공학석사 학위 논문

소형굴착기의 원격 조종을 위한 물체 거리 측정 및 자동 굴착 시스템에 관한 연구

A Study on the Measurement of Object Distance and Automatic Excavation System for Remote Control of Small Excavators

울산대학교 대학원
건설기계공학과
손 태 곤

소형굴착기의 원격 조종을 위한 물체 거리 측정 및 자동 굴착 시스템에 관한 연구

지 도 교 수　　안 경 관

이 논문을 공학석사 학위 논문으로 제출함

2021년　2월

울산대학교 대학원
건설기계공학과
손 태 곤

손태곤의 공학석사 학위 논문을 인준함

심사위원 이 병 룡 인

심사위원 하 철 근 인

심사위원 안 경 관 인

울산대학교 대학원

2021년 2월

소형굴착기의 원격 조종을 위한 물체 거리 측정 및 자동 굴착 시스템에 관한 연구

손 태 곤

울산대학교 대학원 건설기계공학과

국문요약

스마트 컨스트럭션이란, 일반적인 건설 기술에 4차 산업혁명의 신기술을 도입하여 생산성 향상 및 업무 효율을 증진하는 것을 말한다. 스마트 컨스트럭션에 많이 적용되는 기술은 환경인식과 무인자율 작업이 있다. 많은 연구에서 환경인식을 하기 위해 라이다, 초음파센서, 레이더, 카메라 센서를 사용한다. 라이다, 초음파센서, 레이더는 사물의 거리 정보를 정확히 측정할 수 있는 장점이 있고, 카메라는 거리 정보의 정확성 보다는 사물을 직접 인지 할 수 있는 장점으로 많이 사용했었다. 하지만 무인자율 작업의 필요성이 증진됨에 따라 원격에서 작업자가 굴착기를 주변의 환경을 인식하고, 사물을 인지할 필요성이 대두되었고, 카메라를 통한 거리 측정이 필요해졌다. 본 논문에서는 원격에서 굴착기를 조종하기 위해서 설치하는 카메라를 이용하여 거리를 측정하는 시스템을 구축한다. 그리고 측정한 거리 값과 좌표 값을 역방향 기구학적 모델링에 대입하여 굴착기의 관절 각도를 구할 수 있다. 즉, 작업자가 카메라를 통해 사물을 인식하면 굴착기가 사물에 도달하기 위한 필요 각도를 얻을 수 있고 굴착기가 간단한 작업은 자동으로 할 수 있는 시스템을 만든다.

A Study on the Measurement of Object Distance and Automatic Excavation System for Remote Control of Small Excavators

Tae Gon Son

Department of Construction Machinery Engineering

Graduate School, University of Ulsan

Abstract

Smart construction refers to the introduction of new technologies of the Fourth Industrial Revolution to general construction technology to improve productivity and work efficiency. The technology that is often applied to smart construction has environmental awareness and unmanned self-regulation. Many studies use radar, ultrasonic sensors, radar, and camera sensors for environmental awareness. Lida, ultrasonic sensors, and radars have the advantage of being able to accurately measure the distance information of objects, and cameras have often been used as the advantage of recognizing objects directly rather than the accuracy of distance information. However, as the need for unmanned self-regulation work increased, it became necessary for workers to recognize the surroundings of excavators, to recognize objects at remote locations, and to measure distances through cameras. In this paper, a system for measuring distance is established using cameras installed to control excavators remotely. And the joint angle of the excavator can be obtained by substituting the measured distance and coordinate values for reverse instrumental modeling. In other words, if the operator recognizes the object through the camera, the excavator can obtain the necessary angle to reach the object and the excavator can automatically do simple work.

목 차

표 및 그림 목차

1. 표 목차

2. 그림 목차

1. 서 론

1.1. 연구 배경 및 목적

Smart Construction은 전통적인 건설 기술에 BIM (Building Information Modeling), IoT (Internet of Things), Big Data, Drone, Robot 등 4차 산업혁명의 신기술을 도입하여 생산성 향상 발전을 도모하는 기술을 말하며 현재의 인력·경험 의존적인 산업에서 지식·첨단 산업으로 건설 산업의 패러다임 전환을 목표로 하는 것이다.[1]

Smart Construction 기술 중 대표적인 것으로 환경인식과 자율주행이 있다. 일반적으로 환경을 인식하는 센서로 라이다, 초음파 센서, 레이저 센서, 카메라를 사용하고 있다. 카메라의 경우, 날씨 조건에 민감하고, 거리가 멀어질수록 인식률이 떨어지며 화면에 변형이 많은 단점이 있어 일반적으로 라이다와 초음파센서를 사용하여 거리 측정 시스템을 만들고 있다.

