C# 기반
OpenCV4를 사용한
객체 정렬 시스템

개인프로젝트

2022.04.04 - 2022.04.09

김혜진
010-7238-0904
a5341663@naver.com
https://github.com/MEAJIN/OpenCV

CONTENT

객체 정렬 시스템 프로젝트 목차

C# 기반 OpenCV4를 사용한

객체 정렬 시스템

#정렬 #aligner

입사검토 메일을 받고 내용을 확인하던 중 자동화기기 OLED Aligner SW라는 문구가 눈에 띄었습니다.

OLED Aligner가 어떤 공정인지 궁금하여 검색을 해보니, 회로 패턴이 담긴 Mask와 회로 패턴을 찍어내기 위해 필요한 Wafer를 정렬하는 공정이었습니다.

만약 학부생 시절에 OLED Aligner 공정을 알고 있었더라면 "공장자동화 시스템"에 접목 시켜볼 수 있지 않았을까?" 하는 **아쉬움**이 들었습니다.

비록 이 공정을 공장자동화 시스템에 접목 시키진 못 했지만 OLED Aligner라는 공정을 접하게 된 이번 기회를 발판으로,

OLED Aligner 공정과 비슷한 작업을 하는 프로그램을 만들어 보는 것도 좋을 것 같다는 생각이 들었습니다.

만들게 된다면 아쉬움도 털고 새로운 분야에 접해보는 좋은 기회이자 경험이 될 것 같았기 때문입니다.

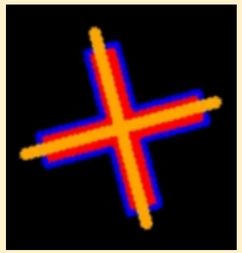
이러한 생각을 바탕으로 Aligner S/W를 개발하게 되었고, 지금의 '객체 정렬 시스템'이 만들어진 이유입니다.

C# 기반 OpenCV4를 사용한

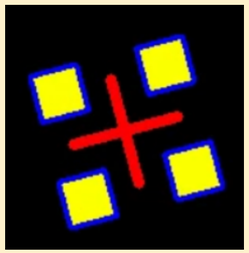
객체 정렬 시스템

#프로젝트 소개 및 설명

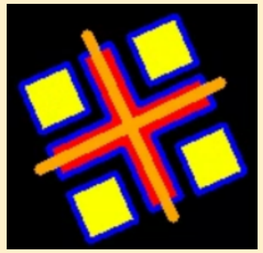
소개



[Cross]
: Mask



[fourBox]
: Wafer

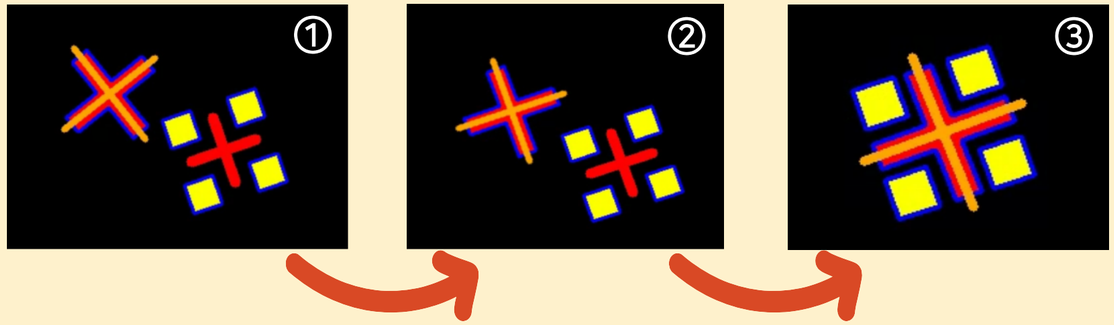


[Align 완료]

Mask와 Wafer의 좌표 값과 각도 값을 구하고,
이 둘의 차이 값을 구한다.

각도 차이 만큼 회전시키고,
좌표 차이 만큼 이동시켜 정렬한다.

