컴퓨터비전 프로젝트 2

2021-04-13

20162874 이준협

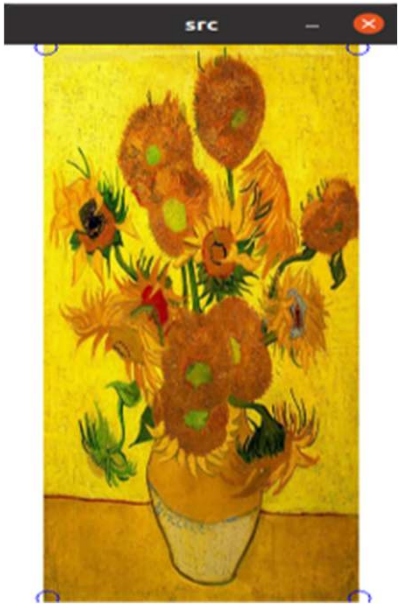
Basic Information

Software : MATLAB

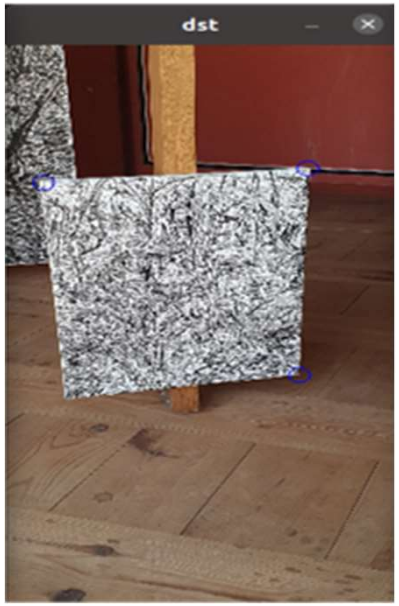
Used File Exchange : https://kr.mathworks.com/matlabcentral/fileexchange/30809-ransac-algorithm-with-example-of-finding-homography?s_tid=srchtitle
(필요한 부분은 수정 후 재구현 진행하였음)

Github repository : https://github.com/jjunhyub/computer_vision_project_2

Basic Project setting



Source image



Target image

Basic Homography setting

- Outliers는 없다는 가정하에 진행하므로 ransac의 rotation의 횟수는 1이며 random selection도 필요 없다.
- Homography를 구하는 과정에서 mosaic.pdf 에 나와 있는대로 eigenvalue를 직접 구하는 대신 이미 구현 된 svd 함수를 통해 간편하게 결과를 얻었다.

Compute Homography using DLT

```
n = size(pts1,2);
A = zeros(2*n,9);

% x1 200  y1 337  x2 171 y2 94
%    23      337     166    210
%    23      24      20     100
%    200     24      34     222

A(1:2:2*n,1:2) = -pts1';
A(1:2:2*n,3) = -1;
A(2:2:2*n,4:5) = -pts1';
A(2:2:2*n,6) = -1;
x1 = pts1(1,:)';
y1 = pts1(2,:)';
x2 = pts2(1,:)';
y2 = pts2(2,:)';
A(1:2:2*n,7) = x2.*x1;
A(2:2:2*n,7) = y2.*x1;
A(1:2:2*n,8) = x2.*y1;
A(2:2:2*n,8) = y2.*y1;
A(1:2:2*n,9) = x2;
A(2:2:2*n,9) = y2;
disp(A);
```

$$\begin{bmatrix} -u_1 & -v_1 & -1 & 0 & 0 & 0 & u_1u_1' & v_1u_1' & u_1' \\ 0 & 0 & 0 & -u_1 & -v_1 & -1 & u_1v_1' & v_1v_1' & v_1' \\ & & & & \vdots & & & & \\ 0 & 0 & 0 & -u_n & -v_n & -1 & u_nv_n' & v_nv_n' & v_n' \end{bmatrix}$$

-23	-24	-1	0	0	0	460	480	20
0	0	0	-23	-24	-1	2300	2400	100
-200	-24	-1	0	0	0	34200	4104	171
0	0	0	-200	-24	-1	18800	2256	94
-23	-337	-1	0	0	0	782	11458	34
0	0	0	-23	-337	-1	5106	74814	222
-200	-337	-1	0	0	0	33200	55942	166
0	0	0	-200	-337	-1	42000	70770	210

-23	-24	-1	0	0	0	460	480	20
0	0	0	-23	-24	-1	2300	2400	100
-200	-24	-1	0	0	0	34200	4104	171
0	0	0	-200	-24	-1	18800	2256	94
-23	-337	-1	0	0	0	782	11458	34
0	0	0	-23	-337	-1	5106	74814	222
-200	-337	-1	0	0	0	33200	55942	166
0	0	0	-200	-337	-1	42000	70770	210

다음과 같이 A를 직접 만들어 주었으며 결과는 위와 같이 나온다.
여기서 svd 함수에 A를 적용해 homography를 얻어낸다.

```
%% Homogeneous least squares : find 'h' minimizing ||Ah||^2
% Eigenvectors of A'*A corresponding to smallest eigenvalue
[~, ~, V] = svd(A);
h = V(:, end);
h = reshape(h,[3,3])';
% for easy computation, divide by end of h value
h = h/h(end);
```

0.9364	0.0619	-2.6149
0.0017	0.5037	89.9059
3.8551e-04	4.7748e-04	1

Homography

Show result by using calculated homography

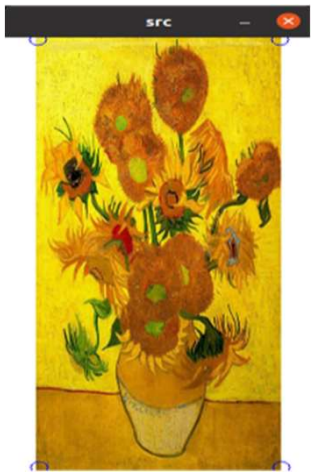
```
point1  = [23,24 ;200,24;23,337;200,337]'; % source
point2  = [20,100;171,94;34,222;166,210]'; % target

[H,~] = findHomography(point2,point1);
H = H';
figure(2);
H_inv = inv(H);
tform = projective2d(H_inv);
warped_pic1 = imwarp(pic1,tform);
imshow(warped_pic1);
```

1. 후방 변환을 적용하기 위해 처음부터 homography를 구할 때 target 으로 부터 source로 변환하는 homography를 구함.

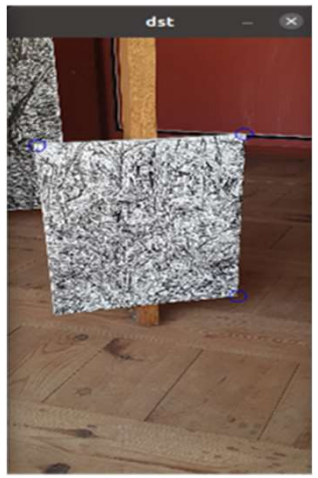
2. Matlab의 x,y축이 설정해준 값이랑 뒤집혀있어 transpose 적용

3. Target 으로부터의 변환이므로 inverse homography를 Projective2d에 적용해 이미지 워핑을 실행



Source image

Warped image



Target image

현재 전체 이미지에 구해진 homography를 적용하는 부분까지 완성했습니다.

Bilinear 2D 보간법을 사용해 source image의 정해진 부분만 warping 해서
target image로 대처하는 부분은 아직 구현하지 못했습니다.

늦더라도 꼭 구현해서 다시 제출하도록 하겠습니다.