일반적으로 많은 논문에서는 센서 융합을 통해 작업기 근처는 빠른 반응 속도를 가진 초음파센서를 사용하고 원거리의 경우는 정밀한 위치 정보를 얻을 수 있는 라이다를, 악천후와 같은 열악한 환경에서는 레이저를 이용하여 사용하여 거리를 측정하고 있다. 그리고 카메라를 이용하여 초음파 센서와 라이다를 통해 볼 수 없는 작업기 주변의 환경을 확인하는 정도로 사용한다.

본 논문에서는 작업자가 사물을 직접 확인할 수 있다는 큰 장점이 있는 카메라를 이용하여 환경인식을 하고, 굴착기 주변의 사물의 거리를 파악 할 수 있는 시스템을 구축한다. 사물의 거리 값과 좌표 값을 알 수 있다면, 4DoF의 역방향 기구학적 모델링을 통해 굴착기가 사물에 도달하기 위한 필요 관절 각도를 도출할 수 있으며, 이를 통해 굴착기가 사물에 도달하는 정도의 간단한 작업은 자동으로 할 수 있는 시스템을 만들 것이다.

1.2. 연구 동향

검색된 논문에 대한 정량 분석을 다음과 같다. 최근 5년간 거리 측정 센서 연구로 진행된 연구 편수는 Fig. 1-1과 같이 볼 수 있다.

키워드 : 거리 측정, 라이다, 초음파 센서, 라이다, 카메라 센서, 환경 인식
검색 사이트 : RISS, DBpia, ScienceDirect

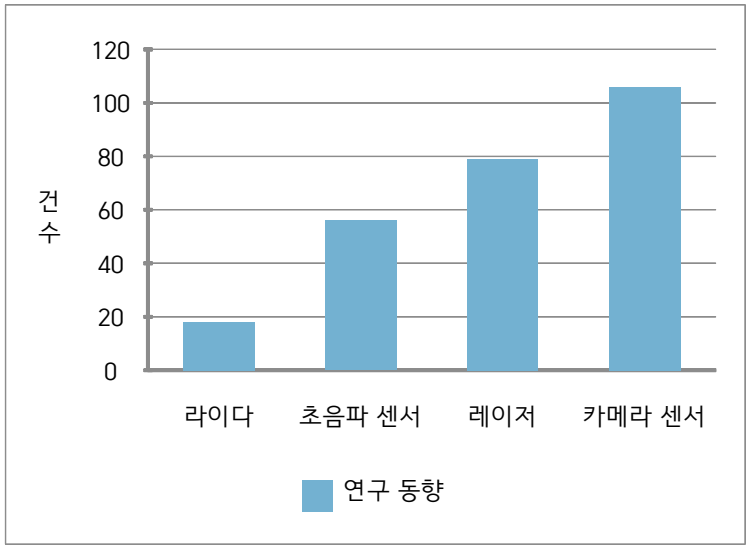


Fig. 1-1 최근 5년 연구 편수

Table. 1-1과 같이 연구된 내용을 분석한 결과는 다음과 같다.

라이다의 경우는 카메라 센서와 센서 융합을 통해 거리 측정과 환경 인식이 가능한 논문이 진행 되고 있고, 초음파 센서의 경우는 근거리용으로 사용되는 장점을 더욱 향상시키기 위한 최소 탐지 거리 및 정확도 향상에 논문을 진행함을 볼 수 있다.

레이저의 경우, 빠른 속도와 악천후와 같은 환경에서도 사용가능한 장점을 더욱 발전시키는 고속 주행용 센서와 수중 촬영용 센서, 인공위성과의 통신을 통한 더욱 정밀한 거리 측정에 관한 연구가 진행됨을 볼 수 있다.

