

**AI 기반컴퓨터비전 중간과제**

**과목명 AI기반컴퓨터비전**

**담당교수님 전병환교수님**

**제출일 20241105**

**전공 컴퓨터공학과**

**학번 202430026**

**이름 이준용**

1. **이미지가 주어졌을 때, Hough Transform기반 직선, 원검출 방법에 대한 일련의 과정을 이해하고 별도의 문서에 본인이 이해한 방법으로 정리하시오.**

Hough Transform은 이미지 내 여러 형태중에 특정 형태를 감지하는 기법으로, 주로 직선이나 원과 같은 기하학적 객체 검출에 사용된다. 이 기법은 특징을 공간 내에서 변환하여, 이미지에서 기하학적 객체가 지나는 픽셀들을 누적하여 특정 형태를 도출합니다.

**직선 검출 (Hough Line Transform)**

Hough Line Transform은 이미지에서 직선을 검출하는 방법으로, 이 방법의 원리는 이미지 공간에서 직선을 찾는 대신 매개변수 공간에서 직선 후보를 찾는 것입니다. 직선은 극좌표계 (polar coordinates)에서 표현되며, 이로 인해 변환이 단순화됩니다.

**직선의 표현**

이미지의 픽셀 좌표 (x, y)와 직선의 각도 θ를 이용해, 직선의 거리 ρ를 정의하여 직선을 나타낼 수 있습니다. 직선은 다음의 수식으로 표현됩니다.

여기서:

* : 직선이 원점에서부터의 거리
* : 직선과 x축이 이루는 각도

텍스트, 라인, 폰트, 도표이(가) 표시된 사진

자동 생성된 설명

**누적 배열 (Accumulator Array) 사용**

1. 이미지에서 edge를 검출한 후, edge pixel마다 ρ와 θ 값의 가능한 조합을 계산해 누적 배열 (accumulator array)에 값을 저장한다.
2. 누적 배열의 각 위치는 (ρ, θ) 값 쌍에 대응되며, 직선을 구성하는 edge pixel들이 해당 값에 선택.
3. 선택받은 수가 일정 기준을 초과하는 (ρ, θ) 값이 직선을 구성한다고 판단.

라인, 도표, 그래프이(가) 표시된 사진

자동 생성된 설명

**장점과 단점**

* **장점**: 직선을 검출하는 데 효과적이며 noise에 강함.
* **단점**: 많은 계산이 필요하며 매개변수 튜닝이 요구.

## 원 검출 (Hough Circle Transform)

Hough Circle Transform은 Hough Transform을 원에 맞게 응용한 방법이다. 원은 직선보다 매개변수가 더 많기 때문에 직선 검출보다 복잡하지만, 동일한 기본 원리입니다.

### 원의 표현

원은 중심 (a, b)와 반지름 r로 정의됩니다. 원의 방정식은

이 식을 통해 중심 좌표 (a, b)와 반지름 r을 찾을 수 있다.

도표, 텍스트, 원, 라인이(가) 표시된 사진

자동 생성된 설명

**누적 배열 (Accumulator Array) 사용**

1. 이미지의 edge pixel을 순회하며 가능한 원의 중심과 반지름 조합을 누적 배열에 저장.
2. 모든 조합에 대해 선택이 이루어지고, 특정 기준 이상의 선택 받은 수를 가진 pixel을 중심으로 하는 원을 검출.

도표, 라인, 스케치, 원이(가) 표시된 사진

자동 생성된 설명

**장점과 단점**

* **장점**: 원 검출에 매우 효과적이며, noise에 강함.
* **단점**: 직선 검출보다 많은 계산이 필요하며, 반지름 크기나 거리와 같은 여러 매개변수 튜닝이 필요.

Hough Transform은 이미지에서 기하학적 객체를 검출하는 강력한 기법으로, 특히 직선과 원 검출에 뛰어난 성능을 발휘합니다. 그러나 매개변수 조정이 필요하며, 원형 또는 선형 형태가 완벽하지 않은 객체에서는 성능이 제한될 수 있다.

**1. 7주차 강의자료에 주어진 code를 기반으로 본인이 해석한 방법으로 주석을 작성하시오.**

    def houghCircleTransform(self, edgeMap, min\_rad, max\_rad):

        """

        주어진 에지 맵을 기반으로 여러 반지름을 고려해 누적 배열(accumulator)을 채우고,

        누적 배열의 특정 기준을 초과하는 위치를 중심으로 원을 검출

        Args:

            edgeMap (\_type\_): edgeDetection을 사용해 이미지의 에지를 추출한 맵

            min\_rad (\_type\_): 검출할 원의 최소 반지름

            max\_rad (\_type\_): 검출할 원의 최대 반지름

        Returns:

            centers: 원의 중심과 반지름 정보가 포함된 좌표 배열

            accumulator: 누적 배열로, 각 좌표와 반지름에 대한 투표 결과가 포함

        """

