

시스템 설계

TP3 System Design Use OpenCV

메카트로닉스공학부

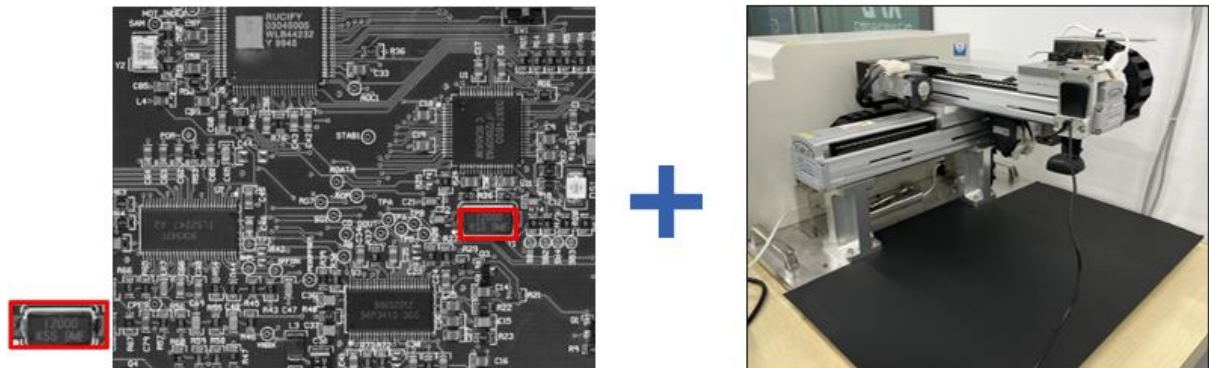
2017130010 김영준

담당교수 : 김효영 교수님

제출일 : 2022/06/21

TP3 주제선정 : 영상처리를 활용해 사용자가 원하는 객체만을 이용해 Template matching을 수행하는 공정. 또한 X-Y stage를 이용해 영역을 스캔하여 Object Detection을 수행하는 공정.

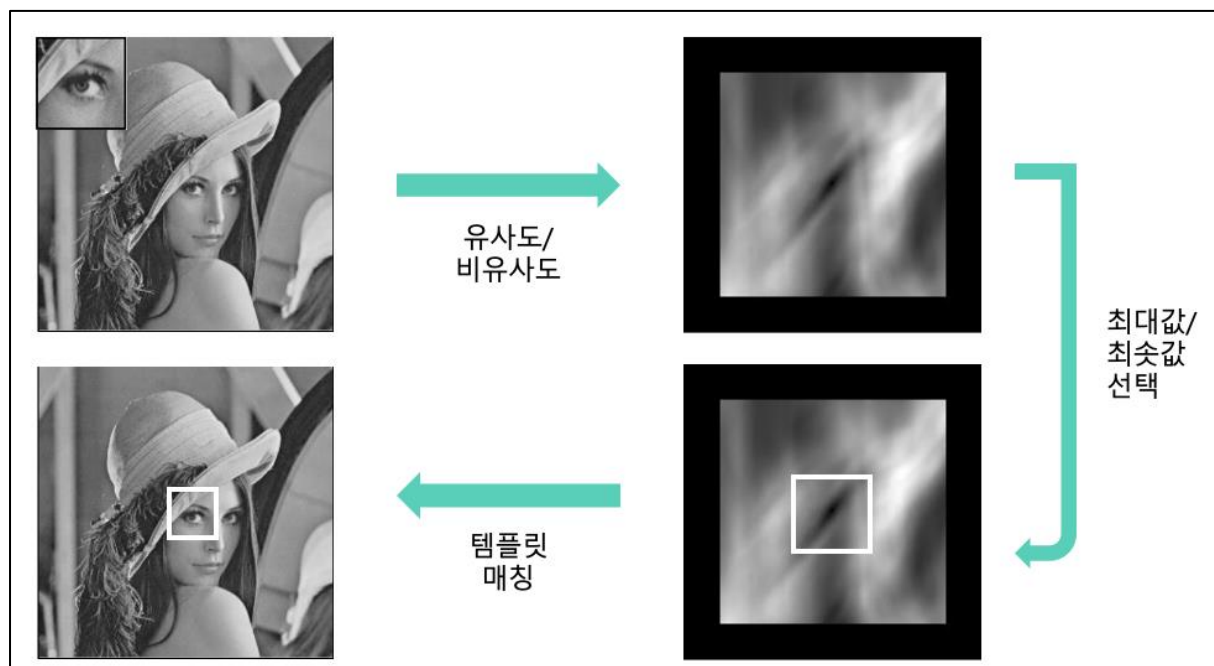
주로 FA분야에서 1) 부품 표면 결함 검사 2) 작업 상태 검사 3) 반도체 웨이퍼 결함 검사 4) 조립을 위한 위치 정합에 자주 쓰인다. 검사하는 물체는 동일하기 때문에 템플릿 매칭이 적합하다고 판단하였다.