카메라의 경우, 최근 5년간 연구 중 제일 확발히 진행되는 연구 주제이며, 스테레오 카메라를 통한 거리측정을 넘어서 뎁스 카메라의 개발과 모노카메라를 통한 거리 측정이 가능한 연구가 활발히 진행되고 있으며, 4차 산업혁명의 머신러닝, 딥러닝, 빅데이터의 연구와 맞물려 더욱 정확한 영상 데이터를 만드는 논문과 사물을 장비 스스로 판단하는 알고리즘 개발이 진행되고 있다.[2]

Table. 1-1 최근 5년 연구 동향

센서	최근 5년 연구 동향	주요 내용
라이다	18 건	라이다 거리측정 정확도 향상 센서 융합을 통한 물체 거리 측정 및 인식 시스템 개발 라이다 기반 소형 자율 주행 로봇 개발
초음파 센서	56 건	초음파 센서 최소 탐지 거리 및 정확도 향상
레이저	79 건	레이저 스캐너 정확도 분석 고속 주행용 정밀 측정 장치 개발 수중 입체 촬영을 위한 레이저 개발 펨토초 레이저 거리 측정 장치 개발
카메라 센서	106 건	다중 카메라를 이용한 실내 측위 카메라 영상에서의 실시간 객체 인식 및 거리 측정 연구 스테레오 카메라 거리 측정 뎁스 카메라를 통한 거리 측정 머신러닝을 통한 시정 거리 판단 시스템 구현 단일 카메라를 통한 거리 측정

최근 5년 연구 동향을 보았을 때 거리 측정 시스템에서 사용하는 센서는 초음파 센서, 라이다 센서, 레이더 센서, 카메라 센서가 사용하며, 거리 측정 방식에는 능동 탐지 방식과 수동 센서 방식이 있다.

능동 탐지 방식은 타겟에 신호를 전송하여 거리를 측정한다. 이러한 시스템은 레이저, 초음파, 전파 신호를 이용하며, 신호가 물체에 부딪혀 돌아오는 시간을 계산하여 물체거리를 추정한다.

수동 센서 방식은 사물의 위치에 대한 정보를 받아 거리를 측정한다. 일반적으로 모노 카메라와 스테레오 카메라를 사용하며 실제 사물의 크기와 현재 영상에 보이는 사물의 크기 비례를 이용해 거리를 측정한다. 각 센서의 장・단점은 다음 Table. 1-2와 같다.

Table. 1-2 거리 측정 센서

	장점	단점
라이다	장거리 탐지 가능, 날씨조건에 상대적으로 자유로움, 측정가능 거리 및 공간 분해능이 높음.	저해상도, 비·안개에 의해 빛이 굴절되면 제대로 작동 되지 않음. 거리 측정은 가능하나 객체 추정이 어려움.
초음파 센서	저가, 반응속도가 빠름, 수중에서도 사용 가능.	감지 거리가 짧음, 고속에서는 측정이 어려움, 다양한 정보를 얻기 어려움.
레이더	물체를 투과할 수 있음, 환경에 의한 영향이 적음, 빛에는 어떠한 영향도 받지 않음, 기상 환경에서도 작동이 가능함.	프로세싱이 복잡함, 분해능이 낮음, 전파 환경에서 사용이 불가능
카메라 센서	인간의 눈과 같이 다양한 사물을 한 번에 인지할 수 있음, 영상을 통해 얻어지는 객체와 실체 객체가 동일함, 본연의 색을 표현할 수 있음.	날씨 조건에 민감함, 렌즈 조건에 따라 화면에 변형이 많음, 거리가 멀어질수록 인식률이 떨어짐.

거리 측정 센서 중 원격에서 굴착기를 조종하기 위해서는 카메라를 통한 환경인식 및 사물 인식이 필요하다.

따라서 본 논문에서는 카메라를 사용하여 굴착기 주변을 인식하고 작업자가 지정한 사물의 거리를 측정하는 시스템을 구축하여 거리를 측정했다.

본 논문에서 하고자하는 시스템과 유사한 논문을 확인하기 위하여 카메라와 건설현장, 굴착기를 키워드로 논문을 검색하였고 연구 내용은 Table. 1-3과 같다.

Table. 1-3 자율 굴착 시스템 논문 내용

자율 굴착 시스템	연구 내용
무인 굴착기 자율 제어 시스템 개발	거리 측정 센서를 통한 환경인식 효율적인 작업을 하기위한 머신 러닝
굴착기 머신 컨트롤	자동화 작업 시스템 개발 머신 컨트롤을 통한 굴착 작업 효율화
건설현장 영상 분석을 위한 웹 크롤링 기반 학습 데이터베이스 구축 자동화	CCTV 카메라를 이용하여 건설현장 원격관찰 카메라를 통한 데이터베이스 학습
건설현장에 적합한 영상 기반 굴착기 접근 감지 시스템	카메라를 통한 굴착기 주변 환경인식 초음파 센서를 이용한 작업자 접근 경고
YOLO-v3을 활용한 건설 장비 주변 위험 상황 인지 알고리즘 개발	카메라를 통한 굴착기 주변 환경인식 작업자의 위험상황 판단 여부 알고리즘 구현

