' is selected, the bounding box labels will be saved in '*/seq01/box', and the segmentation labels will be saved in '*/seq01/seg'. If 'out_sub' is selected, the paths will be '*/box' and '*/seg', respectively, and the user can configure them based on their needs. For example, when constructing a dataset, if there is an 'image' subfolder containing the images ('*/seq01/images'), 'out_sub' may be a better choice. Otherwise, in most cases, 'cur_sub' should be selected. Additionally, since the bounding box annotations are stored in a TXT file, there will be no conflict with the original images, so the bounding box labels can be saved in the same directory as the original images by selecting 'cur'.

Choose Label Style' refers to the saving format for the bounding box annotations, specifically whether the coordinate system starts from 0 or 1. Selecting 'Matlab' will set the coordinates to start from 1, while the other option will use 0 as the starting point. These have been presented in the previous section.

2.1.2 Overlap Setting

Since it is common to accidentally mark the same target multiple times during the sequence labeling process, enabling the overlap detection shown in Figure 9 will trigger a warning interface, as shown in Figure 10 when label overlap occurs. The user can then choose to keep one of the labels or retain both.

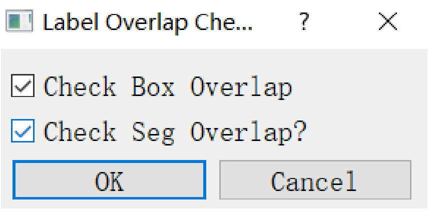


Figure 9 The interface of label overlap setting.

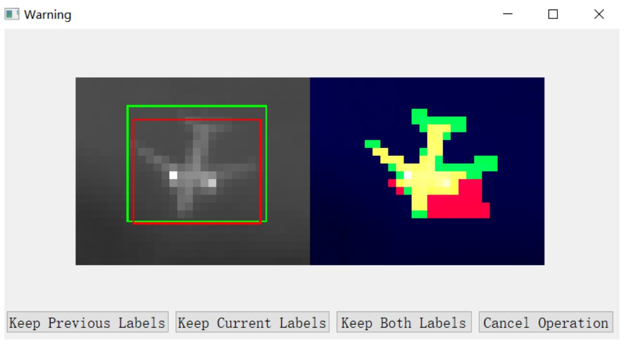


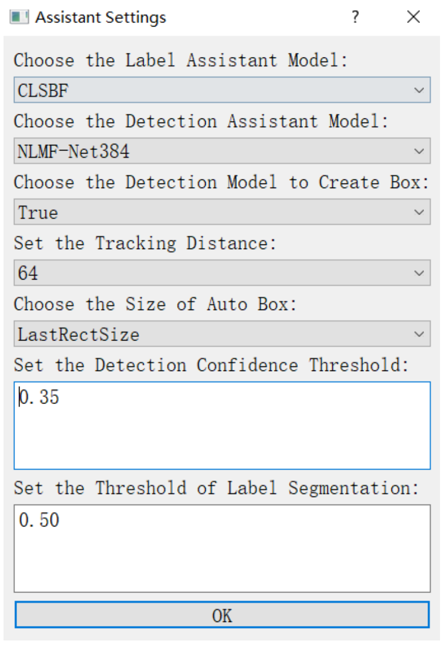
Figure 10 Interface of overlap warning.

In Figure 10, the green box represents the previously marked bounding box, while the red box indicates the currently marked bounding box. In the segmentation label on the right, green areas represent the pixels that are occupied by the previous annotation while the rea areas are only occupied by the current one. The yellow and white areas represent the overlapping portions between the two labels.

2.1.3 Model Assistant

The auxiliary model needs to be configured as presented in Figure 11, and the meanings of most of the functions are straightforward. 'Choose the Detection Model to Create Box' determines whether the detection model can assist in finding and localizing targets during the labeling process. 'Tracking Distance' specifies the range for local positioning when the previous function is enabled. 'Choose the Size of Auto Box' determines whether the preset size for automatic target localization relies on the detection model or the size from the last labeled frame. The default 'LastRectSize' option means that the box size will be directly based on the result from the previous frame's label, using only the center prediction capability of the assistant detection model.

The only current assistant model for automatic label generation is CLSBF, which generates target labels within a 32×32 local area based on the roughly accurate center points. This method works well for small targets (it can be understood as converting strong pixel-level supervision learning into point supervision). However, the segmentation process, similar to traditional adaptive segmentation, is influenced by the threshold value, which will inevitably affect the label generation results. Considering that a regression box will be drawn, the software directly erases the pixel-level labels outside the bounding box. In the future, we plan to incorporate technologies such as image completion to learn the background of the target area based on the bounding box, which may lead to more accurate labels (this can be understood as converting pixel-level strong supervision into approximate regression box supervision).