(그림 1) 기판과 같은 경우 특정 부분의 결함을 검사하는데 사용됨

Template matching 이란?

➔ 원본 이미지에서 템플릿 이미지와 일치하는 영역을 찾는 알고리즘.



(그림2) Lena 눈을 찾는 과정으로 추출된 템플릿 영상이 원본 영상을 쭉 스캔하여 유사도를 검사.

현재 C++ 에서 사용하는 템플릿 매칭 수행 방법은 총 6가지가 있다.

method	설명
cv2.TM_SQDIFF	$R(x, y) = \sum_{x', y'} (T(x', y') - I(x + x', y + y'))^2$ <p>완전히 같으면 0, 다르면 값이 커짐</p>
cv2.TM_SQDIFF_NORMED	$R(x, y) = \frac{\sum_{x', y'} (T(x', y') - I(x + x', y + y'))}{\sqrt{\sum_{x', y'} T(x', y')^2 \cdot \sum_{x', y'} I(x + x', y + y')^2}}$ <p>[0, 1] 정규화</p>
cv2.TM_CCORR	$R(x, y) = \sum_{x', y'} (T(x', y') \cdot I(x + x', y + y'))$ <p>같으면 큰 값, 다르면 작은 값</p>
cv2.TM_CCORR_NORMED	$R(x, y) = \frac{\sum_{x', y'} (T(x', y') \cdot I(x + x', y + y'))}{\sqrt{\sum_{x', y'} T(x', y')^2 \cdot \sum_{x', y'} I(x + x', y + y')^2}}$ <p>[0, 1] 정규화</p>
method	설명
cv2.TM_CCOEFF	$R(x, y) = \sum_{x', y'} (T'(x', y') \cdot I'(x + x', y + y'))$ <p>평균 보정 후 Correlation 연산</p> $T'(x', y') = T(x', y') - 1 / (w \cdot h) \cdot \sum_{x', y'} T(x', y')$ $I'(x + x', y + y') = I(x + x', y + y') - 1 / (w \cdot h) \cdot \sum_{x', y'} I(x + x', y + y')$
cv2.TM_CCOEFF_NORMED	$R(x, y) = \frac{\sum_{x', y'} (T'(x', y') \cdot I'(x + x', y + y'))}{\sqrt{\sum_{x', y'} T'(x', y')^2 \cdot \sum_{x', y'} I'(x + x', y + y')^2}}$ <p>완전히 일치하면 1, 역일치하면 -1, 상호 연관성이 없으면 0</p>

(그림3) 템플릿 매칭 Method

해당 원리를 이용해 템플릿 이미지와 원본 이미지간의 매칭을 수행한다.

템플릿 매칭을 수행 할 때 주의할 점은 영상의 전처리가 수행되어야 한다.

원본영상에서 추출한 이미지를 사용하기 때문에 이미지 사이즈에 대한 변화가 있으면 안된다.

