Align Algorithm

{S}

World coordinate

Mark

{G}

{W}

Stage

{C}

Camera

Glass

Coordinates

{W} 은 World Coordinates 으로 고정된 좌표계이다.

{S} 은 Stage Coordinates 으로 x,y,theta 으로 이동하면 원점에 있을때 {W} 와 동일하다. 이는 스테이지의 가상 회전 중심이고 실제로 스테이지의 원점을 왼쪽하단이라고 가정할때 이를 {B} 이라 한다.

{G} 은 대상 글라스좌표계이고 Glass 에 그려져 있거나 그릴 대상의 원점을 의미한다. 글라스의 가상회전 중심을 {GC} 이라 한다. 실제 글라스는 틀어져서 놓이기 때문에 이렇게 틀어져서 놓인 좌표계를 {GL} 이라 한다.

{C} 은 카메라 좌표계이다. 실제 픽셀의 좌표계는 {P} 이다. 카메라가 스테이지와 x,y방향이 다를때 이를 보정하는 좌표계는 {CA} 이다.

* 1. 좌표계변환

x

y

{W}

x

y

{B}

P\_BORG

P

P

위의 그림과 같이 좌표 변환이 일어난다고 할때

 - (3.1)

* 1. 마크 위치 변환

카메라에 어떤 마크가 픽셀 좌표계로 값을 가질때 이를 {W}으로 변환하는 식을 아래와 같다.

 - (3.2)

: i 번째 마크를 카메라로 측정했을때 pixel 좌표계로 측정된 값

 : 카메라의 pixel 좌표계를 카메라의 좌표계로 변환하는 변환 매트릭스이다. 아래 3.4를 참조한다. 카메라의 사양과 렌즈에 의해 결정된다.

: 카메라가 {W} 좌표계의 중심으로 부터 얼마나 떨어져 설치되 있는지를 의미하는 변환 매트릭스이다. 기계 설계시 결정된다.

: 카메라가 실제 설계위치에서 얼마나 벗어나 설치되었는지를 나타내고, 카메라 얼라인 알고리즘의 측정 대상이다.

이 마크가 글라스 위의 특정 위치에 해당하면 이를 {W} 으로 변환하는 식은 아래와 같다.

 -(3.3)

: 마크의 좌표값으로 이는 글라스 좌표계로 표현된 값이다.

: 글라스 좌표계로부터 글라스 중심으로의 변환 매트릭스이다. 이는 글라스 설계시 정해진다.

: 글라스가 스테이지에 놓일때 얼마나 틀어지는지를 나타내는 양이고 글라스 얼라인 측정 대상이다.

: 글라스가 스테이지에 놓이는 위치에 대한 좌표를 나타낸다. 기계 설계시 결정된다.

: 스테이지의 중심에서 스테이지 좌표계 까지의 변환을 나타낸다. 기계 설계시나 장비 제어시 스테이지 제어 편의성에 의해 결정된다.

: 스테이지의 제어값 다시 말해서 서보의 위치에 해당한다.

* 1. 카메라 위치 구하기

카메라의 resolution ( mm/pixel ) 과 dimension 을 rx, ry, W, H 이라 하면 아래와 같이 {C}->{P} 변환 메트릭스를 구할수있다.

-(3.5)

{C}->{CA} 은 여기서 구하고자 하는 카메라의 위치에 해당한다.

{CA}->{W} 가 장비에 카메라가 어떻게 설치되는지에 따라 사용자가 정한다.

스테이지 자체에 특정 마크를 만들었다고 가정하고 이를 스테이지를 이동하여 카메라로 이 마크의 위치를 얻었다고 하면 아래의 식을 얻을수 있다.

- (3.6)

(3.6) 식을 (3.2) 식에 대입하면 아래와 같다.



이때 {CA}->{W} 의 회전량이 작다고 가정하면 ( 5도 이하 ), 아래와 같이 Least Square Method 을 적용할 수 있다.

 - (3.7)

* 1. 글라스 로딩 에러 구하기

글라스의 특정 마크를 카메라를 사용하여 픽셀위치를 얻었다고 가정하면 (3.2) 을 (3.3) 에 대입할 수 있다.



위의 식은 (3.7) 의 least square method 을 적용하여 풀면 글라스가 얼마나 틀어져서 스테이지에 올라왔는지 구할수 있다.

* 1. 글라스의 마크를 카메라로 보기 위한 스테이지 이동량

식 (3.2) 와 (3.3) 에서 스테이지 이동량을  라 하면, 이는 다음과 같다.



여기서 카메라의 중심에서 마크를 보기위한 계산량이므로 pixel 좌표계는 사용되지 않고 카메라 좌표계에서 중심이므로 이 된다.

* 1. 얼라인 에러 계산

얼라인 에러는 측정된 offset 을 적용했을때 주어진 입력데이터와의 평균에러량으로 정의한다.



* + 1. 카메라 얼라인 보정 에러



카메라로 잡힌 마크의 {W} 좌표와 마크의 실제 레퍼런스 마크의 {W}좌표 사이의 차이를 에러로 계산한다.

* + 1. 글라스 로딩 보정 에러



글라스 마크의 {W} 좌표와 카메라로 측정된 마크의 {W} 좌표 사이의 차이를 에러로 계산한다.