Assistant Settings

Choose the Label Assistant Model: CLSBF

Choose the Detection Assistant Model: NLMF-Net384

Choose the Detection Model to Create Box: True

Set the Tracking Distance: 64

Choose the Size of Auto Box: LastRectSize

Set the Detection Confidence Threshold: 0.35

Set the Threshold of Label Segmentation: 0.50

Figure 11 Interface of the model assistant.

The current assistant detection models include NLMF-Net[1] and YOLOX[2], based on results trained with over 160,000 data samples. If your task typically involves larger infrared small targets, YOLOX might be the better choice. For smaller targets, especially those not exceeding 6×6 in size, or with an average size below 4×4, NLMF-Net is recommended. If anyone believes their detection model performs better than the ones mentioned and is willing to share it for specific tasks, they are welcome to contact us and provide the corresponding ONNX and demo.py files. We will integrate your model into the software. Certainly, if you have a better demo for label generation assistance, we would also consider incorporating it into the software.

2.1.4 Quick Button

As presented in Figure 12, five keyboard shortcuts need to be configured here, with 'Continue Setting' currently having no functionality. The settings for the other four keys are mandatory and must not conflict with each other. The setup method is to click the corresponding button on the interface with the mouse (the button will be green), and then the user can press the desired key on the keyboard according to their preference. (The button will be white again and the caption under the corresponding button will show the text of the desired key.)

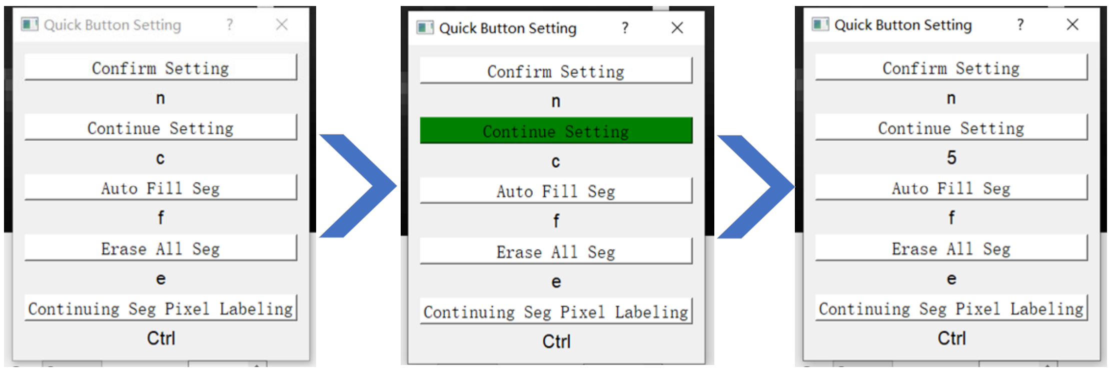


Figure 12 Settings of quick button

Confirm Setting means the setting of confirming the current label. Under the default settings, the shortcut key is set to 'n'. After the user has completed the labeling in Display_PN and Display_S, they can press the designated shortcut key ('n') to confirm the label is marked and automatically jump to the next frame. If the assistant detection model is enabled, the system will use the detection results to locate the target area in the vicinity.

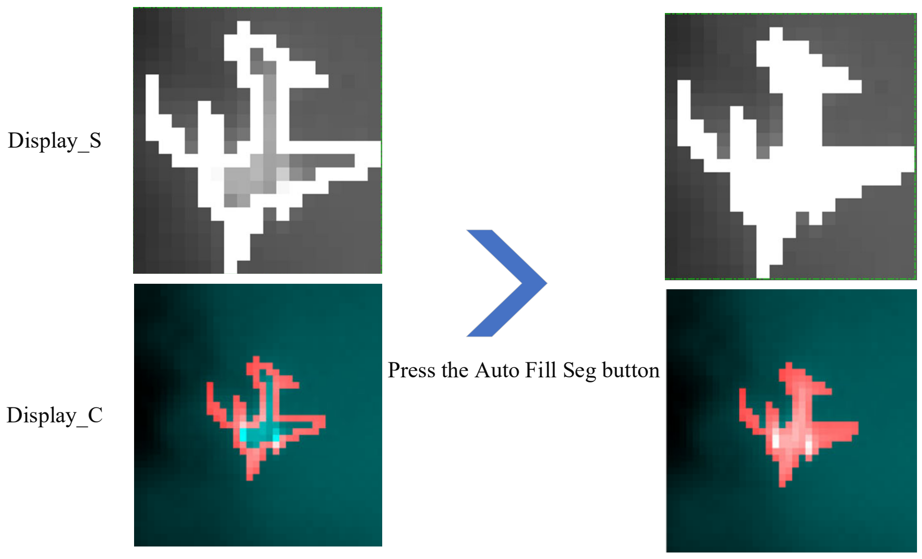


Figure 13 The setting of the Auto Fill Seg

Auto Fill Seg means the quick filling of unlabeled gaps in the segmentation labels in Display_S. This allows the users to focus more on labeling and refining the target's edge pixels. Once the edges are closed, pressing the designated shortcut key will automatically complete the filling. An extreme example is shown in Figure 13, where the segmentation label is manually marked along the target's edges manually. After pressing the "f" key under the default settings, all pixels within the enclosed central area are automatically filled.

