바둑판 최외각 네 점 검출 알고리즘

윤여경, *이윤구
광운대학교 컴퓨터 과학과
cahny0628@gmail.com, *yglee96@kw.ac.kr

Algorithm of detecting Go board corners

Yeo Kyung Yoon *Yun Gu Lee
Department of Computer Science, Kwangwoon University

요 약

본 논문에서는 바둑판을 촬영한 정지 영상에서 격자무늬의 특징을 이용하여 바둑판의 최외각 네 점을 검출하는 방법을 제시한다. 바둑판은 수평 및 수직 방향으로 각각 19 개의 선이 격자무늬를 이룬다. 영상에는 일반적으로 노이즈가 포함되어 있기 때문에 허프(Hough) 변환의 임계값(threshold)을 이용한 선 패턴 추출 방식으로는 모든 선에 대한 정보를 얻을 수 없다. 따라서, 제안하는 알고리즘은 허프(Hough) 변환을 수행한 뒤에 추출된 선들이 이루는 교차점의 정보를 이용하여 바둑판의 최외각 네 점을 예측한다. 실험 결과는 실제 바둑판의 최외각 네 점과 비교하여 예측된 최외각 네 점의 에러 값을 제시한다. 이는 제안된 알고리즘이 바둑판의 최외각 네 점을 성공적으로 검출한다는 것을 입증한다.

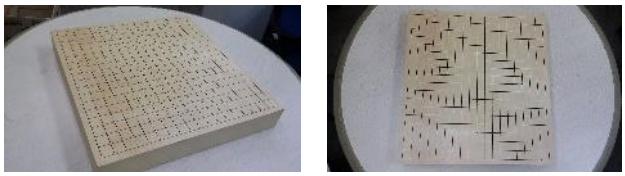
1. 서론

2016 년 Google Deep Mind 챌린지 매치에서 인공지능 알파고(Alphago)와 이세돌 9 단의 바둑 대국 이후 바둑에 대한 관심이 증가하고 있다. 더불어 바둑 기보에 대한 관심도 증가하는 실정이다. 바둑 대국이 벌어지는 동안 바둑판 위에 놓아지는 바둑돌의 순서와 위치를 기록한 정보를 기보라 한다. 기보를 통해 바둑 학습이 가능하고 이는 알파고와 같은 인공지능의 데이터로도 활용 가능하다. 그러나 아직까지 기보 작성은 전적으로 수작업에 의존하고 있다. 기존에 이루어진 영상처리 기술을 이용한 기보 작성의 자동화에 관한 연구 [1, 2, 3]는 한계점으로 인해 상용화에 어려움이 있다.

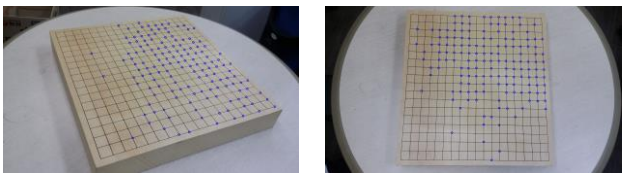
자동 기보 작성에서 가장 먼저 수행되어야 할 작업은 영상에서 바둑판을 인식하는 것이다. 그 다음 바둑돌을 인식함으로써 자동 기보 작성이 이루어진다.

바둑판은 어떤 각도에서 바라봐도 사각형의 모양을 가지며 가로방향의 19 개 선과 세로방향의 19 개 선으로 이루어진 격자무늬가 있다. 그러므로 바둑판을 수직 위에서 촬영한 영상의 경우에는 기존에 연구된 방법 [2]에 의해 선 패턴을 이용하여 바둑판 최외각 네 점을 검출할 수 있다. 그러나 이 방법을 이용하기 위해서는 카메라를 항상 바둑판의 수직 위에 설치해야 한다는 제약 조건이 발생한다. 따라서 [그림 1]과 같이 바둑판을 측면에서 촬영한 경우에도 바둑판의 최외각 네 점 인식이 가능해야 한다.