설명



① Mask와 Wafer의 각도 값의 차이 만큼 회전

② Mask와 Wafer의 좌표 값의 차이 만큼 이동

③ 정렬(Align) 완료

C# 기반 OpenCV4를 사용한

객체 정렬 시스템

#주요 기능 #도형 회전 ① #Spin

Spin 함수 호출

```
for (int i = 0; i <= (int)(axisDegree + 0.5); i++)
        {
            Spin(crossBoard, crossBoardTemp, randomValueXY_C, direct * i); // 1
            Cv2.Add(fourBoxBoard, crossBoardTemp, add); // 4
            Cv2.ImShow("FPaint", add); // 5
        }
```

Spin 함수

```
    static void Spin(Mat board, Mat dst, Point randomValueXY, int randomValueAngle)
    {
        Mat matrix = Cv2.GetRotationMatrix2D(new Point2f(randomValueXY.X + 50, randomValueXY.Y + 50), randomValueAngle, 1.0); // 2
        Cv2.WarpAffine(board, dst, matrix, new Size(board.Width, board.Height)); // 3
    }
```

C# 기반 OpenCV4를 사용한

객체 정렬 시스템

#주요 기능 #도형 회전 ① #Spin #설명

1 :
- 인자 설명: Spin(cross 도형을 그릴 Board, crossBoard에서 생성된 도형을 복사하고 이를 축 정렬에 사용 , 랜덤 좌표값, 회전 방향)

2 :
- GetRotationMatrix2D 함수를 이용해 도형의 중앙을 기준점으로 회전 한다.
- 함수 설명 : GetRotationMatrix2D(회전 중심 좌표(X, Y), 회전 각도(degree), 회전 방향(음수는 시계방향))

3 :
- WarpAffine 함수를 이용해 도형을 특정 크기만큼 이동시킨다.
- 함수 설명 : WarpAffine(cross 도형을 그릴 Board, 결과값, 회전 방향, cross 도형의 크기)

4 :
- 회전된 도형을 보여주기 위해 fourBoxBoard와 crossBoardTemp를 합친다.
- fourBoxBoard와 crossBoardTemp는 서로 다른 Board에서 그렸기 때문에 이 두 Board를 합쳐서 하나의 Board로 보여주기 위한 것

5 :
- add 출력

C# 기반 OpenCV4를 사용한

객체 정렬 시스템

#주요 기능 #도형 이동 ② #move

move 함수 호출

```
Mat MoveValueXY_Board = new Mat();
        crossBoardTemp = new Mat(crossBoard.Size(), MatType.CV_8UC3);
        for (int i = 0; i <= (int)(Distance + 0.5); i++)
        {
           int newX = randomValueXY_C.X + -(int)(Math.Cos(pointDegree / RADIAN) * i),  // ~
                newY = randomValueXY_C.Y + (int)(Math.Sin(pointDegree / RADIAN) * i);    // 1
           MoveValueXY_Board= Move(crossBoard, newX, newY); // 2
           Spin(MoveValueXY_Board, crossBoardTemp, new Point(newX, newY), randomValueAngle_C + (int)axisDegree * direct); // 6
           DrawLine(Contour(crossBoardTemp), crossBoardTemp, 0);
           Cv2.Add(crossBoardTemp, fourBoxBoard, add);  // 7
           Cv2.ImShow("FPaint", add);  // 8
           Cv2.WaitKey(1);
        }
```

move 함수

```
static Mat Move(Mat board, int x, int y)
{
   Mat MoveValueXY_Board = new Mat(board.Size(), MatType.CV_8UC3); // 3
   Cv2.Rectangle(MoveValueXY_Board, new Point(x + 40, y) , new Point(x + 60, y + 100) , Scalar.Red, -1); // ~
   Cv2.Rectangle(MoveValueXY_Board, new Point(x, y + 40)  , new Point(x + 100, y + 60) , Scalar.Red, -1); // 4
   return MoveValueXY_Board; // 5
}
```

C# 기반 OpenCV4를 사용한

객체 정렬 시스템

#주요 기능 #도형 이동 ② #move #설명

1 :
- 이동 좌표 구하기

2 :
- 인자 설명 : (cross 도형을 그릴 Board, X축으로 이동할 좌표, Y축으로 이동할 좌표)

3 ~ 4 :
- 3 : 이동할 cross 도형을 그릴 보드 생성
- 4 : cross 도형 그리기 (cross를 그릴 Board, X축 좌표, Y축 좌표, 도형 색상, 도형 테두리)

