

문)

다음의 사진을 이용하여 반은 정상인데 반은 사자인 얼굴을 어색하지 않게 만들어보자.

사용파일 : man_face, lion_face



<힌트>



두 영상을 절반씩 가져다가 새로운 영상으로 단순히 합성하면 그냥 두 개의 영상을 이어 붙인 것처럼 보이기 때문에 어색하다.

따라서 수업시간에 배운 알파블렌딩을 이용하여 두 영상이 만나는 지점의 일정 부분을 서서히 변하게 하면서 자연스럽게 두 개의 얼굴이 하나의 얼굴로 합성할 수 있도록 해보자.

영상의 15%를 알파 블렌딩의 범위로 지정합니다.

답)

1. X축 - 그냥 두 영상을 합성	2. X축 - 알파 블렌딩 적용
	
<pre>import cv2 import numpy as np # 영상의 15%를 알파 블렌딩의 범위로 지정 alpha_width_rate = 15 # 합성할 두 영상 읽기 img_face = cv2.imread('./img/man_face.jpg') img_skull = cv2.imread('./img/lion_face.jpg') # 입력 영상과 같은 크기의 결과 영상 준비 img_comp = np.zeros_like(img_face) # 연산에 필요한 좌표 계산 height, width = img_face.shape[:2] middle = width//2 alpha_width = width * alpha_width_rate // 100 # 영상의 중앙 좌표 start = middle - alpha_width//2 # 알파 블렌딩 범위 step = 100/alpha_width # 알파 블렌딩 시작 지점 # 입력 영상의 절반씩 복사해서 결과 영상에 합성 img_comp[:, :middle, :] = img_face[:, :middle, :].copy() img_comp[:, middle:, :] = img_skull[:, middle:, :].copy() cv2.imshow('half', img_comp) # 알파 값을 바꾸면서 알파 블렌딩 적용 for i in range(alpha_width+1): alpha = (100 - step * i) / 100 # 증감 간격에 따른 알파 값 (1~0) beta = 1 - alpha # 베타 값 (0~1) # 알파 블렌딩 적용 img_comp[:, start+i] = img_face[:, start+i] * \ alpha + img_skull[:, start+i] * beta print(i, alpha, beta) cv2.imshow('half skull', img_comp) cv2.waitKey() cv2.destroyAllWindows()</pre>	<pre>import cv2 import numpy as np # 영상의 15%를 알파 블렌딩의 범위로 지정 alpha_width_rate = 15 # 합성할 두 영상 읽기 img_face = cv2.imread('./img/man_face.jpg') img_skull = cv2.imread('./img/lion_face.jpg') # 입력 영상과 같은 크기의 결과 영상 준비 img_comp = np.zeros_like(img_face) # 연산에 필요한 좌표 계산 height, width = img_face.shape[:2] middle = width//2 alpha_width = width * alpha_width_rate // 100 # 영상의 중앙 좌표 start = middle - alpha_width//2 # 알파 블렌딩 범위 step = 100/alpha_width # 알파 블렌딩 시작 지점 # 입력 영상의 절반씩 복사해서 결과 영상에 합성 img_comp[:, :middle, :] = img_face[:, :middle, :].copy() img_comp[:, middle:, :] = img_skull[:, middle:, :].copy() cv2.imshow('half', img_comp) # 알파 값을 바꾸면서 알파 블렌딩 적용 for i in range(alpha_width+1): alpha = (100 - step * i) / 100 # 증감 간격에 따른 알파 값 (1~0) beta = 1 - alpha # 베타 값 (0~1) # 알파 블렌딩 적용 img_comp[:, start+i] = img_face[:, start+i] * \ alpha + img_skull[:, start+i] * beta print(i, alpha, beta) cv2.imshow('half skull', img_comp) cv2.waitKey() cv2.destroyAllWindows()</pre>

3. Y축 - 그냥 두 영상을 합성	4. Y축 - 알파 블렌딩 적용
	
<pre>import cv2 import numpy as np # 영상의 15%를 알파 블렌딩의 범위로 지정 alpha_height_rate = 15 # 합성할 두 영상 읽기 img_face = cv2.imread('../img/man_face.jpg') img_skull = cv2.imread('../img/lion_face.jpg') # 입력 영상과 같은 크기의 결과 영상 준비 img_comp = np.zeros_like(img_face) # 연산에 필요한 좌표 계산 height, width = img_face.shape[:2] middle = height//2 # 영상의 중앙 좌표 alpha_height = height * alpha_height_rate // 100 # 알파 블렌딩 범위 start = middle - alpha_height//2 # 알파 블렌딩 시작 지정 step = 100/alpha_height # 알파 값 간격 # 입력 영상의 절반씩 복사해서 결과 영상에 합성 img_comp[middle, :, :] = img_face[middle, :, :].copy() img_comp[middle, :, :] = img_skull[middle, :, :].copy() cv2.imshow('half', img_comp) # 알파 값을 바꾸면서 알파 블렌딩 적용 for i in range(alpha_height+1): alpha = (100 - step * i) / 100 # 증감 간격에 따른 알파 값 (1~0) beta = 1 - alpha # 베타 값 (0~1) # 알파 블렌딩 적용 img_comp[start+i, :] = img_face[start+i, :] * W alpha + img_skull[start+i, :] * beta print(i, alpha, beta) cv2.imshow('half skull', img_comp) cv2.waitKey() cv2.destroyAllWindows()</pre>	<pre>import cv2 import numpy as np # 영상의 15%를 알파 블렌딩의 범위로 지정 alpha_height_rate = 15 # 합성할 두 영상 읽기 img_face = cv2.imread('../img/man_face.jpg') img_skull = cv2.imread('../img/lion_face.jpg') # 입력 영상과 같은 크기의 결과 영상 준비 img_comp = np.zeros_like(img_face) # 연산에 필요한 좌표 계산 height, width = img_face.shape[:2] middle = height//2 # 영상의 중앙 좌표 alpha_height = height * alpha_height_rate // 100 # 알파 블렌딩 범위 start = middle - alpha_height//2 # 알파 블렌딩 시작 지정 step = 100/alpha_height # 알파 값 간격 # 입력 영상의 절반씩 복사해서 결과 영상에 합성 img_comp[middle, :, :] = img_face[middle, :, :].copy() img_comp[middle, :, :] = img_skull[middle, :, :].copy() cv2.imshow('half', img_comp) # 알파 값을 바꾸면서 알파 블렌딩 적용 for i in range(alpha_height+1): alpha = (100 - step * i) / 100 # 증감 간격에 따른 알파 값 (1~0) beta = 1 - alpha # 베타 값 (0~1) # 알파 블렌딩 적용 img_comp[start+i, :] = img_face[start+i, :] * W alpha + img_skull[start+i, :] * beta print(i, alpha, beta) cv2.imshow('half skull', img_comp) cv2.waitKey() cv2.destroyAllWindows()</pre>