드론 에이전트 스카우팅 시스템

3차시 (3시간)

강의 개요

1. 드론 에이전트 시스템 아키텍처 (45분)
2. 비전 인식 및 분석 시스템 (45분)
3. 스카우팅 기능 구현 (45분)
4. 실전 응용 및 실습 (45분)

1. 드론 에이전트 시스템 아키텍처

에이전트 시스템 구조

- 드론 컨트롤러 (TelloController)
- 비전 처리 시스템 (VisionSystem)
- 명령 처리기 (CommandProcessor)
- 상태 관리자 (StateManager)

드론 컨트롤러 구현

```
class TelloController:
    def __init__(self):
        self.tello = Tello()
        self.frame_reader = None
        self.is_streaming = False

    def connect(self):
        print("드론에 연결 중...")
        self.tello.connect()
        battery = self.tello.get_battery()
        print(f"배터리 잔량: {battery}%")

        if battery < 20:
            raise Exception("배터리 부족")
```

비전 처리 시스템

```
class VisionSystem:
    def __init__(self):
        self.frame_queue = Queue(maxsize=10)
        self.client = OpenAI()

    def analyze_image(self, image_path: str) -> str:
        with open(image_path, "rb") as image_file:
            base64_image = base64.b64encode(
                image_file.read()
            ).decode('utf-8')

        response = self.client.chat.completions.create(
            model="gpt-4o-mini",
            messages=[{
                "role": "user",
                "content": [
                    {"type": "text", "text": "이미지 분석"},
                    {"type": "image_url",
                     "image_url": {
                        "url": f"{base64_image}"
                     }}
                ]
            }]
        )
        return response.choices[0].message.content
```

2. 비전 인식 및 분석 시스템

이미지 캡처 시스템

- 실시간 비디오 스트리밍
- 프레임 캡처 및 저장
- 이미지 전처리
- 메모리 관리

GPT Vision 통합

- API 설정 및 초기화
- 이미지 인코딩
- 프롬프트 엔지니어링
- 응답 처리

분석 결과 처리

- 텍스트 분석
- 객체 인식
- 상황 판단
- 행동 결정

3. 스카우팅 기능 구현

자동 스캔 시스템

```
def scan_surroundings(self):
    try:
        images = []
        for i in range(4):
            filename = self.take_photo()
            images.append(filename)
            self.tello.rotate_clockwise(90) ## 90도마다 돌아가며 사진찍음
            time.sleep(2)
        analyses = []
        for img in images:
            analysis = self.analyze_image(img)
            analyses.append(analysis)
        return analyses
```

파노라마 생성

```python
def create_panorama(self):
    """파노라마 이미지 생성"""
    images = []
    for i in range(4):
        frame = self.get_frame()
        images.append(frame)
        self.tello.rotate_clockwise(90)
        time.sleep(2)

    stitcher = cv2.Stitcher.create()
    status, panorama = stitcher.stitch(images)

    if status == cv2.Stitcher_OK:
        cv2.imwrite('panorama.jpg', panorama)
        return 'panorama.jpg'
    else:
        raise Exception("파노라마 생성 실패")
```

웹 인터페이스 통합

```python
@app.route('/scan', methods=['POST'])
def scan_surroundings():
    try:
        analyses = controller.scan_surroundings()
        return jsonify({
            "status": "success",
            "analyses": analyses
        })
    except Exception as e:
        return jsonify({
            "status": "error",
            "message": str(e)
        })
```

4. 실전 응용 및 실습

실습 프로젝트

1. 자동 순찰 시스템 구현
2. 객체 추적 기능 추가
3. 상황 보고 시스템 개발
4. 비상 대응 로직 구현

고려사항

- 레이턴시
- 네트워크 품질
- 충돌 방지
- 컴퓨팅 성능

성능 최적화

- 메모리 관리
- 스트리밍 최적화
- API 호출 최소화
- 응답 시간 개선

참고 자료

API 문서

- DJITelloPy: https://djitellopy.readthedocs.io/
- OpenAI Vision: https://platform.openai.com/docs/guides/vision
- OpenCV: https://docs.opencv.org/
- YOLOv8: https://docs.ultralytics.com/

예제 코드

- tello-scan-surroundings.py
- tello-webui.py
- voice-control-tello-gemini.py