WaferDetector

Wafer Sawing Line Detector project for DIP 2023, Guangming Lu, HITSZ.

1. QuickStart

```
pip install numpy, opencv-python, matplotlib, scipy
```

1.1 Running Naive Detector

```
cd detectors && python detect.py
```

- The preprocessed results will be saved in the `detectors/pre-output` folder.
- The mid results will be saved in the `detectors/mid-output` folder.
- The final results will be saved in the `detectors/final-output` folder.

You can also run ND with notebook at `notebooks/naive_detector.ipynb`. Just click the `Run All` button and you will get the results.

1.2 Running Line Through Detector

```
cd detectors && python detectPB.py
```

Note that PB here denotes point-based.

- The mid results will be saved in the `detectors/pb-mid-output` folder.
- The final results will be saved in the `detectors/pb-final-output` folder.

You can cancel the showing image by assigning `showResult = False` in the `detectPB.py` file, line 107.

1.3 Comparing with Golden Results

```
cd detectors && python measurement.py
```

1.4 Performance Evaluation

```
cd detectors

echo "Naive Detector"
time python detect.py

echo "Line Through Detector"
time python detectPB.py
```

The *real* field in the output indicates the real time of processing the images.

2. Introduction

2.1 Naive Detector (*ND*)

The key idea of Naive Detector is to extract the longest line in the image, and then use the line to extract the wafer sawing lines. The challenge here is that the line tends to be noisy and discontinuous; thus, we need a way to connect these lines. To address the challenges, we introduce a three-step-processing algorithm, as described below.

1. Preprocessing - Used for extract the sedges from the background in a coarse granularity.
    - Gaussian Blur
    - Sobel Edge Detection
    - Morphological Transformations for Edge Enhancement
    - Segment (Line) Detection
    - Select top K lines with an given error threshold (i.e., 100)
    - Draw the lines on the original image
2. Processing - Used for extract the sedges in a fine-tuned granularity.
    - Gaussian Blur
    - Sobel Edge Detection with a smaller threshold to obtain the coarse wafer boundary
    - Segment (Line) Detection
    - Select top K lines with a smaller error threshold (i.e., 10)
    - Group the selected lines into two groups, one for the top and one for the bottom
    - Choose the top and bottom lines with the smallest center distance
    - Adjust two selected lines to be parallel
    - Calculate the sawing line and its angle, and we're good
3. Postprocessing - Used for detect if there are any other wafer sawing lines.
    - Fill the sawing path with white-black segment
    - Re-run Processing to try to find the second wafer sawing line
    - Compare the new wafer sawing line with the previous one, determine if they are the ``same''
    - If they are the same, then we're good
    - Otherwise, we obtain the new wafer sawing line, and try to find the third one until we cannot find any more possible new wafer sawing lines.

Naive Detector is able to handle the case where two sawing paths are crossed. However, it is fragile when there are multiple `horizontal'' and` vertical'' sawing paths in the image since Naive Detector is primarily designed to detect the longest line in the image. We further design a detection algorithm as below.