또한 RGB 3채널 영상을 사용할 경우 연산량이 많기 때문에 실시간에서 프레임이 굉장히 떨어진다는 단점이 있다. 따라서 Grayscale로 변환해주어 연산량을 대폭 줄여준다.

마무리로 가우시안 블러링을 사용해 템플릿 이미지에 대해 노이즈를 감소시켜 주어 데이터 활용을 용이하게 만든다.

Soultion

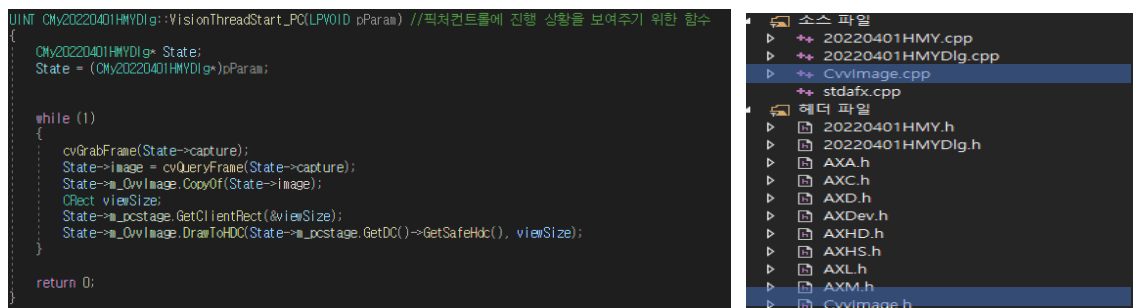
- Thread는 3개로 구성되어 있다.

- 1) 오브젝트를 캡처하기 위한 Thread
- 2) Picture Control에 영상을 띄우기 위한 Thread
- 3) Template Matching을 수행하는 Thread

```
static UINT VisionThreadStart_ObjectCapture(LPVOID pParam);
static UINT VisionThreadStart_PC(LPVOID pParam);
static UINT VisionThreadStart_TemplateMathcing(LPVOID nParam);
```

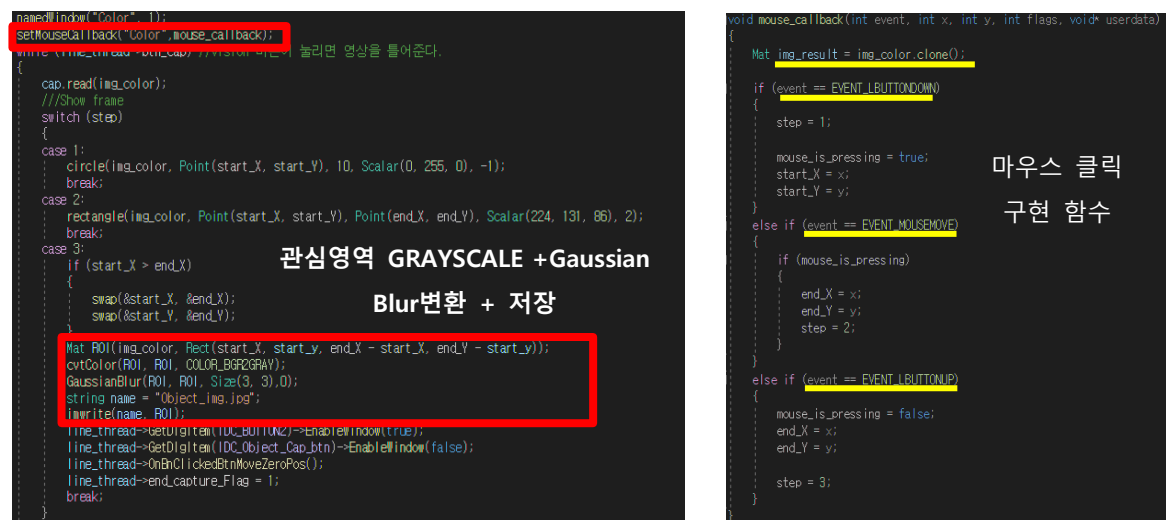
(그림4) 실제 코드에서 구현한 쓰레드

- Picture Control에 영상을 띄우기 위해서는 CvwImage class를 사용하였다.