본 논문에서는 바둑판의 특징을 이용하여 선과 선이 이루는 교점을 추출함으로써 바둑판 격자무늬의 최외각 네 점을 예측하는 방법을 제안한다. 교점을 추출하기 위해 코너점을 검출하는 방법인 해리스 코너(Harris Corner) [4]를



[그림 1], 카메라 위치에 따른 바둑판의 모양



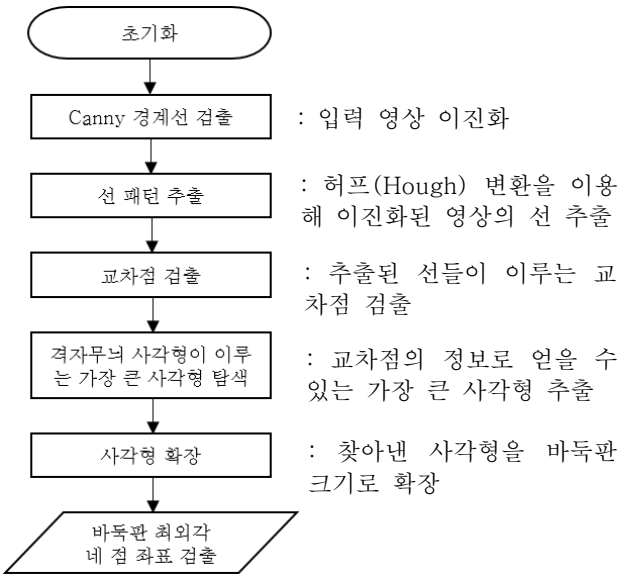
[그림 2], 해리스 코너(Harris Corner)로 코너점을 검출한 결과 영상 (검출된 코너점은 파란색 원으로 표시됨)

사용하면 모든 코너점을 완벽하게 검출하지 못하는 한계점이 있다([그림 2]). 따라서, 제안된 방법에서는 허프(Hough) 변환을 이용하여 선을 추출한 뒤에 두 선이 교차하는지에 대한 여부를 판단하는 작업을 수행함으로써 교점을 검출한다. 이 방법은 바둑판 격자무늬의 모든 선과 교점에 대한 정보를 완벽하게 수집하지 않고 격자무늬를 이루는 사각형을 이용하여 바둑판의 최외각 네 점을 예측하는 알고리즘을 제안한다. 이는 다양한 위치 및 각도에서 바둑판을 촬영하더라도 최외각 네 점 인식을 가능하게 한다.

본 논문에서 제안된 알고리즘의 입력 영상은 바둑돌이 한 개도 놓여있지 않은 바둑판을 촬영한 정지 영상이며 바둑판이 전체 영상 크기의 2/3를 차지한다는 조건을 만족해야 한다.

2. 본론

2.1 알고리즘 프레임워크(Framework)



[그림 3], 알고리즘 프레임워크

2.2 Canny 연산자를 이용한 경계선 검출

선 검출을 위해서 입력 영상을 이진화 하는 작업을 선행한다. 이진화 과정에서 물체의 특징을 최대한 유지할 수 있는 Canny 연산자를 이용한 경계선 검출 알고리즘 [5]을 적용한다. Canny 경계선 검출 임계값(threshold)의 low 값과 high 값을 각각 50, 200 으로 설정한다. 이는 70 개의 영상으로 테스트한 결과를 통해 실험적으로 정하였다.

2.3 허프(Hough) 변환을 이용한 선 패턴 추출

바둑판 격자무늬의 선 패턴을 추출하기 위해 직선 허프 변환(Line Hough Transform)을 이용한다. OpenCV 에 구현된 허프(Hough) 변환 함수 cvHoughLines2 의 CV_HOUGH_PROBABILISTIC 옵션을 이용하여 선을 검출한다. 수직 픽셀 크기가 1920 이고 수평 픽셀 크기가 1080 인 영상에 대해 유효한 직선의 크기를 30 픽셀로 정하여 영상 좌표계에서 길이가 30 픽셀보다 작은 직선은 제거한다. 또한 잡음을 보다 최소화하기 위해 선분 사이의 최대 간격을 10 으로 설정하였다. 이를 통해 동일한 직선이 아닌 작은 선분이 모여 하나의 선분이 되는 것을 방지한다.

