

지능형 휴머노이드

“ 어플리케이션 사례를 통한
개발 접근 방법”

이재성

1. 개발 환경 구축

1) 운영체제 및 영상처리 라이브러리 ,
프로그래밍 언어 설치

- Raspbian 설치
- OpenCV 설치
- 프로그래밍 언어 (Python) 선택 , 설치

2. 기본 드라이버 작성 , 테스트

2) 하드웨어 모듈 기본 구동

- 카메라 동작 체크 ([Picam@Python_OpenCV](#))
- 동영상 캡처 및 처리 이미지 획득

```
import picamera  
from picamera.array import PiRGBArray  
from picamera import PiCamera
```

2. 기본 드라이버 작성 , 테스트

2) 하드웨어 모듈 기본 구동

```
MAX_SCREEN_CAPTURE_WIDTH=400  
MAX_SCREEN_CAPTURE_HEIGHT=400  
cam=PiCamera()  
cam.resolution=(MAX_SCREEN_CAPTURE_WIDTH,MAX_  
SCREEN_CAPTURE_HEIGHT)  
cam.framerate=32  
raw_cap=PiRGBArray(cam, size=(MAX_SCREEN_CAPTU  
RE_WIDTH, MAX_SCREEN_CAPTURE_HEIGHT))
```

2. 기본 드라이버 작성 , 테스트

2) 하드웨어 모듈 기본 구동

```
for frame in cam.capture_continuous(raw_cap, format='bgr',
    use_video_port=True):
    _img_=frame.array
    _img_=cv2.GaussianBlur(_img_,(3,3),2)
    _img_=cv2.dilate(_img_, np.ones((5,5),np.uint8))

    img_hsv=cv2.cvtColor(_img_, cv2.COLOR_BGR2HSV)
```

2. 기본 드라이버 작성 , 테스트

2) 하드웨어 모듈 기본 구동

- 라즈베리파이보드 (호스트),
제어보드 (디바이스) 간 통신 체크

```
import serial
```

```
def serialport_init():  
    print 'serialport_init'  
    global ser  
    #ser=serial.Serial('/dev/ttyUSB0', 4800)  
    ser=serial.Serial('/dev/ttyAMA0', 4800)  
    print(ser)
```

2. 기본 드라이버 작성 , 테스트

2) 하드웨어 모듈 기본 구동

- 라즈베리파이보드 (호스트),
제어보드 (디바이스) 간 통신 체크

```
def send_cmdcode(code):  
    global ser  
    ser.write(code)  
  
if (ser.inWaiting() != 0):  
    abyte = ser.read()  
    remocon_value = int(abyte)  
    print 'remocon=', remocon_value
```

2. 기본 드라이버 작성 , 테스트

2) 하드웨어 모듈 기본 구동

- 라즈베리파이보드 (호스트),
제어보드 (디바이스) 간 통신 체크

ERX 4800, RX_VAR, NO_RX_LABEL

ETX 4800, TX_VAR

3. 어플리케이션 설계

1) 동작 (모션) 처리

- 호스트 , 디바이스 간 통신 설계
 - 호스트에서 모듈화 된 동작 코드를 전송 시 작동 여부 체크
 - 디바이스에서 동작 처리 후 결과 값을 호스트에 전송
 - 이후 호스트는 유효값으로 재 처리 여부를 판별

3. 어플리케이션 설계

1) 동작 (모션) 처리

- 경기 운영 방식 분석 및 행동 패턴 설계
 - 로봇의 기본 동작 체크 및 , 영점 조정
 - 효율적 행동으로 골 (GOAL) 에 이르는 방법을 단순화 , 체계화
 - 행동을 위한 동작 세분화 , 모듈화
 - 보정이 가능하도록 행동 패턴 설계

3. 어플리케이션 설계

2) 영상처리

- 카메라 영상인식

- 각 대상 목표물 (오브젝트) 을 인지 하기 위한 색 , 모양 등의 특성을 정의
- HSV 색 좌표계를 사용
- 연속적으로 획득한 동영상 프레임에서 정의된 오브젝트 인식
- 오브젝트 인식 테스트 - 색 변화 범주 및 인식 거리 등을 체크

3. 어플리케이션 설계

3) 통합 처리

- 영상인식에 의한 목표치 산출 및 해당 행동패턴 처리
- 알고리즘 수정, 보완

4. 테스트 및 디버깅

- 1) 경기률을 단순화 , 단계별로 설정
- 2) 각 단계별 단위 테스트 (UNIT TEST)
 - 테스트 결과에 따른 피드백 , 디버깅 , 알고리즘 수정 , 보완
 - 동작 , 영상 처리 설계 시 병렬 진행
(※ 테스트 영상물을 녹화 후 모니터링 , 분석)
- 3) 통합 테스트
 - 단위 테스트 검증 (신뢰성 , 효율성) 후 실제 경기률 적용에 의한 전체 테스트 , 디버깅 , 알고리즘 수정 , 보완

감사합니다

Q&A