2.2 Point-based Line Through Detector (*LTD*)

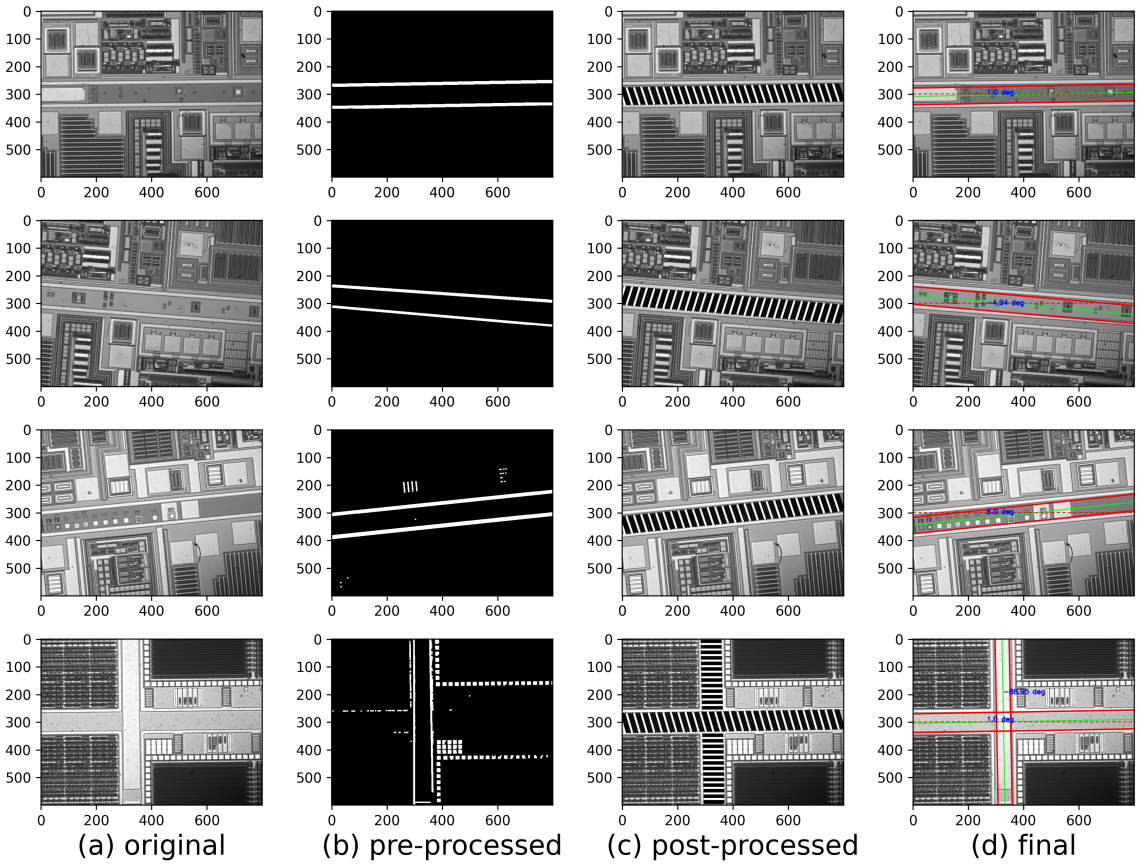
The key idea of Point-based LineThrough Detector is to carefully determine the wafer sawing line based on the border points, enabling the more precise and robust detection of wafers sawing line. The challenges here are that the border points are noisy. To address the challenges, we introduce a clean-area based algorithm, as described below.

1. DetectEdge - Used for extracting the edges from the image.
    - Sobel Edge Detection
2. DetectCuttingRegion - Used for extracting the clean area in the image.
    - Extract connected domain from image based on edge image in 1.
    - Remove connected domain with degree Less than 2
    - Use dilation and erosion operations to remove empty holes in the retained connected domain
3. DetectPoint - Used for detecting the points which is in the boundaries of the image.
    - Use ring mask to get the points which is in the boundaries of the image
    - Use closing operation to connect connected domain with gaps less than threshold
    - Use dilation and erosion operation to get the endpoints of the connected domain
    - Consider the center coordinate of connected domain as the coordinate of edge points
4. DetectLineThrough - Used for detecting the valid through-lines based on the edge points.
    - Combine two edge points to form a through-line
    - Check the validity of through-lines
5. FilterOutLine - Used for further filtering out invalid through-lines.
    - Use Hough Transformation to get the line segments in the image
    - For every through-line, check if it passes through a line segment
    - Remove through-line that does not pass through any line segment
6. CheckCuttingLine - Used for getting all possible Combinations.
    - Combine two through-lines to make a sawing path and check the compatibility of this sawing path
    - Combine some compatible sawing path to form a Combination
7. GetMinimumAreaCombination - Used for obtaining the relatively good Combination with the minimum area.
    - Sort the Combinations with their error
    - Get the Combination with minimum error
    - Get the number of sawing paths contained in the best Combination
    - Get the Combination with the minimum area
8. Postprocess - Used for converting Combination into information of sawing line.
    - Get the center line(s)(sawing line(s)) of the sawing path(s) in the Combination

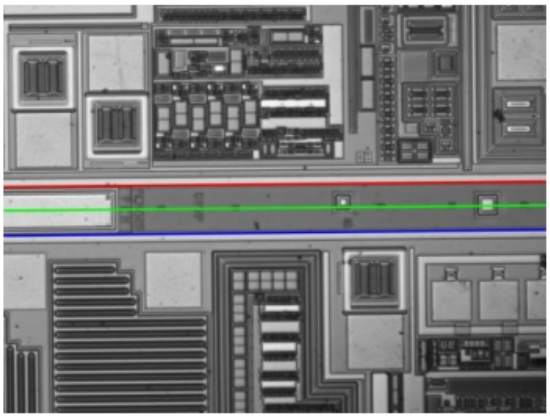
- Calculate the angle between the sawing line and the horizontal vector