(그림5) CvwImage를 추가로 다운받아 Picture Control 에서 실시간 영상을 송출한다.

- 마우스 드래그를 통해 ROI 영역 설정



(그림6) mouse_callback 함수로 마우스 클릭 좌표를 알아내고, 이에 해당하는 영역을 잘라낸다.

- Template Matching 구현

<code>matchTemplate(frame2, img, result, 3);</code>	<code>cv2.TM_CCOEFF</code>	상관계수 매칭: 완벽한 매칭 : 1, 나쁜 매칭 : -1, 상관관계 없음 : 0
<code>minMaxLoc(result, &minVal, &maxVal, &minLoc, &maxLoc, Mat());</code>	<code>cv2.TM_CCOEFF_NORMED</code>	상관계수 매칭 정규화
<code>printf("minva : %.2f, maxval : %.2f\n", minVal, (maxVal));</code>	<code>cv2.TM_CCORR</code>	상관관계 매칭: 완벽한 매칭일수록 값이 크며, 나쁜 매칭일 때 0
<code>if (maxVal > 0.92)</code>	<code>cv2.TM_CCORR_NORMED</code>	상관관계 매칭 정규화
<code>{</code>	<code>cv2.TM_SQDIFF</code>	제곱 차이매칭: 완벽한 매칭 : 0, 나쁜 매칭일수록 값이 커짐
<code> rectangle(frame2, minLoc, Point(minLoc.x + img.cols, minLoc.y + img.rows), Scalar(0, 0, 0, 0), 1);</code>	<code>cv2.TM_SQDIFF_NORMED</code>	제곱 차이 매칭 정규화
<code> TM_>OnBnClickedBtnXEstop();</code>		
<code> TM_>OnBnClickedBtnYEstop();</code>		
<code> imshow("Video", frame2);</code>		
<code> TM_>detect_object = true;</code>		
<code> TM_>MessageBox("객체를 찾았습니다!");</code>		
<code> TM_> pWinThread2= NULL;</code>		
<code> break;</code>		
<code>}</code>		

(그림7) matchTemplate() 함수를 사용해서 method중 하나를 채택해서 구현한다.

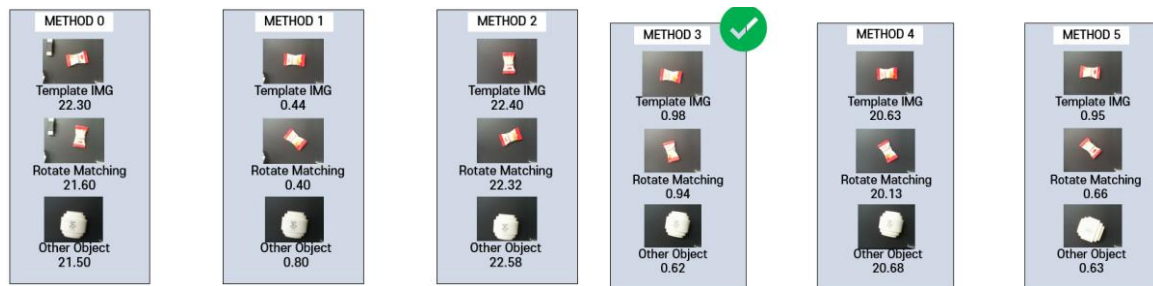
<code>matchTemplate(frame2, img, result, 3);</code>	연산 결과 변수에 저장하는 함수.
<code>minMaxLoc(result, &minVal, &maxVal, &minLoc, &maxLoc, Mat());</code>	
<code>printf("minva : %.2f, maxval : %.2f\n", minVal, (maxVal));</code>	
<code>if (maxVal > 0.92)</code>	
<code>{</code>	
<code> rectangle(frame2, minLoc, Point(minLoc.x + img.cols, minLoc.y + img.rows), Scalar(0, 0, 0, 0), 1);</code>	
<code> TM_>OnBnClickedBtnXEstop();</code>	
<code> TM_>OnBnClickedBtnYEstop();</code>	
<code> imshow("Video", frame2);</code>	
<code> TM_>detect_object = true;</code>	
<code> TM_>MessageBox("객체를 찾았습니다!");</code>	
<code> TM_> pWinThread2= NULL;</code>	
<code> break;</code>	
<code>}</code>	

