**컴퓨터 비전**

Assign#2. ERP 영상의 Retification

|  |
| --- |
| 스 쿨 : ETRI 스쿨 |
| 이 름 : 김 기 범 |
| 학 번 : 02321078 |

1. 서론

과제의 주된 목표는 주어진 ERP (Equirectangular Projection) 이미지를 정면 뷰와 탑 뷰 영상으로 변환하는 프로그램을 구현하는 것입니다. ERP 이미지는 360도 파노라마 이미지를 2D 이미지로 표현하는 한 방법으로, 3D 공간의 모든 방향을 하나의 2D 이미지로 표현한다. 하지만 이러한 이미지는 직관적인 해석이 어려울 수 있으며, 특정 방향 또는 시점에서의 뷰를 얻기 위해 다른 투영 방법으로 변환하는 것이 필요합니다.

본 과제에서는 이러한 ERP 이미지를 정면 뷰와 탑 뷰로 변환하는 알고리즘을 구현합니다. 정면 뷰는 사용자가 특정 방향을 바라보고 있을 때의 시점을, 탑 뷰는 사용자가 공간의 최상단에서 바라보는 시점을 나타냅니다. 이러한 변환을 통해 ERP 이미지의 특정 부분을 더 자세하고 직관적으로 관찰할 수 있습니다.

1. 본론
2. 주요 코드
3. ERP 이미지의 정면 뷰

def erp2rect(src, theta, hfov, vfov):

# 초점 길이 계산

f = src.shape[1] / (2 \* np.pi)

# 평면 이미지의 크기 계산

front\_view\_image\_rows = int(2 \* f \* np.tan(vfov / 2) + 0.5)

front\_view\_image\_cols = int(2 \* f \* np.tan(hfov / 2) + 0.5)

# 평면 이미지 초기화

front\_view\_image = np.zeros((front\_view\_image\_rows, front\_view\_image\_cols, 3), dtype=np.uint8)

front\_view\_image\_cx = front\_view\_image\_cols / 2

front\_view\_image\_cy = front\_view\_image\_rows / 2

for x in range(front\_view\_image\_cols):

xth = np.arctan((x - front\_view\_image\_cx) / f)

# 원본 이미지에서의 x 좌표 계산

src\_x = int((xth + theta) \* src.shape[0] / np.pi + 0.5) % src.shape[1]

yf = f / np.cos(xth)

for y in range(front\_view\_image\_rows):

yth = np.arctan((y - front\_view\_image\_cy) / yf)

# 원본 이미지에서의 y 좌표 계산

src\_y = int(yth \* src.shape[0] / np.pi + src.shape[0] / 2 + 0.5)

src\_y = max(0, min(src\_y, src.shape[0] - 1)) # y 좌표를 이미지 내로 제한

front\_view\_image[y, x] = src[src\_y, src\_x]

return front\_view\_image

erp2rect 함수는 주어진 Equirectangular Projection (ERP) 이미지, 즉 360도 구형 투영 이미지를 직선 투영 이미지로 변환하는 함수입니다. 함수는 다음과 같은 순서로 작동합니다.

* 1. 초점 길이 계산: 함수 내에서 f라는 변수는 ERP 이미지의 너비를 기반으로 초점 길이를 계산합니다. 이 값은 변환 과정에서 중요한 역할을 합니다.
  2. 결과 이미지 크기 설정: 주어진 시야각(hfov, vfov)을 기반으로 변환될 직선 투영 이미지의 크기를 계산합니다. 그리고 front\_view\_image라는 이름의 빈 이미지를 만들고, 변환된 픽셀 값을 저장할 공간을 준비합니다.
  3. 각 픽셀의 변환: 함수는 직선 투영 이미지의 모든 픽셀을 순회하며 해당 픽셀의 색상 값을 결정하기 위해 원본 ERP 이미지의 어느 부분을 참조해야 할지 계산합니다. 이를 위해, 각 픽셀의 위치에 따른 수평 및 수직 각도를 계산합니다(xth, yth). 이 각도는 원본 ERP 이미지에서의 좌표(src\_x, src\_y)를 결정하는 데 사용됩니다. 계산된 원본 이미지의 좌표를 사용하여 픽셀 값을 가져와 front\_view\_image에 할당합니다.
  4. 결과 반환: 모든 픽셀에 대한 변환이 완료되면, 변환된 front\_view\_image를 반환합니다.