1.3. 연구 내용 및 방법

본 연구에서는 렌즈가 1개인 카메라를 2개 이용하여 렌즈가 2개인 스테레오 카메라와 같은 역할을 한다. 다른 센서와 다르게 카메라의 장점은 인간의 눈과 같이 다양한 사물을 한 번에 인지할 수 있으며, 영상을 통해 얻어지는 객체와 실체 객체가 동일하고, 사물 본연의 색을 표현할 수 있는 것이다. 단점으로 날씨 조건에 민감하며, 렌즈 조건에 따라 화면에 변형이 많고, 거리가 멀어질수록 인식률이 떨어지는 것이 있습니다. 좌, 우 카메라에 보이는 사물의 위상 차이를 이용하여 사물과 카메라 사이의 거리를 측정한다.

이번 연구의 경우, 소형굴착기가 동작하는 범위인 3m 이내의 작업공간을 선정하여 카메라가 작업 공간 내부의 거리 값을 정확히 계산하도록 연구 한다.

비전 시스템의 영상 처리와 머신러닝을 통한 사물 인식이 아닌 작업자가 카메라를 통해 환경을 인지하고 사물을 파악하여 지정하는 방식으로 연구를 진행한다. 그리고 작업자가 지정한 사물로 굴착기의 붐, 암, 버켓이 구동하도록 전자 비례 감압밸브를 제어하는 시스템을 만든다.

제작된 카메라 거리측정 시스템과 전자 비례 감압밸브 제어 시스템을 실제 1.5톤 굴착기에 부착하여 작업 범위인 3m 이내에서 거리 측정 시험과 굴착기 동작시험을 진행하여 거리 측정 시스템과 굴착기 자동화 시스템 동작 검증을 진행한다.

2. 자율 굴착기 시스템 및 제어

2.1. 굴착기의 모델링

본 논문에 사용된 1.5톤 굴착기의 구조는 Fig. 2-1과 같다. 굴착기의 유압 시스템과 에너지의 변환 과정은 엔진에서 연료의 화학 에너지를 기계적 에너지로 변환하고, 유압 펌프에서 다시 유압 에너지로 변환시켜 각 액추에이터에서 사용된다. 액추에이터로는 유압 실린더 및 유압 모터가 주로 사용된다.[3-5]

유압 실린더는 붐, 암, 버켓으로 이루어진 작업 장치를 움직이는 목적으로 사용되며 선회와 주행을 위해 유압 모터가 사용된다.

각 액추에이터의 방향, 유압, 유량을 제어하기 위해 펌프에서 송출된 유압 에너지는 메인 컨트롤 밸브(Main Control Valve, MCV)를 거치게 된다.

본 논문에서는 1.5톤 크롤러형 굴착기를 대상으로 기구 및 유압 시스템에 관하여 모델링하고 실차 테스트를 하였다.

Fig. 2-1 유압 굴착기(1.5톤)

1) 순방향 기구학적 모델링

순방향 기구학적 모델링은 각 조인트 공간에서 조인트 각도(θ)가 주어졌을 경우에 직교 공

간에서 링크 끝단의 위치(X)를 나타내는 것이다. 기준 좌표계에서 바라본 끝단의 위치를 구하기 위하여 식 (1)에서 나타낸 변환 행렬을 사용한다.

$$T_{i-1}^{i} = \begin{bmatrix} \cos\theta_i & -\cos\alpha_i\sin\theta_i & \sin\alpha_i\sin\theta_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\alpha_i\cos\theta_i & -\sin\alpha_i\cos\theta_i & a_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1)$$

여기서 a_i는 z축 사이의 거리, α_i는 z축 간의 각도, d_i는 x축 사이의 거리, θ_i는 x축 간의 각도를 의미한다.

최종적으로 기준 좌표계에서 바라본 끝단의 좌표는 각 좌표계의 변환 행렬을 모두 곱함으로서 얻을 수 있으며 식 (2)와 같다.

$$T_0^n = T_0^1 \times T_1^2 \times T_2^3 \times \ldots \times T_{n-1}^n \quad (2)$$

엔드 이펙터의 좌표를 구하기 위해 1.5톤 굴착기의 기준 좌표계를 설정한 모습은 그림 Fig. 2-2와 같고 간단히 도식화 한 형상은 Fig. 2-3과 같다.