(그림8) minMaxLoc함수를 사용하여 각 변수에 값을 저장해준다.

<code>matchTemplate(frame2, img, result, 3);</code>	
<code>minMaxLoc(result, &minVal, &maxVal, &minLoc, &maxLoc, Mat());</code>	
<code>printf("minva : %.2f, maxval : %.2f\n", minVal, (maxVal));</code>	
<code>if (maxVal > 0.9;</code>	
<code>{</code>	
<code> rectangle(frame2, minLoc, Point(minLoc.x + img.cols, minLoc.y + img.rows), Scalar(0, 0, 0, 0), 1);</code>	
<code> TM_>OnBnClickedBtnXEstop();</code>	
<code> TM_>OnBnClickedBtnYEstop();</code>	
<code> imshow("Video", frame2);</code>	
<code> TM_>detect_object = true;</code>	
<code> TM_>MessageBox("객체를 찾았습니다!");</code>	
<code> TM_> pWinThread2= NULL;</code>	
<code> break;</code>	
<code>}</code>	

(그림9) Method3에는 maxVal을 사용하는게 효과적, 본인 시스템에는 0.90 ↑ 적합.

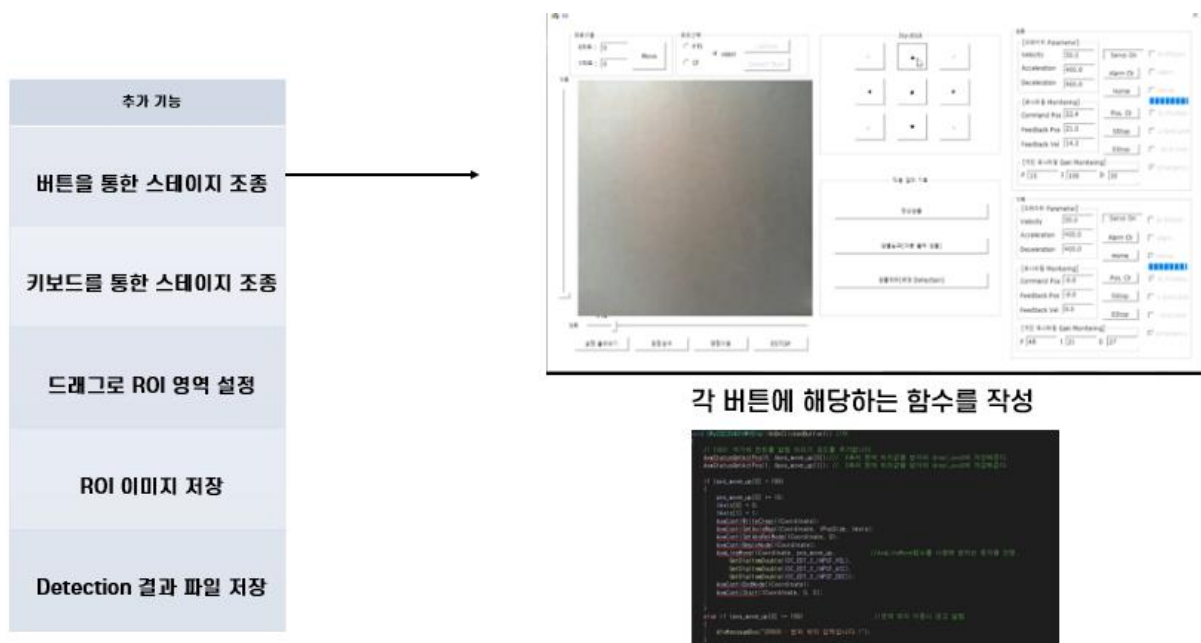
Template Matching을 할 때 총 6가지의 방법을 실험을 통해 어떤 방법이 적합한지 측정한다.