1. ERP 이미지의 탑 뷰

import numpy as np

import cv2

import matplotlib.pyplot as plt

# D가 무한대가 되는 경우를 처리하기 위해 업데이트된 함수

def erp2topdown(src, hfov, vfov):

# ERP 이미지의 가로, 세로 해상도

W, H = src.shape[1], src.shape[0]

# 초점 거리(구의 반지름) 계산

f = W / (2 \* np.pi)

# 탑다운 이미지 해상도 계산

W\_prime = int(2 \* f \* np.tan(hfov / 2) + 0.5)

H\_prime = int(2 \* f \* np.tan(vfov / 2) + 0.5)

# 탑다운 이미지의 중심 좌표 계산

cx\_prime, cy\_prime = W\_prime // 2, H\_prime // 2

# 결과 이미지를 저장할 배열 초기화

top\_view\_image = np.zeros((H\_prime, W\_prime, 3), dtype=np.uint8)

# ERP 이미지의 중심 좌표

cx, cy = W // 2, H // 2

# D의 최대 값 설정하여 무한대 값을 방지

max\_D = W\_prime

# 모든 픽셀에 대해 반복

for y in range(H):

# 수직 각도 계산

phi = (y - cy) \* np.pi / H

# 중심부터의 거리 D 계산

D = f / np.tan(phi)

# D가 max\_D보다 큰 경우 계산 생략

if abs(D) > max\_D:

continue

for x in range(W):

# 수평 각도 계산

theta = (x - cx) \* 2 \* np.pi / W

# 탑다운 이미지에서의 픽셀 좌표 계산

x\_prime = int(cx\_prime + D \* np.sin(theta))

y\_prime = int(cy\_prime - D \* np.cos(theta))

# 좌표를 이미지 크기 내로 제한

x\_prime = max(0, min(x\_prime, W\_prime - 1))

y\_prime = max(0, min(y\_prime, H\_prime - 1))

# 탑다운 이미지에 픽셀 값 할당

top\_view\_image[y\_prime, x\_prime] = src[y, x]

return top\_view\_image

# ERP 이미지 파일 경로

file = "./erp.png"

# ERP 이미지 불러오기

erp\_image = cv2.imread(file)

# 시야각을 라디안으로 변환

hfov\_topdown, vfov\_topdown = np.deg2rad(120), np.deg2rad(120)

# ERP 이미지를 탑다운 뷰로 변환

topdown\_view = erp2topdown(erp\_image, hfov\_topdown, vfov\_topdown)

# 변환된 탑다운 이미지 출력

plt.imshow(topdown\_view)

plt.title('Top-Down View')

plt.axis('off') # 축 정보 숨기기

plt.show()

erp2topdown 함수는 주어진 Equirectangular Projection (ERP) 이미지를 탑다운(top-down) 뷰로 변환하는 함수입니다. 함수는 다음과 같은 순서로 작동합니다.

1. 필요한 변수 설정: 초점 거리(f), 즉 구의 반지름을 계산합니다. 이는 ERP 이미지의 너비를 기반으로 합니다. 주어진 시야각(hfov, vfov)을 기반으로 탑다운 이미지의 크기(W\_prime, H\_prime)를 계산합니다. 그리고 이를 바탕으로 탑다운 이미지의 중심 좌표(cx\_prime, cy\_prime)를 결정합니다.
2. 결과 이미지 초기화: 변환될 이미지인 top\_view\_image를 초기화합니다. 이 배열은 최종적인 탑다운 뷰를 저장하기 위한 공간입니다.
3. 각 픽셀에 대한 변환 계산: 함수는 ERP 이미지의 모든 픽셀을 순회하며 해당 픽셀의 색상 값을 결정하기 위해 탑다운 이미지의 어느 위치에 배치될지 계산합니다. 이를 위해, 각 픽셀의 위치에 따른 수평 및 수직 각도(theta, phi)를 계산합니다. 이 각도들은 탑다운 이미지에서의 좌표(x\_prime, y\_prime)를 결정하는 데 사용됩니다. 계산된 좌표에 원본 픽셀 값을 할당합니다.
4. 결과 반환: 모든 픽셀에 대한 변환이 완료되면, 변환된 dst를 반환합니다.