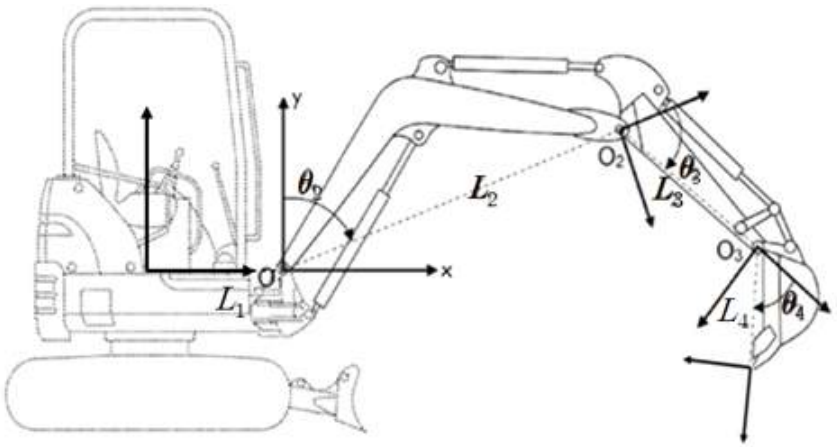


Fig. 2-2 유압 굴착기 기준 좌표계(1.5톤)

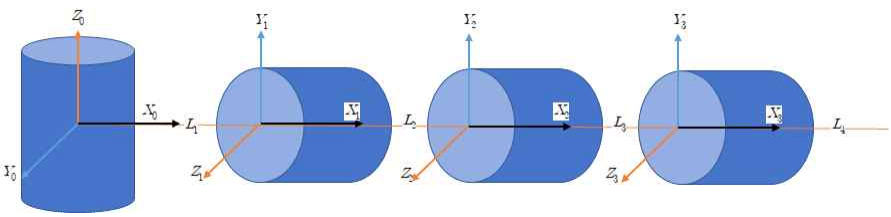


Fig. 2-3 유압 굴착기 기준 좌표계(간략화)

기준 좌표계에서 바라본 끝단의 위치를 알아내려면 각 좌표계 간의 변환 행렬을 구해야 한다. 이를 위해 대너빗-하텐버그 테이블을 작성하면 Table. 2-1과 같다.

Table. 2-1 대너빗-하텐버그 테이블

i	a_i	α_i	d_i	θ_i
1	L_1	$\frac{\pi}{2}$	0	θ_1
2	L_2	0	0	θ_2
3	L_3	0	0	θ_3
4	L_4	0	0	θ_4

Table. 2-1을 활용하여 각 좌표계의 변환 행렬을 구하면 수식 (3), (4)와 같고 기준 좌표계에서 바라본 끝단의 좌표는 식 (5)와 같다.

$$T_0^1 = \begin{bmatrix} \cos\theta_1 & 0 & \sin\theta_1 & L_1\cos\theta_1 \\ \sin\theta_1 & 0 & -\cos\theta_1 & L_1\sin\theta_1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},\ T_2^1 = \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & 0 & L_2\cos\theta_2 \\ \sin\theta_2 & \cos\theta_2 & 0 & L_2\sin\theta_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

$$T_3^2 = \begin{bmatrix} \cos\theta_3 & -\sin\theta_3 & 0 & L_3\cos\theta_3 \\ \sin\theta_3 & \cos\theta_3 & 0 & L_3\sin\theta_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},\ T_4^3 = \begin{bmatrix} \cos\theta_4 & -\sin\theta_4 & 0 & L_4\cos\theta_4 \\ \sin\theta_4 & \cos\theta_4 & 0 & L_4\sin\theta_4 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4}$$

$$P = T_0^1 T_1^2 T_2^3 T_3^4 = \begin{bmatrix} \cos\theta_{234}\cos\theta_1 - \sin\theta_{234}\cos\theta_1 & \sin\theta_1 & \cos\theta_1(L_1 + L_3\cos\theta_{23} + L_2\cos\theta_2 + L_4\cos\theta_{234}) \\ \cos\theta_{234}\sin\theta_1 - \sin\theta_{234}\sin\theta_1 & -\cos\theta_1 & \sin\theta_1(L_1 + L_3\cos\theta_{23} + L_2\cos\theta_2 + L_4\cos\theta_{234}) \\ \sin\theta_{234} & \cos\theta_{234} & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{matrix} \\ \\ L_3\sin\theta_{23} + L_2\sin\theta_2 + L_4\sin\theta_{234} \\ 1 \end{matrix} \tag{5}$$

이는 버켓의 끝단 위치가 기준 좌표계에서 x축 방향, y축 방향, z축 방향으로 식 (6), (7), (8)과 같이 떨어져 있음을 나타낸다.