5 :
- 이동할 cross도형이 그려진 Board 리턴

6 :
- crossBoardTemp 만큼 회전하고, MoveValueXY_Board 만큼 이동

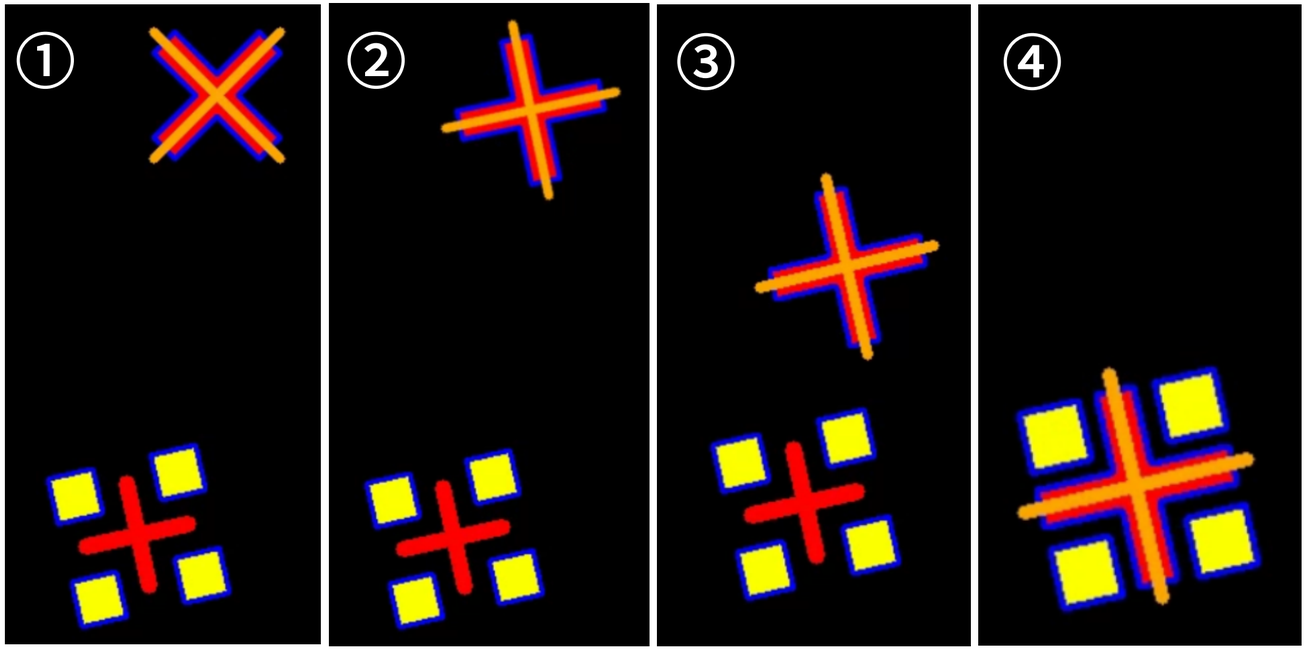
7:
- Align된 도형을 보여주기 위해 crossBoardTemp와 fourBoxBoard를 합친다.
- crossBoardTemp와 fourBoxBoard는 서로 다른 Board에서 그렸기 때문에 이 두 Board를 합쳐서 하나의 Board로 보여주기 위한 것

8 :
- 출력

C# 기반 OpenCV4를 사용한

객체 정렬 시스템

#결과



결과

☑ 두 도형이 떨어져 있는 거리와 틀어져 있는 각도를 계산하여 이 두 값의 차이 값 만큼 cross 도형이 fourBox로 align함을 확인할 수 있다.

C# 기반 OpenCV4를 사용한

객체 정렬 시스템

#배운점 #느낀점

배운점

- 객체 인식을 위한 인공지능과 정렬 알고리즘이 OLED Align S/W에 왜 필요한 기술인지 알게 되었다.

느낀점

- 친숙하지 않았던 C#과 이를 활용한 영상처리는 처음이라 초반에 의아한 부분이 많았지만,
  하다보니 흥미로운 점이 많았고 객체를 인식 하는 게 신기하고 재밌었다.
- OpenCV는 처음 접해봐서 구글링을 했어야 했는데 공식 홈페이지에는 C++, Java, Python에 관한 튜토리얼만 있어서 아쉬웠다.
- 친숙하지 않은 언어와 라이브러리로 단기프로젝트를 진행하다보니 여러 문제도 많아 힘들었지만 영상처리를 활용하여 align S/W를 개발하는 방법,
  처음 접하는 기술을 익히는 방법, 문제를 해결하는 방법을 배워갈 수 있어서 좋은 경험이었다.

C# 기반 OpenCV4를 사용한

객체 정렬 시스템

#아쉬운 점 #향후 개선점

아쉬운 점

- 약간의 오차값이 있음
- 코드가 깔끔하지 못 함

향후 개선 방향

- 오차값 범위를 최소화 시키기
- 오차 발생 횟수를 최소화 시키기
- 중복 되는 함수, 변수를 정리하기
- 모호하지 않고 직관성 있도록 깔끔하게 코드 정리 하기

감사합니다