1. 시뮬레이션
   1. Matlab 시뮬레이션
      1. 카메라 위치 구하기

|  |
| --- |
| Orientation = inline( '[ cos(x(3)) -sin(x(3)) x(1); sin(x(3)) cos(x(3)) x(2); 0 0 1]', 'x' );  TO\_RADIAN = pi / 180;  TO\_DEGREE = 180 / pi;    AlignCameraPixels = [ 1380 1035 ]';  AlignCameraResolution = [ 0.00465 0.00465 ]'; % um / pixel  MARK\_PIXEL\_WIDTH = 400; MARK\_PIXEL\_HEIGHT = 200;    Rca2w = Orientation( [ 0 0 0 ] );    CameraPosition = [ -275 375 5\*TO\_RADIAN ]'; % unknown value  Rc2ca\_u = Orientation( CameraPosition );    rx = AlignCameraResolution(1); ry = AlignCameraResolution(2);  W = AlignCameraPixels(1); H = AlignCameraPixels(2);  Rc2p = [ 1/rx 0 W/2 ; 0 -1/ry H/2; 0 0 1];  Rp2c = inv( Rc2p );    % Mark position known w.r.t Stage Bed  mw = (MARK\_PIXEL\_WIDTH/2) \* AlignCameraResolution(1);  mh = (MARK\_PIXEL\_HEIGHT/2) \* AlignCameraResolution(2);  M\_b = [  0+mw 0 1;  0 0+mh 1;  0-mw 0 1;  0 0-mh 1; ]';    % Jog x & y axis and find the mark  S = [ 100-50 700-50 0 ]';  R\_b2s = Orientation( [ -StageSize(1)/2+50 -StageSize(2)/2+50 0 ] );  R\_s2w = Orientation( S );    % capture image from the align camera and locate mark's edge - cross mark  cx = AlignCameraPixels(1)/2; cy = AlignCameraPixels(2)/2;  w = MARK\_PIXEL\_WIDTH/2; h = MARK\_PIXEL\_HEIGHT/2;  M\_p = [ +w 0 1; 0 -h 1; -w 0 1; 0 +h 1; ]';  M\_p = [ Rc2ca\_u(1:2,1:2) [ cx cy ]'; 0 0 1 ] \* M\_p;    b = inv(Rca2w) \* R\_s2w \* R\_b2s \* M\_b;  u = Rc2ca \* Rp2c \* M\_p;    A = [];  B = [];  for i = 1 : size( u, 2 )  A = [ A; 1 0 -u(2,i) u(1,i) ];  A = [ A; 0 1 u(1,i) u(2,i) ];  B = [ B; b(1,i) ];  B = [ B; b(2,i) ];  end    X = inv (A'\* A ) \* A' \* B;  Rc2ca = Orientation( X );  x = Rca2w \* Rc2ca \* [ 0 0 1]';  x(3) = X(3)  error\_distance\_of\_camera = sqrt( sum( ( x(1:3) - CameraPosition ) .^ 2 ) ); |

error\_distance\_of\_camera = 1.1072e-004

error = 7.7227e-005



* + 1. 글라스 틀어짐 구하기

|  |
| --- |
| Rc2ca = Orientation( [ 0 0 0 ] );  Rca2w = Orientation( [ -275 375 0\*TO\_RADIAN ] );    rx = AlignCameraResolution(1); ry = AlignCameraResolution(2);  W = AlignCameraPixels(1); H = AlignCameraPixels(2);  Rc2p = [ 1/rx 0 W/2 ; 0 -1/ry H/2; 0 0 1];  Rp2c = inv( Rc2p );    Rg2gc = Orientation( [ -150 -100 0 ] );  Rgc2gl = Orientation( [ 0 0 0 ] );  Rgl2b = Orientation( [ 200 150 0 ] );  Rb2s = Orientation( [ -StageSize(1)/2+50 -StageSize(2)/2+50 0 ] );    GlassMark\_g = [ 10 10 1; 290 10 1; 290 190 1; 10 190 1; ]';    % find out stage displacement to see the mark on the camera  u = Rb2s \* Rgl2b \* Rg2gc \* GlassMark\_g;  v = Rca2w \* Rc2ca \* Rp2c \* [ ones(1,4) \* W/2; ones(1,4) \* H/2; ones(1,4) ];  StageMove = v - u;    % move the Stage as StageMove  % then capture image from the align camera and get the mark position  cx = AlignCameraPixels(1)/2; cy = AlignCameraPixels(2)/2;  GlassMark\_p = [ cx+2 cy+1 1; cx+2 cy+1 1; cx+2 cy+1 1; cx+2 cy+1 1; ]';      U = []; V = [];  for i=1:size(u,2)  u = Rg2gc\* GlassMark\_g(:,i);    Rs2w = Orientation( StageMove(:,i)' );  v = inv( Rs2w \* Rb2s \* Rgl2b ) \* ( Rca2w \* Rc2ca \* Rp2c \* GlassMark\_p(:,i) );    U = [ U;  1 0 -u(2) u(1);  0 1 u(1) u(2); ];  V = [ V; v(1); v(2) ];  end    X = inv(U' \* U) \* U' \* V; |

X = 0.0093 -0.0046 0 1.0000