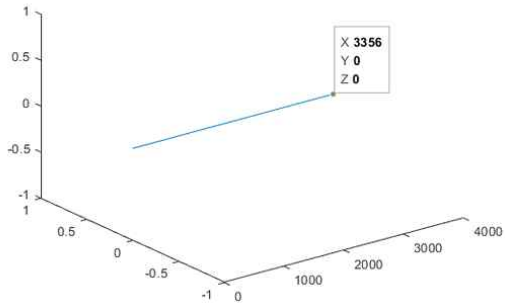
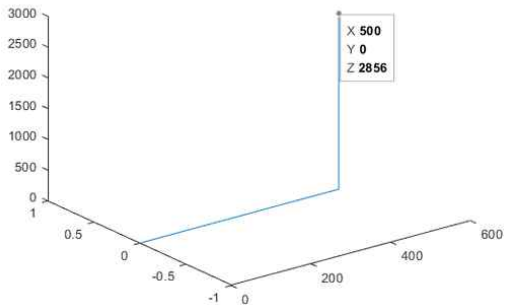
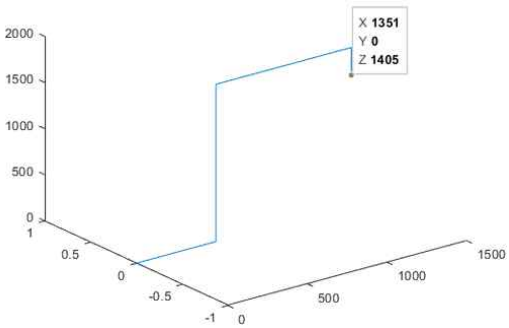
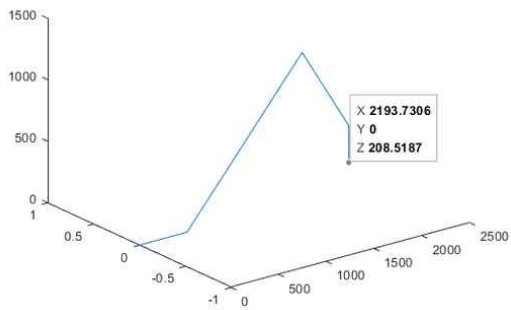
$$\cos\theta_1(L_1 + L_3\cos\theta_{23} + L_2\cos\theta_2 + L_4\cos\theta_{234}) \tag{6}$$

$$\sin\theta_1(L_1 + L_3\cos\theta_{23} + L_2\cos\theta_2 + L_4\cos_{234}) \tag{7}$$

$$L_3\sin\theta_{23} + L_2\sin\theta_2 + L_4\sin\theta_{234} \tag{8}$$

위의 결과를 MATLAB & Simulink을 사용하여 임의의 각도를 입력하여 검증하였으며 결과는 Table. 2-2와 같다.

Table. 2-2 Matlab & Simulink 검증 결과

(a) the1= 0, the2= 0, the3= 0, the4= 0	(b) the1= 0, the2= 90, the3= 0, the4= 0
	
End Point : (3356, 0, 0)	End Point : (500, 0, 2856)
(c) the1= 0, the2= 90, the3= -90, the4= -90	(b) the1= 0, the2= 45, the3= -100, the4= -35
	
End Point : (1351, 0, 1405)	End Point : (2194, 0, 209)

2) 역방향 기구학적 모델링

역방향 기구학적 모델링은 앞 절의 순방향 기구학적 모델링과는 반대로 위치가 주어졌을 때 조인트 각도를 구하는 것을 의미한다. 역방향 기구학적 모델링을 통해 조인트 각도를 구하는 방법은 크게 기하학적 방법과 변환행렬의 특성을 사용하여 구하는 방법 두 가지가 있다. 본 논문에서는 변환행렬의 특성을 사용하여 조인트 각도를 구하였다.