그러나 허프(Hough) 변환은 임계값(threshold)으로 선 성분을 판단하기 때문에 실제로는 직선이 아닌 픽셀들이 임계값 이상의 값을 갖게 되는 경우가 발생한다. 이는 선 검출 결과에서 노이즈(noise)가 된다. 따라서, 바둑판 최외각 네 점을 인식하기 위한 신뢰할 수 있는 정보를 수집하기 위해 추출된 선들이 이루는 교차점을 검출한다.

2.4 교차점 검출

2.3 에서 추출한 모든 선들에 대해 교차 여부를 판단하여 교차점 P_{c_i}을 검출한다(수식 1,2). 그리고 P_{c_i}의 좌표와 P_{c_i}에서 교차하는 두 직선의 기울기, g_{1_i} 과 g_{2_i} 을 수집한다. 이는 격자무늬의 가로줄 19 개와 세로줄 19 개 즉, 38 개의 선 중에서 동일 선상에 존재하는 교차점들을 분류할 수 있게 한다.

$$A = (x_3 - x_2)\times(y_0 - y_2) - (y_3 - y_2)\times(x_0 - x_2)$$
$$B = (x_1 - x_0)\times(y_0 - y_2) - (y_1 - y_0)\times(x_0 - x_2)$$
$$C = (y_3 - y_2)\times(x_1 - x_0) - (x_3 - x_2)\times(y_1 - y_0)$$

$$P_{c_i}(x,y) = \begin{cases} \text{true}, & \text{if}\left(0 \le \frac{A}{C} \le 1, 0 \le \frac{B}{C} \le 1\right) \\ false, & \text{otherwise} \end{cases} \quad (1)$$

$$x_{p_{c_i}} = x_0 \times \frac{A}{C} \times (x_1 - x_0), \quad y_{p_{c_i}} = y_0 \times \frac{A}{C} \times (y_1 - y_0) \quad (2)$$

수식(1,2)에서 $(x_0, y_0), (x_1, y_1)$ 와 $(x_2, y_2), (x_3, y_3)$ 는 각각 2.3 에서 추출된 선분들 중 교차 여부를 판단하는 임의의 두 선분의 양 끝 점의 좌표를 의미한다.

2.4.1 불필요한 교차점 제거

2.3 을 수행하면 격자무늬의 직선들 주변에 다중으로 선분이 추출되는 경우가 발생한다. 이는 격자무늬 사각형의 꼭지점 외에 불필요한 교차점들 즉, 노이즈(noise)를 생성한다. 따라서 노이즈 제거를 위해 일정 범위 내 존재하는 교차점들 N 을 모두 제거하고 N 의 원소들의 평균 좌표 값에 위치한 점, $P_{g_i}(x,y)$ 로 대신한다(수식 3).

$$N = \left\{P_{c_\alpha}, P_{c_{\alpha+1}}, \cdots, P_{c_{\alpha+\beta}}\right\} \ (\alpha \ge 0, \beta \ge 0, \alpha + \beta \le n)$$

$$P_{g_i}(x,y) = \left(\frac{\sum_{i=\alpha}^{\alpha+\beta} x_{p_{c_i}}}{\beta - \alpha + 1}, \frac{\sum_{i=\alpha}^{\alpha+\beta} y_{p_{c_i}}}{\beta - \alpha + 1}\right) \quad (3)$$

2.5 격자무늬 사각형이 이루는 가장 큰 사각형 탐색

바둑판의 격자무늬는 동일한 크기를 가진 18×18 개의 사각형 Q로 이루어져 있으므로 Q에 대한 정보를 알면 바둑판의 최외각 네 점을 인식할 수 있다. 이 정보는 Q 의 네 꼭지점 좌표와 가로 및 세로 길이 값을 뜻한다. 그러나 단순히 교차점의 정보만으로는 Q의 정보를 얻는 데 어려움이 있다.

2.4 는 격자무늬의 모든 교차점을 완벽하게 검출하지 않는다. 2.3 에서 추출한 선이 모든 격자무늬의 직선을 추출하지 못하기 때문이다. 이로 인해 2.4 에서 생성된 집합 P_c은 몇몇 Q 의 꼭지점만을 포함한다. 따라서 Q 의 정보를 얻기 위해 P_c 의 원소 중 네 개를 꼭지점으로 가지고 가장 많은 수$(m \times n)$의 Q로 이루어진 사각형 Q_{max}을 찾는다. 이 작업은 다음 절차에 따라 수행한다.