(그림10) 추출된 이미지와 회전된 이미지, 전혀 다른 물체에 대해 검출한 값을 비교한다.


추가 기능

1) 버튼 클릭을 통한 스테이지 조종



버튼 클릭으로 세세한 움직임을 구현할 수 있도록 한다.


2) 키보드를 통한 스테이지 조종

추가 기능		<pre> SQL_OMy202040(HRWD)g::PreTranslateMessage(MSG->Msg); if (move_key_flag) { if (((MSG->message == WM_KEYDOWN) (MSG->message == WM_SYSKEYDOWN)) && (MSG->wParam == VK_UP)) { MSG->wParam = VK_DOWN; } if (((MSG->message == WM_KEYDOWN) (MSG->message == WM_SYSKEYDOWN)) && (MSG->wParam == VK_DOWN)) { MSG->wParam = VK_UP; } if (((MSG->message == WM_KEYDOWN) (MSG->message == WM_SYSKEYDOWN)) && (MSG->wParam == VK_LEFT)) { MSG->wParam = VK_RIGHT; } if (((MSG->message == WM_KEYDOWN) (MSG->message == WM_SYSKEYDOWN)) && (MSG->wParam == VK_RIGHT)) { MSG->wParam = VK_LEFT; } } </pre>
버튼을 통한 스테이지 조종		
키보드를 통한 스테이지 조종		
드래그로 ROI 영역 설정		
ROI 이미지 저장		
Detection 결과 파일 저장		

**PretranslateMessage에서
키 입력에 대해 알아내
키보드 입력으로
스테이지를 조종**

키보드 입력으로 원하는 지점까지 빠르게 이동할 수 있도록 한다. 또한 영역 밖의 이동을 시도할 경우 알림 메시지가 나온다.

3) 드래그로 ROI 영역 캡처

추가 기능	
버튼을 통한 스테이지 조종	
키보드를 통한 스테이지 조종	
드래그로 ROI 영역 설정	
ROI 이미지 저장	
Detection 결과 파일 저장	

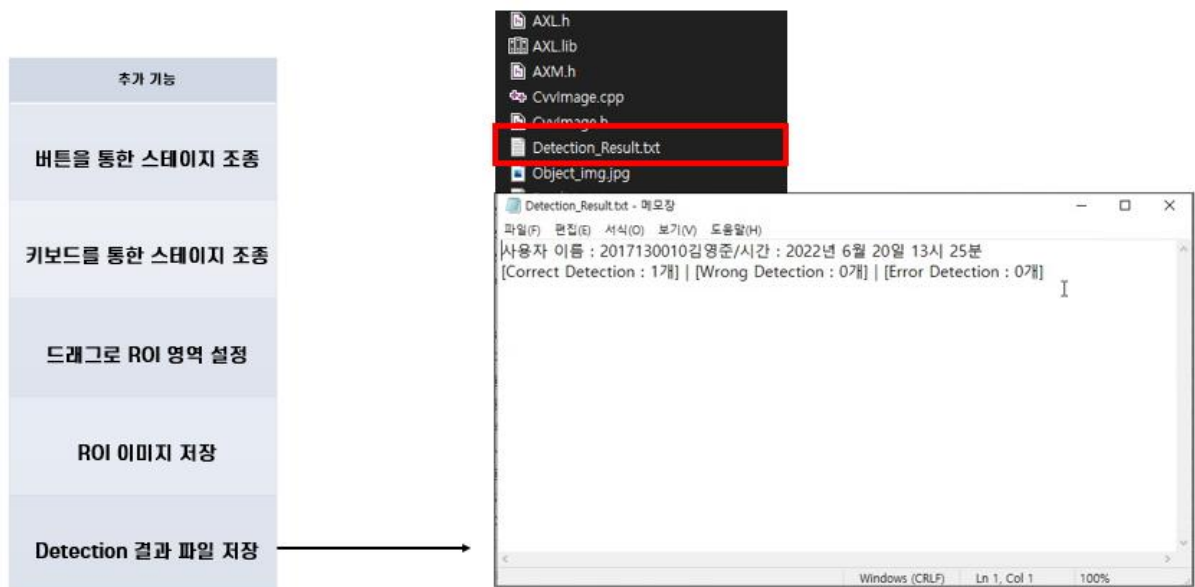
찾고자 하는 object에 대해 crop하게 캡처하여 매칭 정확도를 높여준다.