식 (2)에서 알 수 있듯이 기준 좌표계에서 바라본 작업 장치 끝단의 위치는 각 조인트의 변환행렬을 곱한 것과 같다. 여기서 T_0^1의 역행렬을 양변에 곱하면 식 (9)와 같고 전개하면 식 (10)과 같다.

$$\left[T_0^1\right]^{-1} \times P = \left[T_0^1\right]^{-1} \times T_0^4 = T_1^2 \times T_2^3 \times T_3^4 \tag{9}$$

$$\begin{bmatrix} n_x\cos\theta_1 + n_y\sin\theta_1 & o_x\cos\theta_1 + o_y\sin\theta_1 & a_x\cos\theta_1 + a_y\sin\theta_1 & p_x\cos\theta_1 - L_1 + p_y\sin\theta_1 \\ n_z & o_z & a_z & p_z \\ n_x\sin\theta_1 - n_y\cos\theta_1 & o_x\sin\theta_1 - o_y\cos\theta_1 & a_x\sin\theta_1 - a_y\cos\theta_1 & p_x\sin\theta_1 - p_y\cos\theta_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos\theta_{234} & -\sin\theta_{234} & 0 & l_3\cos\theta_{23} + l_2\cos\theta_2 + l_4\cos\theta_{234} \\ \sin\theta_{234} & \cos\theta_{234} & 0 & l_3\sin\theta_{23} + l_2\sin\theta_2 + l_4\sin\theta_{234} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{10}$$

θ_1을 구하기 위해 두 행렬의 (3,4) 성분을 비교하여 θ_1을 구한 결과는 식 (11)과 같다.

$$\tan\theta_1 = \frac{p_y}{p_x} \rightarrow \therefore \text{atan2}(p_y, p_x) \tag{11}$$

$\theta_2 + \theta_3 + \theta_4$ 값을 구하기 위해 두 행렬의 (1,1) 성분 및 (2,1) 성분을 비교하여 $\theta_2 + \theta_3 + \theta_4$을 구한 결과는 식 (12)와 같다.

$$\frac{\sin_{234}}{\cos_{234}} = \frac{-(o_x\cos\theta_1 + o_y\sin\theta_1)}{n_x\cos\theta_1 + n_y\sin\theta_1} \rightarrow \tag{12}$$

$$\therefore \theta_2 + \theta_3 + \theta_4 = \text{atan2}(-(o_x\cos\theta_1 + o_y\sin\theta_1), n_x\cos\theta_1 + n_y\sin\theta_1)$$

θ_2, θ_3을 구하기 위해 두 행렬의 (1,4) 성분 및 (2,4) 성분을 비교하며 식 (13), (14)와 같이 정의한다.

$$p_x\cos\theta_1 - L_1 - p_y\sin\theta_1 = L_2\cos\theta_2 + L_3\cos\theta_{23} + L_4\cos\theta_{234} \rightarrow L_2\cos\theta_2 + L_3\cos\theta_{23} = X_1 \tag{13}$$

$$p_z = L_2\sin\theta_2 + L_3\sin\theta_{23} + L_4\sin\theta_{234} \rightarrow L_2\sin\theta_2 + L_3\sin_{23} = X_2 \tag{14}$$

식 (10)과 (11)을 제곱하여 더해 θ_3을 구한 결과는 식 (15)와 같다.

$$\cos\theta_3 = \frac{(X_1)^2+(X_2)^2-(L_2)^2-(L_3)^2}{2L_2L_3} \rightarrow \therefore \text{acos}(\frac{(X_1)^2+(X_2)^2-(L_2)^2-(L_3)^2}{2L_2L_3}) \quad (15)$$

식 (10)과 (11)에 있는 $\sin\theta_{23}$ 및 $\cos\theta_{23}$을 전개하여 정리하면 식 (16) 및 (17)와 같다.

$$(L_2+L_3\cos\theta_3)\cos\theta_2 - L_3\sin\theta_2\sin\theta_3 \rightarrow k_1\cos\theta_2 - k_2\sin\theta_2 = X_1 \quad (16)$$

$$(L_2+L_3\cos\theta_3)\sin\theta_2 + L_3\cos\theta_2\sin\theta_3 \rightarrow k_1\sin\theta_2 + k_2\cos\theta_2 = X_2 \quad (17)$$

이러한 형태의 수식을 풀기 위해서는 $L_2+L_3\cos\theta_3 = k_1$, $L_3\sin\theta_3 = k_2$로 치환하여 다음과 같이 풀 수 있다.