1. 모든 P_C 에 대해서 임의의 P_{c_i} 에 대해 특정 조건을 만족하면서 P_{c_i} 에서부터 일정 거리 내 위치한 P_C 의 원소들로 이루어진 집합 S_i을 구한다. 다음 조건(a), (b)은 P_C의 원소들이 가지는 g_1과 g_2을 이용하여 비교함으로써 판단이 가능하다.

(a) P_{c_i}와 같은 방향의 직선에 위치한 점인지 판단한다.

(b) P_{c_i}와 동일 직선상에 위치한 점인지 판단한다.

2. 수식 4 와 같이 집합 $L=\{L_0,L_1,\cdots,L_i\}$ ($i \geq 38$)이 정의된다. L 의 원소 중 수평 방향의 기울기를 가지는 임의의 L_i, L_j와 수직 방향의 기울기를 가지는 임의의 L_k, L_l을 선택하여 사각형 q_i을 만들 수 있는 네 꼭지점을 구한다. 이는 수식 5 에 의해 표현된다. 수식 5 에서 q_{i_0}, q_{i_1}, q_{i_2}, q_{i_3}은 q_i의 네 꼭지점을 의미한다.

$$L_i=\Big(\big((S_i\cup S_j)\cup S_k\big)\cdots\Big)\cup S_n,\quad \big(S_i\cap S_j\neq\emptyset,\quad (S_i\cup S_j)\cap S_k\neq\emptyset,\quad\cdots,\quad \Big(\big((S_i\cup S_j)\cup S_k\big)\cdots\Big)\cap S_n\neq\emptyset\big) \tag{4}$$

$$\begin{aligned}&\big(L_i(r)=L_k(s)\big)\rightarrow\big(q_{i_0}(x,y)=L_i\big(r(x,y)\big)=L_k\big(s(x,y)\big)\big)\\&\big(L_k(s)=L_j(t)\big)\rightarrow\big(q_{i_1}(x,y)=L_k\big(s(x,y)\big)=L_j\big(t(x,y)\big)\big)\\&\big(L_j(t)=L_l(u)\big)\rightarrow\big(q_{i_2}(x,y)=L_j\big(t(x,y)\big)=L_l\big(u(x,y)\big)\big)\\&\big(L_l(u)=L_i(r)\big)\rightarrow\big(q_{i_3}(x,y)=L_l\big(u(x,y)\big)=L_i\big(r(x,y)\big)\big)\\&q_i\{q_{i_0},q_{i_1},q_{i_2},q_{i_3}\}\\&=\begin{cases}\text{true}, & \text{if}\big(q_{i_0}\neq\emptyset,q_{i_1}\neq\emptyset,q_{i_2}\neq\emptyset,q_{i_3}\neq\emptyset\big)\\ false, & \text{otherwise}\end{cases}\end{aligned} \tag{5}$$

3. 가장 큰 크기를 가지는 q_i 즉, Q_{max}를 찾는다. (수식 6). 결과적으로 Q_{max}의 네 꼭지점의 좌표를 검출한다.

$$\begin{aligned}&\mathrm{d}(q_{i_0},q_{i_1})=\sqrt{\Big(\big(x_{q_{i_0}}-x_{q_{i_1}}\big)^2+\big(y_{q_{i_0}}-y_{q_{i_1}}\big)^2\Big)}\\&\mathrm{a}(q_i)=\ \mathrm{d}(q_{i_0},q_{i_1})\times\mathrm{d}(q_{i_0},q_{i_2})\\&Q_{max}=\max\{\mathrm{a}(q_0),\mathrm{a}(q_1),\cdots,\mathrm{a}(q_n)\}\end{aligned} \tag{6}$$

2.6 격자무늬 사각형 확장

$m\times n$개의 Q로 이루어진 Q_{max}를 18×18 개의 Q로 이루어진 형태가 되도록 수직 및 수평 방향으로 확장함으로써 바둑판의 최외각 네 점의 좌표를 검출한다.