위의 함수는 수업 자료를 바탕으로 하여 작성되었습니다, 다만, ERP 이미지는 전체 360도 환경을 나타내기 때문에, 이를 평면 이미지로 변환할 때 일부 픽셀들은 평면 이미지의 여러 위치에 매핑될 수 있습니다. 반면, 일부 평면 이미지의 픽셀은 ERP 이미지에서 대응되는 픽셀이 없을 수 있습니다. 따라서, 이러한 픽셀들은 빈값으로 남아서 검은색의 픽셀로 표현됩니다.

이 문제를 해결하기 위해 보간법을 사용할 수 있지만, ERP 이미지의 구조와 특성 상 보간법을 직접 적용하기는 쉽지 않습니다. 따라서, 위의 코드에서 평면 이미지의 각 픽셀을 ERP 이미지에 역투영하여 해당 픽셀의 색상 값을 얻어오는 방식으로 수정하여 (ㄷ)으로 사용했습니다. 이 방법을 통해 ERP 이미지의 픽셀과 평면 이미지의 픽셀을 보다 정확하게 매핑할 수 있으며, 결과적으로 빈값 없이 이미지 변환을 완료할 수 있습니다.

1. 평면 이미지에서 ERP 이미지로 투영하고 보간법을 적용

def bilinear\_interpolation(img, x, y):

x0 = int(x)

y0 = int(y)

x1 = min(x0 + 1, img.shape[1] - 1)

y1 = min(y0 + 1, img.shape[0] - 1)

P00 = img[y0, x0]

P01 = img[y1, x0]

P10 = img[y0, x1]

P11 = img[y1, x1]

dx = x - x0

dy = y - y0

interpolated\_value = (1 - dx) \* (1 - dy) \* P00 + \

dx \* (1 - dy) \* P10 + \

(1 - dx) \* dy \* P01 + \

dx \* dy \* P11

return interpolated\_value.astype(np.uint8)

def erp2topdown(src, hfov, vfov):

# ERP 이미지의 가로와 세로 크기를 가져옵니다.

W, H = src.shape[1], src.shape[0]

# ERP 이미지를 기반으로 초점 거리(구의 반지름)을 계산합니다.

f = W / (2 \* np.pi)

# 최종 top-down 이미지의 가로와 세로 크기를 계산합니다.

W\_prime = int(2 \* f \* np.tan(hfov / 2) + 0.5)

H\_prime = int(2 \* f \* np.tan(vfov / 2) + 0.5)

# top-down 이미지의 중심 좌표를 계산합니다.

cx\_prime, cy\_prime = W\_prime // 2, H\_prime // 2

# ERP 이미지의 중심 좌표를 계산합니다.

cx, cy = W // 2, H // 2

# top-down 이미지를 저장할 빈 배열을 생성합니다.

top\_view\_image = np.zeros((H\_prime, W\_prime, 3), dtype=np.uint8)

# 모든 픽셀에 대해 반복을 수행합니다.

for y\_prime in range(H\_prime):

for x\_prime in range(W\_prime):

# 중심에서 현재 픽셀까지의 거리를 계산합니다.

D = np.sqrt((x\_prime - cx\_prime)\*\*2 + (y\_prime - cy\_prime)\*\*2)

# 거리가 0인 경우, 즉 중심인 경우 ERP 이미지의 중심 픽셀 값을 사용합니다.

if D == 0:

top\_view\_image[y\_prime, x\_prime] = src[H-1, 0]

continue

# theta 각도를 계산합니다.

theta = np.arctan2(x\_prime - cx\_prime, cy\_prime - y\_prime)

# phi 각도를 계산합니다.

phi = np.arctan(f / D)

# ERP 이미지에서의 좌표를 계산합니다.

erp\_x = theta \* W / (2 \* np.pi) + cx

erp\_y = phi \* H / np.pi + cy

# 좌표의 범위를 이미지 크기 내로 제한합니다.

erp\_x = np.clip(erp\_x, 0, W - 1)

erp\_y = np.clip(erp\_y, 0, H - 1)