Erase All Seg means erasing all segmentation pixels in Display_S. The automatic generation of segmentation labels may occasionally result in significant errors. In such cases, adjusting based on the auto-generated segmentation pseudo-labels may not be very useful. Therefore, this button allows for quickly deleting all pixel-level segmentation labels and completely labeling them manually.

Continuing Seg Pixel Labeling has been introduced earlier. Here, only the Ctrl, Alt, and Shift keys are allowed to be set, while other keys are not permitted. The user can continuously label the target's segmentation pixels in Display_S while keeping the key and mouse pressed.

Special Note: Setting up and adjusting the quick buttons is essential. Based on our past experience, during the process of continuous target labeling, the lifespan of certain keys on the keyboard may wear out. We have already damaged at least three keyboards during continuous labeling, with the "N" key becoming unresponsive after repeated use. This feature helps us maximize the conservation of economic resources.

2.2 Settings in the Main Window

The details of this section have been covered in Section 1.2.2 and will not be repeated here. It is just important to note that the software settings are distributed across the menu bar and the settings panel.

3. How to Label A Sequence (Quick Use)

In the previous two sections, we have provided a comprehensive overview of the software's basic features. This section will directly explain how to use the software quickly.

3.1 Open a path

The user must select and open a path using 'File -> Open Path' in the menu bar at first. Please note that the selected path must be the location where the sequence of images is stored, and all image names should have the same length and be ordered sequentially, such as '09.png', '10.png', and '11.png'. If the images are named '9.png', '10.png', and '11.png', the software will read the sequence in the wrong order. The software includes a consistency check for image name length. Additionally, this software only supports reading BMP, JPG, and PNG formats. Images in other formats will be filtered out and avoided. If a sequence is successfully imported, the interface will update as shown in Figure 14, with the area marked by the red box reflecting the corresponding image or information.

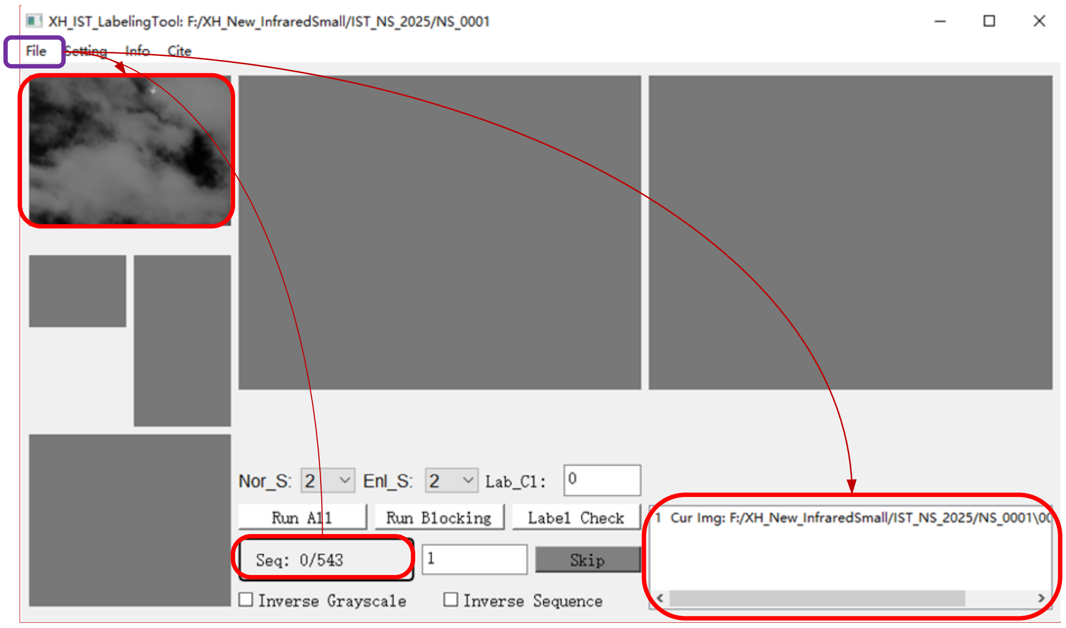


Figure 14　The interface of the software after successfully importing a sequence of images.