Q_{max}을 확장하기 위해서 우선, 바둑판을 수직 위에서 촬영한 것처럼 바둑판이 직사각형 모양이 되도록 영상을 변형시킨다. 임의의 직사각형 Q'_{max}의 꼭지점 좌표 네 개를 설정한 뒤에 Q_{max}와 Q'_{max}간의 호모그래피(Homography) H 을 이용해 Q_{max}을 Q'_{max}의 꼴로 변형시킨다. H 는 OpenCV 에 구현된 호모그래피(Homograhpy)계산 함수 findHomography 를 이용하여 구한다.

Q'_{max}는 Q_{max}을 수직 위에서 본 모양이다. 즉, Q'_{max}는 Q_{max}와 같이 $m\times n$개의 사각형 Q로 이루어져 있다. 또한, Q'_{max}는 직사각형 모양이므로 수식 7 에 의해 m, n의 값을 구할 수 있다. 수식 7 에서 p_0, p_1, p_2, p_3는 각각 Q'_{max}의 네 꼭지점 중 좌 상단, 우 상단, 우 하단, 좌 하단의 점을 의미한다. 격자무늬를 이루는 모든 Q의 네 꼭지점들의 집합 P_v에 대해서 $V\in P_v$ 을 만족하는 α, β는 각각 m, n 의 값에 해당하게 된다. 따라서, m, n 의 값과 Q의 가로 및 세로 길이를 알고 있으므로 Q'_{max}을 확장하여 18×18 개의 Q 로 이루어진 사각형 R의 모양으로 만들 수 있다. 즉, R의 네 꼭지점 좌표를 알 수 있다. 결과적으로 R의 네 꼭지점 좌표에 H을 역으로 취한 값은 원래 영상의 바둑판의 네 꼭지점 좌표 값을 의미한다.

$$\begin{aligned}&w=x_{p_0}-x_{p_1},\quad h=y_{p_0}-y_{p_3}\\&V_i(x,y)=\big(x_{p_0}+(w\div\alpha)\times\gamma,y_{p_0}+(h\div\beta)\times\delta\big),\\&(1\leq\alpha\leq18,1\leq\beta\leq18,0\leq\gamma\leq18,0\leq\delta\leq18)\end{aligned} \tag{7}$$

3. 실험 결과

제안된 바둑판 최외각 네 점 검출 알고리즘은 C++을 사용하여 구현되었으며 다양한 해상도와 카메라 각도에서 바둑판을 촬영한 이미지 세트를 통해 테스트되었다. 캐니 경계선 검출(Canny Edge Detection) 및 허프 변환(Hough Transform)은 OpenCV 에 구현된 이미지 처리 기술로 수행되었다. C++ 코드는 Visual Studio 2015 를 사용하여 컴파일 되었다.

보고된 실험의 모든 입력 영상은 1920 × 1080 로 크기를 조절한 다음 알고리즘을 적용하였다. 그리고 2.4 에서 바둑판의 격자무늬를 이루는 수직 방향의 선 19 개와 수평 방향의 선 19 개 즉, 38 개의 선마다 최소한 2 개 이상의 교차점이 검출될 됐다는 가정 하에 알고리즘을 진행하였다.

[표 1]은 [표 2]의 각 case 별 입력 영상에 대한 정보이다. [표 2]는 영상에서 실제 바둑판 최외각 네 점의 좌표와 알고리즘을 적용하여 검출한 예측 좌표의 오차를 분석한 결과이다. [표 2]의 두번째 행의 P_0, P_1, P_2, P_3 는 각각 바둑판의 최외각 네 점 중 좌 상단, 우 상단, 우 하단, 좌 하단에 위치하는 점의 좌표 값을 의미한다. 네번째 행의 예측 좌표 값은 실험 결과로 검출된 바둑판 최외각 네 점의 좌표이다. 다섯번째 행의 에러 값은 실제 좌표와 예측 좌표

	촬영 카메라 해상도	촬영된 원본 영상 크기(단위: cm)
Case 1	1280 × 720	3264 × 1836
Case 2	1280 × 720	3264 × 1836
Case 3	1920 × 1080	4608 × 3456
Case 4	1920 × 1080	4608 × 3456