2.3 Experimental Results

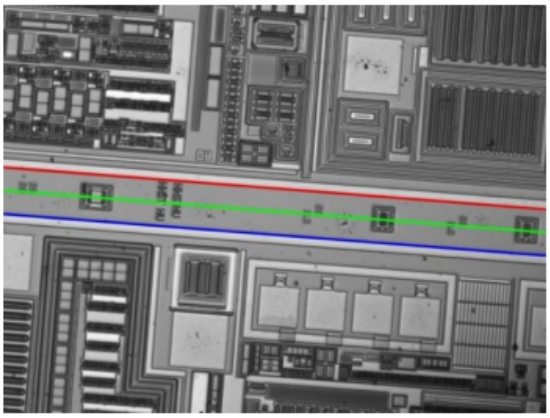
1. ND Result



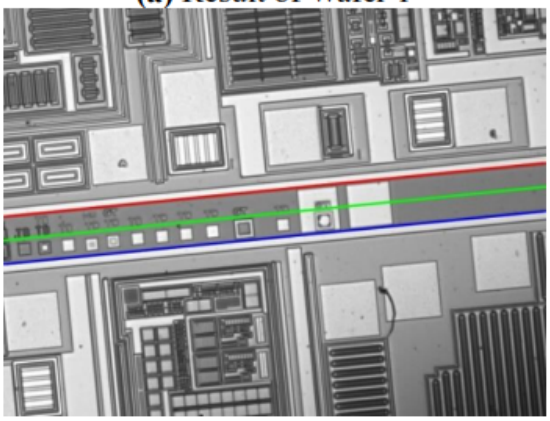
2. LTD Result (Optimal)



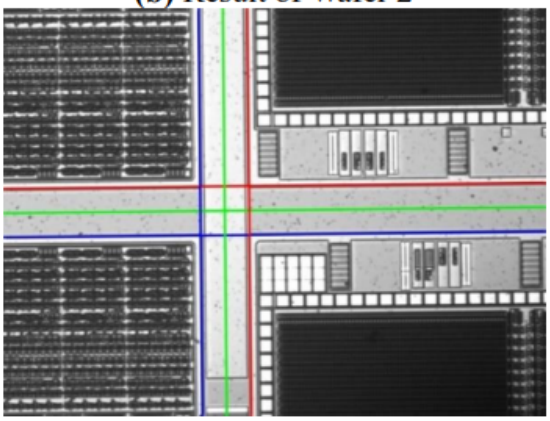
(a) Result of Wafer 1



(b) Result of Wafer 2

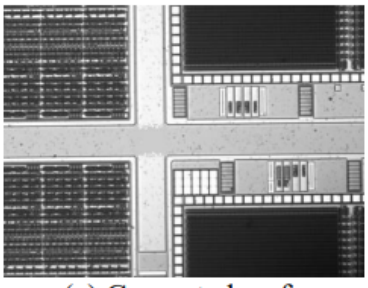


(c) Result of Wafer 3

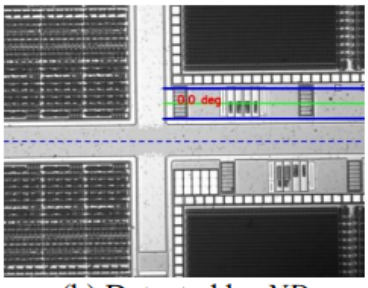


(d) Result of Wafer 4

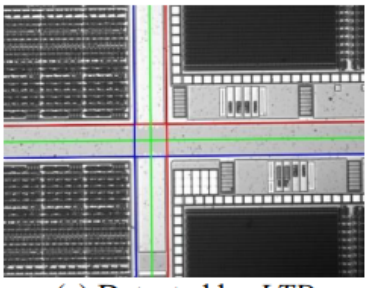
3. Robustness



(a) Corrupted wafer



(b) Detected by *ND*



(c) Detected by *LTD*

4. Performance

- **Testbed**. We run the experiments on a Windows 10 machine with an Intel i7-11800H CPU clocked at 2.30GHz and 16GB memory.
- **Result**. We study the performance of ND and LTD in this section. Specifically, we use time command to measure the real time of processing Wafer 1-4 and the corrupted Wafer 4. ND spends around **3.459s** while LTD spends more than **30s** since LTD performs complex verifications to ensure the correction of detected sedges.