        # edgeMap의 높이와 너비를 구함

        nY, nX = edgeMap.shape

        # 누적 배열 초기화: 높이, 너비, 반지름의 범위에 대한 3차원 배열 생성

        accumulator = np.zeros((nY, nX, max\_rad - min\_rad + 1), dtype=np.uint8)

        # 에지 픽셀 순회

        for y in range(nY):

            print(print(y))  # 진행 상황을 위한 y 값 출력

            for x in range(nX):

                # 에지 픽셀인지 확인 (0보다 큰 값이면 에지로 판단)

                if edgeMap[y, x] > 0:

                    # 최소 반지름에서 최대 반지름까지의 모든 반지름에 대해 순회

                    for r in range(min\_rad, max\_rad + 1):

                        # 각도를 0부터 359도까지 순회하며 원의 점 계산

                        for theta in range(360):

                            # 각도를 라디안으로 변환

                            radian = theta \* np.pi / 180.0

                            # 원의 x, y 좌표 계산

                            x\_ = int(x - r \* np.cos(radian))

                            y\_ = int(y - r \* np.sin(radian))

                            # 계산된 좌표가 이미지 크기 내에 있는지 확인

                            if (0 <= x\_ < nX) and (0 <= y\_ < nY):

                                # 누적 배열에서 해당 위치에 투표 추가

                                accumulator[y\_, x\_, r - min\_rad] += 1

        # 누적 배열에서 최대 투표 값의 80% 이상인 좌표를 추출하여 원으로 판단

        centers = np.argwhere(accumulator > np.max(accumulator) \* 0.8)

        # 검출된 원을 저장할 리스트 초기화

        circles = []

        cnt = 0

        for c in centers:

            # 중심 y, x와 반지름 r 정보 추출

            y, x, r = c

            print(cnt, ", ", y, x, r)  # 진행 상황 출력

            # 원의 중심과 반지름을 circles 리스트에 저장 (반지름은 min\_rad를 더해 보정)

            circles.append((x, y, r + min\_rad))

            cnt += 1

        # 검출된 원의 중심과 반지름 정보, 그리고 누적 배열 반환(원의 중심이 될 가능성이 높은 좌표에

        # 대한 결과가 포함된 3차원 배열)

        return centers, accumulator

**2. 코드 내 Hough Circle Transform의 결과를 centers 변수를 통하여 받는다. 이를 GUI 프로그램 내 원본 이미지에서 opencv라이브러리를 활용하여 circle을 마킹하도록 코드를 작성하시오. 작성 후 변화된 코드 부분과 결과 화면을 캡처하시오.**

Image\_processing.py

    def circleDetection(self, img):

        edgeMap = self.edgeDetection(img)

        edge = (edgeMap > 150) \* 255  # 임계값 적용

        # Hough Circle Transform을 이용해 원 검출

        centers, accumulator = self.houghCircleTransform(edge, 20, 23)

        print(np.min(accumulator[:, :, 0]), np.max(accumulator[:, :, 0]))

        # 검출된 원을 원본 이미지에 그리기

        for center in centers:

            y, x, r = center

            # 원과 중심을 그림

            cv.circle(self.sourceImg, (x, y), r + 20, (255, 0, 0), 2)  # 원 그리기

            cv.circle(self.sourceImg, (x, y), 2, (0, 255, 0), 3)       # 중심점 그리기

        # 원이 그려진 이미지를 대상 이미지로 업데이트

        self.targetImg = self.sourceImg

GUI.py

    def circleDetection(self):

        img = self.ip.sourceImg

        gray\_img = self.ip.toGrayScale(img)  # 그레이스케일로 변환

        self.ip.circleDetection(gray\_img)    # 원 검출 및 그리기 수행

        self.img\_target = self.ip.sourceImg  # 원이 표시된 이미지를 대상 이미지로 업데이트

        self.apply()                         # 결과를 GUI에 표시

        messagebox.showinfo("Info", "원 검출이 완료되었습니다")

결과 출력 사진

스크린샷, 텍스트이(가) 표시된 사진

자동 생성된 설명

1. **0~2까지 과제를 하나의 문서로 작성 후 마지막에 느낀 점 (5줄 이내) 작성 후 pdf로 제출하시오.**

이번 과제를 통해 Hough Transform을 기반으로 한 직선 및 원 검출 기법의 원리와 응용 방법에 대해 깊이 이해할 수 있었습니다. 특히, OpenCV를 활용해봤었지만 GUI 프로그램 에서 원본 이미지에 검출 결과를 시각적으로 표시하면서 이미지 처리해본 경험은 이후에 최근 산업계 특성상 AI기술과 IT분야와의 융복합적으로 적용하고 개발할 때 중요한 경험이 될 것 같습니다. 감사합니다.