[표 1], 입력 영상의 정보

		실제 좌표	예측 좌표	에러	평균 에러
Case 1	$P_0(x,y)$	339 , 179	340.33 , 179.73	1.51	2.51
	$P_1(x,y)$	1169 , 79	1171.17 , 82.90	4.46	
	$P_2(x,y)$	1762 , 651	1762.68 , 652.88	1.99	
	$P_3(x,y)$	635 , 989	637.04 , 989.45	2.08	
Case 2	$P_0(x,y)$	556 , 60	556.81 , 62.96	3.06	2.58
	$P_1(x,y)$	1399 , 83	1397.02 , 85.53	3.21	
	$P_2(x,y)$	1448 , 1003	1446.92 , 1003.06	1.08	
	$P_3(x,y)$	490 , 994	492.97 , 994.06	2.97	
Case 3	$P_0(x,y)$	523 , 190	519.44 , 193.11	4.72	1.89
	$P_1(x,y)$	1409 , 189	1410.77 , 189.66	1.88	
	$P_2(x,y)$	1875 , 625	1874.01 , 625.50	1.10	
	$P_3(x,y)$	27 , 622	27.78 , 622.42	0.88	
Case 4	$P_0(x,y)$	930 , 79	929.98 , 80.56	1.56	1.07
	$P_1(x,y)$	1683 , 326	1683.40 , 326.41	0.57	
	$P_2(x,y)$	1089 , 949	1089.76 , 950.52	1.69	
	$P_3(x,y)$	219 , 452	219.23 , 452.43	0.48	

[표 2], 최외각 네 점의 실제 좌표와 검출된 예측 좌표 및 실험 결과 분석

간의 거리 값을 의미한다.

[표 2]에서 보는 것과 같이 알고리즘을 적용하여 예측된 바둑판 최외각 네 점의 위치와 영상의 실제 바둑판 최외각 네 점의 위치는 평균 2 픽셀의 오차가 발생한다. 이는 영상의 전체 픽셀 수에 비하여 무시할 수 있는 정도이다. 따라서 제안된 검출 알고리즘의 강한 인식 능력을 확인할 수 있다.

4. 결론 및 향후 연구 방향

본 논문에서 제안하는 방식은 검출되어야 하는 교차점의 제약 조건이 있다. 또한 바둑판 주변에 장애물이 많은 경우, 강한 노이즈의 발생으로 바둑판 최외각 네 점을 검출하는 데 어려움이 있다.

알고리즘의 성능을 향상시키기 위해서 영상에 적절한 저역 필터(Low pass filter)를 사용하면 영상의 노이즈를 감소시킬 수 있을 것으로 기대한다.

추후에는 선과 교차점을 제외한 바둑판의 특징을 이용하여 임의의 영상에서 바둑판의 최외각 네 점을 완벽하게 예측할 수 있는 작업을 추가함으로써 교차점 검출의 제약 조건을 보완할 예정이다.

Acknowledgement

본 연구는 미래창조과학부 및 정보통신기술진흥센터의 디지털콘텐스 원천기술개발사업의 일환으로 수행하였음. [R0115-15-1012, 사용자 참여가 가능한 바둑 방송 및 기보 서비스 개발과 컨텐츠 생태계 조성을 위한 바둑 컨텐츠 마켓 플랫폼 개발]

참고문헌

[1] T Hirsimäki, “Gocam: Extracting Go game positions from photographs” 2005.2

[2] 강대현, 이윤구, “바둑판 선 검출 알고리즘” 2016.6, 803-806 (4pages)

[3] 이대규, 이윤구, “자동 바둑 기보 저장을 위한 바둑돌 인식 알고리즘” 2016.6, 809-812 (4pages)

[4] C Harris, M Stephens. “A combined corner and edge detector” 1988, 4th Alvey Vision Conference, Manchester, UK, pp. 147-151.

[5] JF Canny, “A computational approach to edge detection”, 1986, IEEE Trans. Pattern Anal. Machine Intell., vol. PAMI-8, pp. 679-698,