3. Details of LTD

3.1 Hyper Parameters of LTD

Name	Description
edgeThreshold	In the gradient map obtained through Sobel operation, points with gradients greater than this threshold are considered edge points.
CuttingRegionMerge	For each connected domain with a degree greater than 2, the parameter defines the number of dilation operations and the number of subsequent erosion operations performed. The purpose of these two operations is to eliminate internal empty holes.
intervalThreshold	When extracting the outermost point of the image, if the interval between the two connected domains is less than this threshold, they are merged into one connected domain.
lineThreshold	When extracting the outermost point of an image, if the "length" of a connected domain is greater than this threshold, only the endpoint of the connected domain will be preserved, rather than using the center point of the connected domain as the result point.
FOLHLPThreshold	When filtering through lines, this parameter defines the parameter in the Hough Transform (cv2. HoughLinesP).
FOLHLPMinLineLength	When filtering through lines, this parameter defines the parameter in the Hough Transform (cv2. HoughLinesP).
FOLHLPMaxLineGap	When filtering through lines, this parameter defines the parameter in the Hough Transform (cv2. HoughLinesP).
FOLAngleThreshold	When the angle between the through line and the actual line segment in the graph is less than or equal to this threshold, the distance test between this through line and this line segment will be conducted. Otherwise, it is directly determined that the through line does not "pass" through this segment.
FOLInterPThreshold	This algorithm provides multiple options for determining whether a through line passes through a certain segment. This parameter defines that if the overlap degree of BBox between the above two is greater than this threshold, it is determined that this through line passes through this segment. However, this solution has certain issues and is no longer in use.

Name	Description
FOLLineMaxDistance	When determining whether a through line passes through a segment, if the distance between the two lines is less than this parameter, it is determined that the through line passes through the segment. The definition of distance here is: if there is an intersection point within a line segment, the distance is 0; otherwise, the distance is the distance from the point closest to the through line in the line segment to the through line; If the two are parallel, it is the distance between the two lines.
GroupLineAngleStep	The relevant process has not been used.
GroupLineStepScale	The relevant process has not been used.
CCLLineDistanceMin	The minimum distance between the two cutting lines that make up the cutting path.
CCLLineDistanceMax	The maximum distance between the two cutting lines that make up the cutting path.
CCLLinePairInterAngleMax	If the angle between the two cutting lines that make up the cutting path is greater than this threshold, the cutting path they make up will not be considered.
CCLLinePairOuterAngleMax	When constructing a Combination, if the angle between the two cutting paths is less than this threshold, it is determined that the two cutting channels are compatible. Otherwise, further judgment is performed.
CCLLinePairDistanceMin	When constructing a Combination, if two cutting paths are determined to be incompatible during the angle test, and the distance between them is less than this parameter, they will be considered incompatible, otherwise they will be determined to be compatible. That is, the minimum distance between nearly parallel cutting paths.
CCLCuttingRegion-CheckMaxSuccession	When detecting the rationality of the Combination, the "maximum length" of the black area during the clean area matching process.
CCLCuttingRegionOffset	When detecting the rationality of the Combination, the dilation operation starts after offsetting a few pixels outward to obtain the outer contour of the Combination (later used for matching with the "clean area").
CCLCuttingRegionWidth	The assumed minimum "width" of the "clean area" when detecting the rationality of the Combination.

Name	Description
CCLEdgeOffset	When detecting the rationality of a Combination, the edges of the Combination are shifted inward by a few pixels and then matched with the previously obtained edge image.

3.2 Json file entry description of the output of LTD

Name	Description
cutting_path_num	How many cutting paths are there in total.
cutting_path_list	Cutting Path Information List
cutting_line_through	Two point coordinates of the cutting line (not limited within the image).
cutting_line	Two point coordinates of the cutting line (limited within the image).
angle_of_cutting_line	The angle between the cutting line and the horizontal line in the clockwise direction (0-180).
cutting_path_up	Upper edge of cutting path.
cutting_path_bottom	Lower edge of cutting path.