3.2　Draw the target's local area in Display_W

Roughly select the target's bounding area in Display_W, or alternatively, draw a roughly accurate bounding box around the target in the original image by mouse, as the green box in Display_W shown in Figure 15.

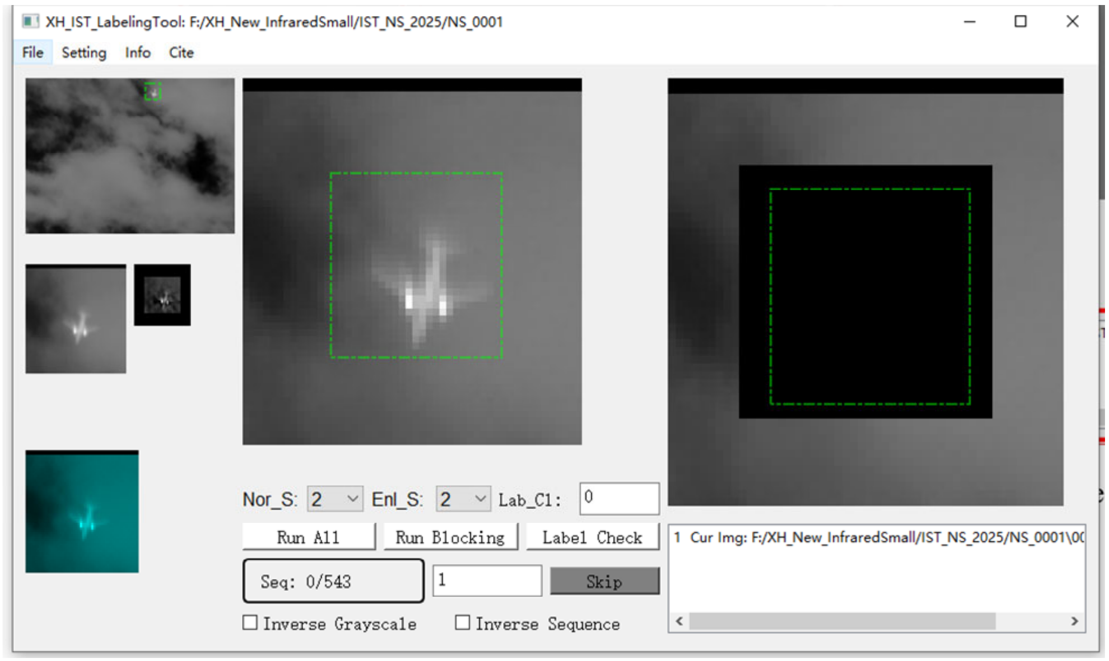
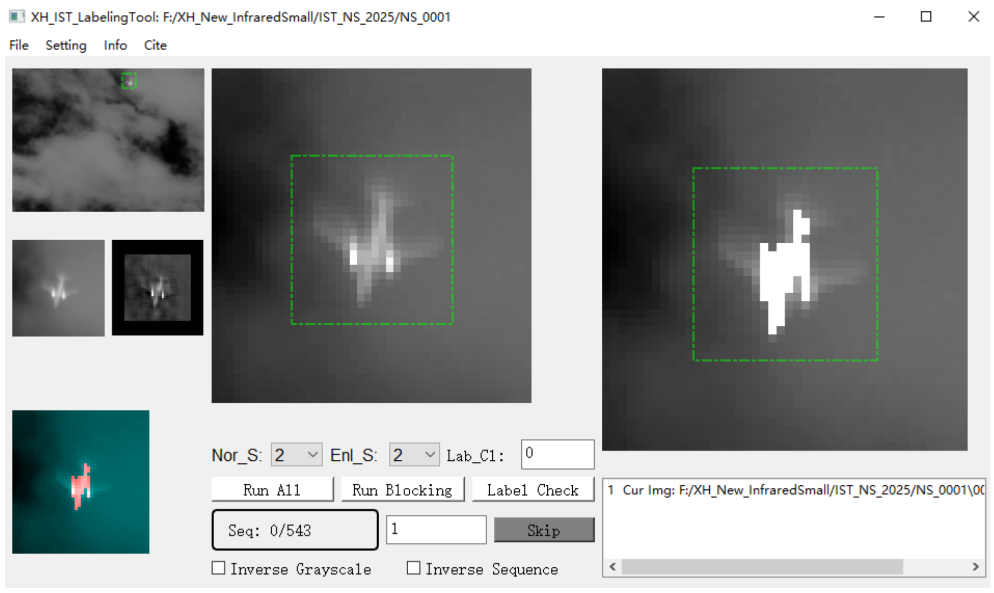


Figure 15　The update of drawing the roughly accurate bounding area.