# bilinear\_interpolation 함수를 사용하여 ERP 이미지에서 해당 좌표의 픽셀 값을 가져와 top-down 이미지에 저장합니다.

top\_view\_image[y\_prime, x\_prime] = bilinear\_interpolation(src, erp\_x, erp\_y)

# 최종 top-down 이미지를 반환합니다.

return top\_view\_image

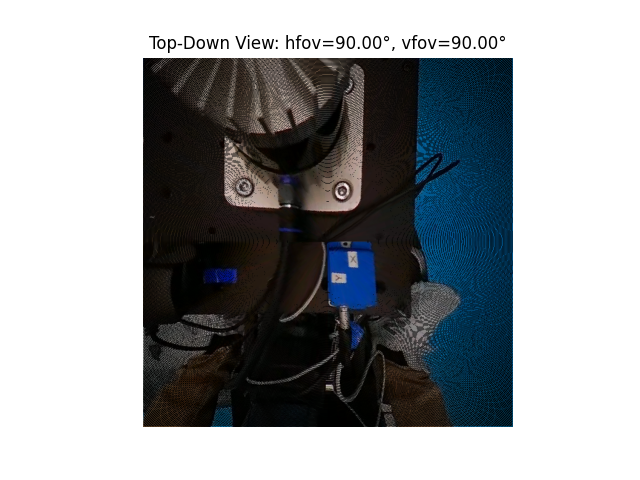
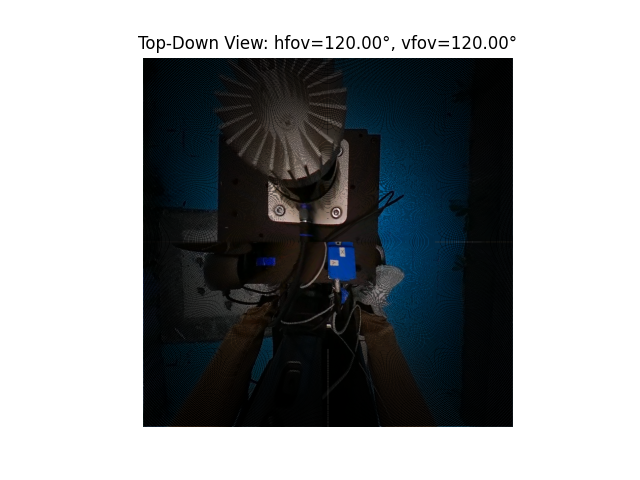
bilinear\_interpolation 함수는 평면 이미지의 픽셀을 ERP 이미지에 투영해서 얻은 실수 좌표값을 기반으로 하여, 해당 좌표 주변 4픽셀의 값을 기반으로 보간법을 실시하여 얻어낸 이미지 값을 평면이미지의 픽셀 값으로 리턴하는 함수입니다.

수정된 erp2topdown 함수는 (ㄴ)에서 있었던 문제를 해결하고자 수정된 코드입니다. 기존의 수식을 바탕으로 하여, 역으로 Top View Image의 픽셀이 ERP 이미지의 좌표로 매핑되도록 작성하였습니다. 함수는 다음과 같은 순서로 작동합니다.