4) ROI 이미지 저장



프로젝트 폴더에 전처리가 완료된 이미지를 저장한다.

5) Detection 결과 파일 저장



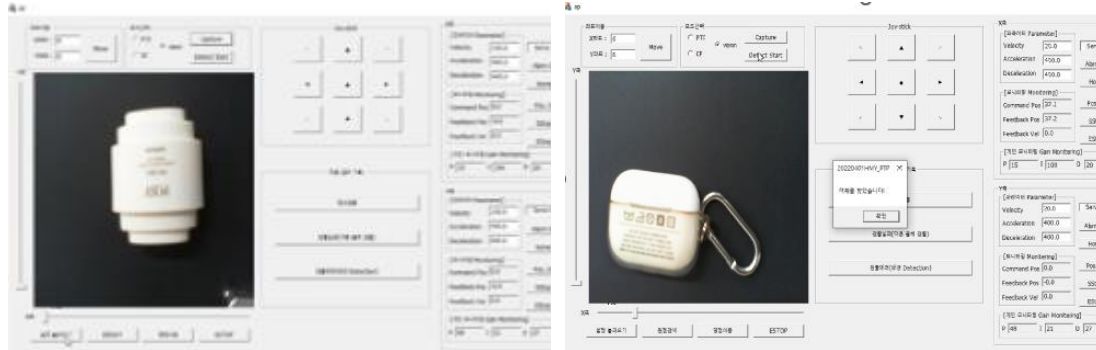
프로그램을 시작할 때 입력한 사용자의 이름과 사용 시간, 검출한 결과를 log파일로 저장한다

보완점

1. 템플릿 매칭의 한계

템플릿 매칭의 큰 단점은 15도 밖의 회전된 물체에 대해서는 정확도가 낮다.

또한 동일하게 생긴 물체에 대해서 잘못 검출하는 결과를 볼 수 있다.



2. 특징점 매칭의 한계

템플릿 매칭의 단점을 보완하기 위해 특징점 매칭 방법을 사용했다. 특징점 매칭은 템플릿 매칭보다 정확도가 높기 때문에 구현하였다.

하지만 각종 전처리를 수행하였음에도 굉장히 낮은 프레임과 큰 움직임이 생길 경우 프로그램이 다운되는 현상이 생겼기 때문에 사용하지 못했다.



추후 이 시스템에 대한 보완점으로는

템플릿 매칭을 통한 Detection 방식이 아닌 머신러닝 기법을 활용하여 같은 물체여도 회전, 겹침에 대한 문제를 해결할 수 있도록 한다.

또한 스테이지의 이동 경로를 다양화 또는 사용자가 직접 티칭 하는 방식으로 진행하고 단순 그레이스케일 변환과 블러 처리만이 아닌 다양한 전처리 기법을 활용하여 정확도를 높일 수 있도록 한다.