3.3 Draw the accurate bounding box in Display_PN

After successfully drawing the approximate range of the target, all other displays will update to show the local area of the target in Figure 15. **The user can then further refine the target's bounding box in Display_PN.** If the user chooses to take the bounding box for the segmentation annotation later in Display_S, the box to draw here can be less accurate, but it must ensure that the entire target is enclosed within the drawn box as shown in Figure 16.



1 Cur Img: F:/XH_New_InfraredSmall/IST_NS_2025/NS_0001\0(

Figure 16. An example of drawing a roughly accurate box in Display_PN. The user can choose to draw an accurate box here, too.

3.4 Finish the accurate segmentation label in Display_S

Lastly, you can **directly perform accurate target segmentation labeling in Display_S. Press the designated confirm button (default is the N key) to complete the labeling of the target**, and the software will automatically switch to the next frame to continue labeling the target until all labels for the target in the sequence are completed.

4. Suggestion or Improvement

If you have any specific ideas for optimizing the software, feel free to contact us via email. As mentioned earlier, if you have better detection models or label generation assistant models, you are also welcome to provide us with the demo and ONNX models.

Email: xuhai_0513@foxmail.com

5. Reference

[1] H. Xu, S. Zhong, T. Zhang, X. Zou. Real-time Infrared Small Target Detection with Non-Local Spatial-Temporal Feature Fusion. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 2024:

[2] Z. Ge, S. Liu, F. Wang, Z. Li, J. Sun. Yolox: Exceeding yolo series in 2021. 2021, arXiv:2107.08430

6. 开发日志

6.1 2025 年 1 月 20 日星期一

基础功能基本完成,接下来要针对好用的需求,完善下部分用户友好型功能。

6.2 2025 年 1 月 21 日星期二

准备开始完成模型辅助检测，直接辅助定位到最近的目标的功能
还要注意修改的问题 ONNX 只能跑 384 的图像

辅助的时候还好，直接选择 384 的图像或者截取 384 的区域检测即可，但是检验 check 的时候可能会有问题，我是选择 resize 还是其他。

后期感觉还是生成两个版本 的 ONNX 吧，争取可以同时跑 384×288 和 640×512 的。

6.3 2025 年 1 月 25 日星期六

正式开始重新整理这部分标记软件，开始回归辅助框生成的部分工作。

暂时就定义了图像的大小只能有两种选择，分别是 384 和 640，如果超出了这个部分，则默认仅选择中心区域。

本来准备配置 detectioncfg.yaml 文件的，但发现部分模型是分两个 ONNX 模型的,这样会导致存在一定问题,而且每个方法内部肯定调用的函数也存在差别,所以这个意义真的不大。因此，决定还是写死固定在代码中。

如果发现模型的输入大小与图像原有的设定大小不匹配的时候，选择 debug_event 显示出来。

开始重点转向模型辅助

6.4 2025 年 1 月 27 日星期一

重新开始干活

为了避免 ModelAssist 类中重复导入 ONNX 模型，增加内存的开销，即没有必要为了辅助检测和辅助生成回归框导入两次模型，将 AutoAroundBox 改为设置 True 或者 False 两种配置，这样，辅助漏标记检测和辅助生成回归框的

基本功能嵌入成功，还需要检查更复杂的逻辑 bug。

另外，检测结果错误的情况会比较场景，目标的外接框大小，尽量要复制上一次的标记框，

晚上截止前，发现存在生成的框经过校正后仍然大小不符合预期的情况，目测是存在逻辑错误或者其他错误的问题。

6.5 2025 年 1 月 28 日星期二

发现传递的位置存偏移，这说明传递的位置绝对位置在 ImageWithRect 里是无法默认处理相对坐标的。自动辅助检测的框和位置基本正确了，

Rect2 和 rect3 都存在无法连续画黄色框的情况，rect1 好像不受影响。根据现在的设定，画定的框是绿色显示，而正在画的框要用黄色显示，因此，原则上，是 move 事件或画图逻辑出现了问题。

经过调试，发现是 rect 判断重复的问题，要么是运动更新不及时，要么是有其他情况的意外更新。

基本排查并解决黄色框不绘制的问题，兼顾考虑注释掉和优化彻底删除的代码，当前代码行数为 3554 行，这一周多仅增加 500 行左右的总量，投入时间不足。目前代码基本功能正常，继续完善边角功能，优化使用速度。