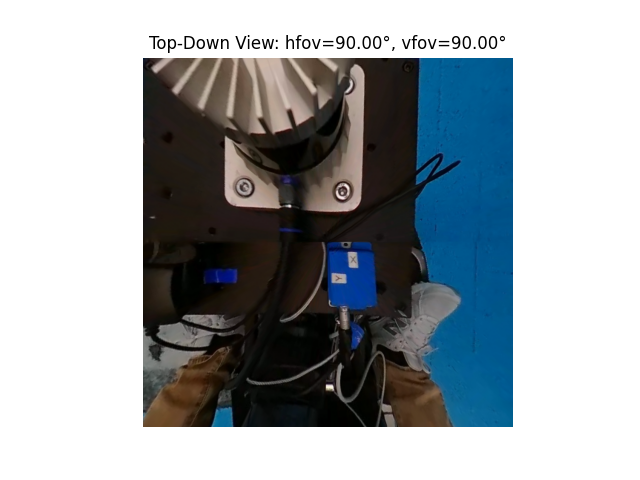
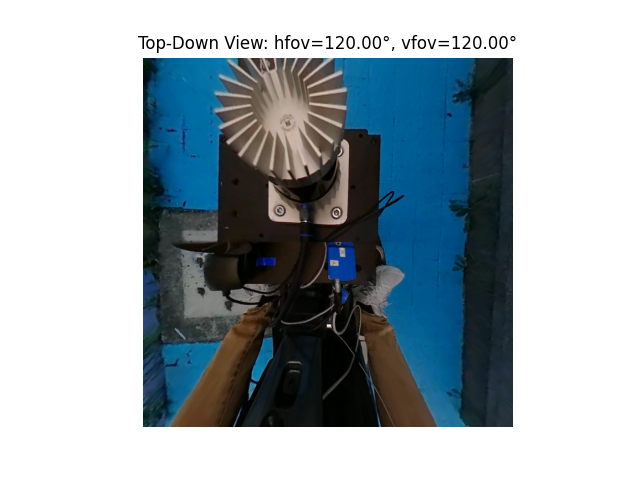
1. 변수 초기화: ERP 이미지의 가로(W)와 세로(H) 크기를 가져옵니다. ERP 이미지의 너비를 기반으로하여 초점 거리(f) 즉, 구의 반지름을 계산합니다. 주어진 수평(hfov) 및 수직(vfov) 시야각을 기반으로 Top View Image의 크기(W\_prime와 H\_prime)를 결정합니다.
2. 중심 좌표 계산: ERP 이미지와 Top View Image의 중심 좌표(cx, cy 및 cx\_prime, cy\_prime)를 계산합니다.
3. 탑다운 이미지 생성: 변환된 픽셀 값을 저장할 top\_view\_image를 만듭니다.
4. 픽셀 변환: top\_view\_image의 각 픽셀에 대해서, 해당 픽셀의 위치를 ERP 이미지의 좌표로 변환합니다. 이때, 구형 좌표계에서의 각도(theta와 phi)를 계산하여 이를 ERP 이미지의 좌표(erp\_x, erp\_y)로 변환합니다.
5. 보간법 적용: 계산된 ERP 좌표는 실수 값이므로, 이를 실제 픽셀 값으로 변환하기 위해 양선형 보간법(bilinear interpolation)을 사용합니다.
6. 결과 반환: 모든 변환이 완료되면, 완성된 Top View Image를 반환합니다.
7. 결과 캡쳐
8. ERP 이미지의 정면 뷰



1. ERP 이미지의 탑 뷰



1. 평면 이미지에서 ERP 이미지로 투영하고 보간법을 적용



코드를 테스트 한 결과, 체커보드의 움직임에 따라 잘 추적하는 모습을 보입니다.

1. 결론

이번 과제를 통해서 Equirectangular Projection (ERP) 이미지를 정면 뷰와 탑 뷰로 변환하는 방법을 배웠습니다. ERP 이미지는 360도 파노라마 이미지를 2D 이미지로 표현하는 효과적인 방법이지만, 특정 방향 또는 시점에서의 뷰를 얻기 위해서는 다른 투영 방법으로 변환하는 과정이 필요하다는 것을 확인하였습니다.

구현한 알고리즘을 통해 ERP 이미지를 정면 뷰와 탑 뷰로 성공적으로 변환할 수 있었습니다. 특히 양선형 보간법을 활용하여 ERP 이미지의 실수 좌표를 정확하게 픽셀 값으로 변환하는 과정에서 빈 값을 최소화하는 것에 성공하였습니다. 이로 인해 변환된 이미지의 질을 크게 향상시킬 수 있었습니다.

마지막으로, 이 연구를 통해 고도의 이미지 처리 기술을 사용하여 복잡한 문제를 해결할 수 있음을 경험하였습니다. 앞으로도 이러한 기술을 다양한 문제에 적용하여 연구 및 개발을 계속해서 진행할 계획입니다.

1. Github 주소

<https://github.com/chroion/computer-vision/blob/main/HW%231/assign%231.py>