在 384 模型下，测试一个 640 的图像序列，发现问题：

（1） 自动辅助标记的坐标，统一出现了 0.5 个像素的偏差，可能是坐标计算的时候，没有注意取整换算；

（2） 不能直接去修改 rect3 中的像素级标记，必须等到第二个框手动调整一次后，才能在第三个框中去修改，这可能是坐标的更新错误导致的；

（3） Skip 功能中，跳转下一页的功能应该要自动生成对应数字，方便操作，且跳转下一张图的时候，局部大小的位置应该保持不变，而不应该让图 2 和图 3 直接变成大图显示，应该继续维持小兔显示；

（4） 应该增加更多的局部范围，因为可能会手动标记大量静态的目标，此时，就没有必要去纠结 32 个像素以上的情况，目标就应该在 16 个像素以内；

（5） 640 的模型测试也存在错误，坐标的计算和填 0 操作都是有问题的，不过 384 模型在 640 图上截取小图疑似没有错误。

（6） 应该增设分割阈值的大小，

（7） 重复警报时，发现绘制的框太大了，要换成画细框的模式，不能直接先画框，要缩放后再画框，否则根本看不清楚。局部显示的外接框的位置，可

能也不是完全正确，需要调整。

（8） 要定义一个最小的显示区间大小，比如 16×16，让拉伸、显示的区间范围不要太小。

（9） 发现 NLMF640 模型存在运行错误，疑似是输入等问题，因此，需要重点投入调试解决，如果解决失败，就先维持使用 384 模型。

（10）第一张图没有成功辅助检测，按理来说，这个目标应该是没有问题的才对，因此，肯定是逻辑处理失败，打开路径或新开图片时，没有调用辅助检测功能。

6.6 2025 年 1 月 29 日星期三

开始解决昨日发现的问题，第一个问题是坐标的取整问题。辅助检测框取整的问题基本解决。

第二个问题和第一个问题是相关连的问题，因此，可以跳过。

Skip 和局部显示功能少量代码即可补充相关功能，且不与主要逻辑有交互关联，容易解决。

关于不同尺寸的图像运行指定模型就会是巨大投入了，这里要详细检查输入图像的逻辑，以及外部代码是否可以独立调试 640ONNX 等等，需要修改的输入口是 run_all_img 函数，重点是更换 assist_detect_process。

此时不能直接 copy 辅助模块里的 run_assist_detection_check，因为这个模块不输出置信度值，且仅仅保留回归框的输出，目前发现是只有 NLMF-Net 存在错误，其余代码不存在，因此，有必要独立调试 NLMF640 的 ONNX 模型。

发现模型的 forward 函数里有一个预设的变量，好像是没啥大用。已经解决该问题。

综合考虑，部分问题可以由其他功能代替。还剩下重复标签时，线过粗的问题。

重复标签的提示显示的线条也换成了细线条，这样提示时，显示的也较为干净，且不影响多个框的显示

昨天发现的问题基本解决，还剩下的问题包括导入新序列后，第一张图未辅助检测的问题，这应该是逻辑上出现了没有调用，或者单纯是显示出现了问题。该问题暂未解决。

截至今天结束，所有代码注释等累计达到 3641 行，较上次多 90 行左右。

6.7 2025 年 1 月 30 日星期四

开始重点解决打开路径时，辅助检测调用和局部检测显示的问题。辅助检测的功能函数是 run_assist_detection_check，返回一个唯一的回归框，这里无非是要检查，并在打开图像的时候，跳转图像的时候，要设立合适的规则，调用 run_assist_detection_check。

关于导入新路径的时候，尽可能复用 index_advance_seq 函数，这样的话，需要提前设定当前帧序号，这样最简便的方式之一是新建一个函数，根据序列方向导入初始编号，然后复用 index_advance_seq 函数。
基本完成。

到这里，打开路径调用模型辅助生成回归框的基本功能算是实现了，接下来重点是检查打开路径时，基本信息的清楚。
注意，要清楚上一次路径的所有信息，包括 rect 里的所有回归框等信息，因此，main 函数应该增加一个功能，去除掉对应的信息。

6.8 2025 年 1 月 31 日星期五

昨天几乎没有投入工作，今天主要完成导入路径时，框体的基本重置问题。打开新路径时，上一次回归框的信息基本删除，且确定调用了模型的检测框功能，如果没有调用，就说明没有成功检测到目标。
调整了 debug_list 的显示，确保永远显示最后一条，并保留 30 条的历史信息。

接下来调整跳转下一页时，也要实现调用功能。目前，下一张图片的调用函数在 main 中的 update_img_index 实现，这里如果项继续复用 index_advance_seq 函数可能会有些麻烦，因为左键和邮件与获取方向没有关系，可能要修改部分函数，这个和 Skip 时的情况比较接近，相当于要具有跳转到指定编号图片，并调用 index_advance_seq 中大部分功能。
第一步拆解了 index_advance_seq 函数，独立出一个 update_specific_index。

修正规划的逻辑错误，read_img_index 是完全独立的最底层读取图片的函数，main 中的 update_img_index 函数调用了 read_img_index，实际上是应该新建出更复杂的函数，同时在 imgManager 中调用 read_img_index 和 update_specific_index。

初步测试似乎没有问题，接下来抽时间标记 5 个左右的序列，如果没有问题，则预计基本功能 OK，后续需要完善的功能以后再说。

代码及注释总理达到 3715 行。

6.9　2025 年 2 月 1 日星期六

先在模块中判断需要的模型是否存在，这部分暂时还是写死在代码中。

发现了奇葩问题，辅助模型 CLSBF 会莫名奇怪地变为了 None，检查发现了问题原因，已修正。

6.10　2025 年 2 月 2 日星期日

这两天基本未实现任何功能，主要目的是测试下标记软件，重新发现了一些问题，并需要在未来两三天时间内重新完成。

这两天根据自身情况，累计有效标记了 500 个左右的目标，发现了大约 4-5 个问题等待优化，其中，提示连通域和检查标签一致性的问题，应该在开源软件前就得完善，其余功能，可能根据以后特定情况，要不要完善。

另外该序列还有至少 2 个连续目标没有标记，可以等待后续功能逐步完善后，再逐一完善后，再决定是否执行。

另外，如果不想太方便外人，可以设置 return 来回避某些功能，这样软件上就可以直接继续优化。

6.11　2025 年 2 月 3 日星期一

今天开始，重点解决可能造成错误标记的问题，首先是连通域问题，因为定义了只接受 4 连通域，所以，出现多个连通域时，必须要提醒修改。

一般情况下，出现标记多个连通域是因为只做到了单一目标的 8 连通域，导致存在两个 4 连通域，因此，正常是不会出现多个连通域的，因此，直接 RGB 循环填充连通域即可。

这里的警报只要能显示图像即可，因此，不需要取设置和执行回调功能，提示后，直接关闭窗口回到主界面即可。

该功能实现。

接下来需要重点完善的一个功能是检查标签，这不会是一个非常常用的一个功能，感觉没有必要占据界面。现在先提前规划下哪些应该长时间占据界面：

标签的局部显示大小和类别，以及是否选择反向标记应该是要暂时保留，跳转也可能是常用功能。但是，检查运行等基本都属于标记完成后再长时的功能，因此，这两个按钮可以往后移动。

新建一个 Menu 标签栏吧，定义为 Check，即可以全局观察检查标签与辅助模型的统一性，可以阻断式观察，可以检查分割标签和回归框标签的统一性，将界

面的显示区域尽量还给标签。

不行，下午刚刚思考的方案其实不具有应用能力。因为，跑一个序列是要占据时间的，这些需要阻断时长的功能，还是应该占据界面可以实时停止最为合适。
界面初步调整，留出了更多的空间显示图像。通过在 1080P 界面上放大，可以在中间一栏再增加一个按钮。

代码及注释总理达到 3956 行。

6.12 2025 年 2 月 4 日星期二

今天要优先解决掉标签重复性检查的问题（昨天完成了部分接口）
这里本来是准备重写函数判断两个标签的一致性的，现在改变思路复用 read_img_index，这样可以直接基于 self.cur_seg 和 self.cur_box_list 来判断二者的统一性，即直接求 self.cur_seg 的连通域，并和 self.cur_box_list 进行一对一完全匹配验证，只要不一致，就可以提示并报错。

这部分的 img_manager 的主函数写完了，明天有空的话介入 main 并调试逻辑即可。

目前代码总量为 4018 行，今天未完成任何一个主要功能。明天要返回租房地。

6.13 2025 年 2 月 5 日星期三

经过整理，删除，代码量调整为 4030 行。目前基本确定，代码暂时封存。

6.14 2025 年 2 月 6 日星期四

首次回到实验室，今天预计可以工作 3 个小时，还得继续花时间整理租房的布局。

今天的第一工作是彻底解决标签的一致性匹配。

发现了一个糟糕的问题，不能在自行建立的线程里操作 UI，需要建立继承线程的 worker，建立一种通讯规则，在 main 的主函数里才能操作 UI。
目前，经过删减，代码下降到 5012 行。

不得不强行修改了，现在把 check_label_block 看成一个普通的函数，因为标签的匹配性检查是很快的。

另外现在逻辑显示上可能有一定错误，如果没有匹配上外接框标签，可以直

接选择拿分割标签作为补充标签，不用自己重新标记，不过现在不了，这个作为以后有空的优化点。

到此，原则上如果测试几个标记序列没有问题，原则上可以开源。

Run all 也存在线程调用界面 UI 的问题，必须要优化

成功将原来的线程，改为了 QThread 的信号与槽的方式，目前来说，报错和闪退问题已经大幅度减少。

6.15 2025 年 2 月 8 日星期六

工作转移到论文的审稿回复中，标记软件的代码部分工作暂定。

7. 存在的问题整理

现在开始将所有发现的问题或待解决的功能按顺序整理在这里，不再在日期里记录对应的功能缺陷等问题。

（1）发现界面放大条件下，在切换功能时存在明显的卡顿，尤其是界面放大到整个界面的情况下，卡顿的具体原因是什么有待进一步查看。
（2）后续可以考虑在各个界面增加不同颜色框的说明
（3）调换图片时，存在框的更新、局部大小的显示、以及辅助检测功能缺失的问题，影响实际使用，这个是非常重大的一个问题。
（4）~~打开路径时，应该清除所有已有框的基本信息~~。
~~（5）跳转页面的时候，下一页的页码也应该能够自动跳转，其实，大多数情况下，跳转下一页的功能不应该是独立的执行跳转功能，而应该是根据方向性，执行 index_advance_seq 更合适，可以增加以下判断，如果满足条件的，就复用 index_advance_seq，否则才去跳跃读取数据。~~
~~（6）辅助模型的导入时，文字项的添加前应该判断该模型是否导入成功~~
（7）应该增加累计标记时长的显示功能。
~~（8）要增加连通域显示的问题，现在定义标签必须是四连通域，但如果发现了两个及以上的连通域，应该及时提醒，避免出错~~
（9）应该考虑让下一帧自动框的大小稍大于上一帧实际外接框，且可以灵活设定
（10）要排版下首页的一些功能，有些功能没有必要占据一整行，必要性整理并列
（11）增加 Ctrl 简易的回归框取并功能，即在 ctrl 按下的条件下，exist_rect 对两给框取并，自动计算一个较大的框。一般是不需要减框功能的，同时，要注意取并的时候；
（12）增加外接框标签和分割标签的统一性判断，如果发现异常，自动跳转到指

定图像并

（13）让 img_manager 部分提前计算或者获得显示框的等比例放大系数，并提前将已经标记的目标外接框标记在第二个显示框上，这样有助于后续标记密集目标出现

（14）爆出了“RuntimeWarning: Mean of empty slice.”，需要全面检查一些库的应用，判断变量是否为 None

（15）关于标签一致性检测以防止标记错误的问题，需要注意，标签过程中，设置了允许标签不完全一致的功能，因此，如果要执行该标签功能，需要一个功能界面，设置具体的标签一致性判据，比如完全一致性，IoU 比例等。中心距离可能不适合在这里使用，如 IoU 设定 100%就算完全一致才通过，其余按比例来，即使目标较小，也可以快速过滤（待实现的重要功能）

（16）语言功能待完善，至少实现中英文的语言切换，当然，也只会

（17）出现不匹配的外接框标签时，可以选择直接拿分割标签的结果作为补充的标签，这个功能以后可以优化。

（18）可能存在要删除标签的问题，比如某个标签可能是错误标记的，因此，需要考虑执行该功能。

（19）应该直接设定一个清除指定标签的基本功能，比如直接清除整张图像的标签，清除特定标签等等。这就要调出一个功能，能够转移到标签所在的局部范围。这个功能其实是比较重要的。

（20）待发现的问题

（21）待发现的问题

（22）待发现的问题

测试功能的时候还是尽量用 debug 模式，起码报错的时候会直接定位到代码。

8. 给自己的提醒

（1） 有些地方看着不重要，但设计到了多界面的交互和隐含的线程问题，还是强行加锁比较安全，毕竟现在发现了一些意外情况下的闪退；

（2） 还有语言设置，语言设置要完成，就保留中文和英文两种。

打包代码(重命名打包似乎失败了)

```
pyinstaller --onefile --icon=app_icon.ico --name=XH_CLSBF_Label main.py
pyinstaller --onefile --icon=app_icon.ico main.py
```

如果提示递归深度的问题，再 spec 文件中添加，在顶部添加 ，即在#号的下一行回车添加：

```
import sys
sys.setrecursionlimit(sys.getrecursionlimit() * 5)
```

执行：

```
pyinstaller -F main.spec
pyinstaller -F XH_CLSBF_Label.spec
pyinstaller XH_CLSBF_Label.spec
```

如果出错，就去掉-F 指令，会有 console 界面

也可以去 spec 里去关闭 console，改为 False

UI 界面要避免过于高频地修改显示，比如一些字符串的显示，感觉频率高了软件很容易就挂掉了。