$$r = \sqrt{(k_1)^2+(k_2)^2}$$

$$\gamma = \text{atan2}(k_2, k_1) \qquad k_1 = r\cos\gamma \text{ and } k_2 = r\sin\gamma$$

$$\frac{X_1}{r} = \cos\gamma\cos\theta_2 - \sin\gamma\sin\theta_2 = \cos(\gamma+\theta_2)$$

$$\frac{X_2}{r} = \cos\gamma\cos\theta_2 + \sin\gamma\sin\theta_2 = \sin(\gamma+\theta_2)$$

$$\gamma+\theta_2 = \text{atan2}(\frac{X_2}{r}, \frac{X_1}{r}) = \text{atan2}(X_2, X_1) \quad (18)$$

식 (18)을 전개하면 θ_2값도 구해진다.

$$\theta_2 = \text{atan2}(X_2, X_1) - \text{atan2}(k_2, k_1) = \text{atan2}(X_2, X_1) - \text{atan2}(L_3\sin\theta_3, L_2+L_3\cos\theta_3)$$

식 (12), (15), (18)를 통해 $\theta_2, \theta_3, \theta_4$를 구할 수 있다.

$$\theta_4 = \theta_{234} - \theta_2 - \theta_3 \quad (19)$$

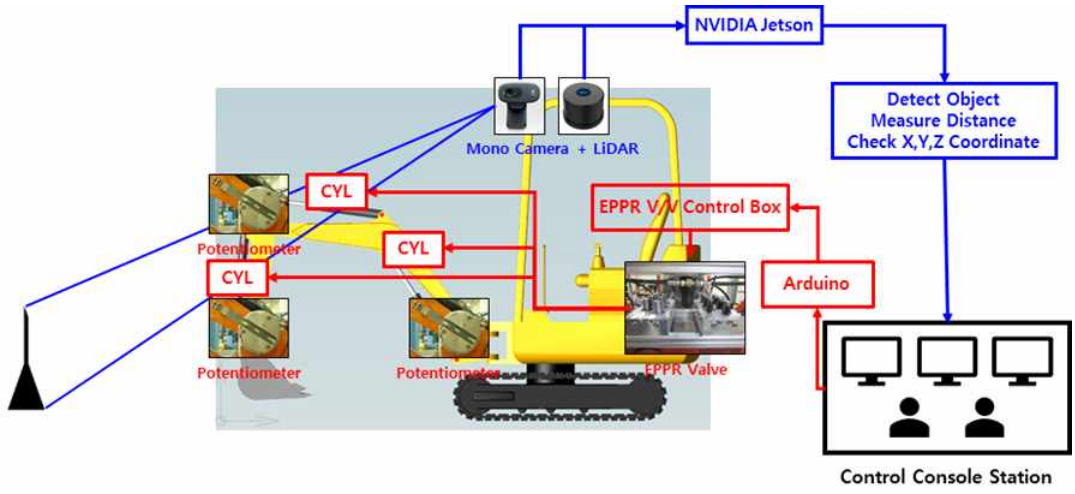
임의의 좌표를 지정하면 역방향 기구학모델링을 통해 $\theta_1, \theta_2, \theta_3, \theta_4$ 값을 찾아 낼 수 있다.

2.2. 자율 굴착 시스템

스테레오 카메라와 라이다를 통한 거리 측정 시스템을 Fig. 2-4와 같이 실제 차량에 설치하였다. 원격에서 작업자가 사물을 인지하고 지정하면 사물에 대한 거리정보와 좌표를 얻을 수 있게 하였다. [6-7]

획득한 거리와 좌표정보를 역방향 기구학적 모델링에 대입하여 굴착기가 사물에 도달하기 위한 필요 각도를 산출해 낸다. 현재 굴착기의 각 관절 데이터 값은 굴착기 관절에 부착된 관절 센서를 통해 얻어내고 제어기를 통해 필요 각도로 구동한다.

카메라를 통한 영상 데이터를 메인 컨트롤러인 젯슨 나노보드로 보내고 작업자가 영상 데이터를 선택한다. 역방향 기구학적 모델링을 통해 계산된 각도 데이터를 제어기인 아두이노로 전송하여 전자 비례 감압밸브로 메인 컨트롤 밸브의 스풀을 움직인다. 이를 통해 붐, 암, 버켓이 동작하고, 실시간으로 관절부에 연결된 관전 센서를 통해 각도 데이터를 피드백하여 제어기인 아두이노로 전송한다.



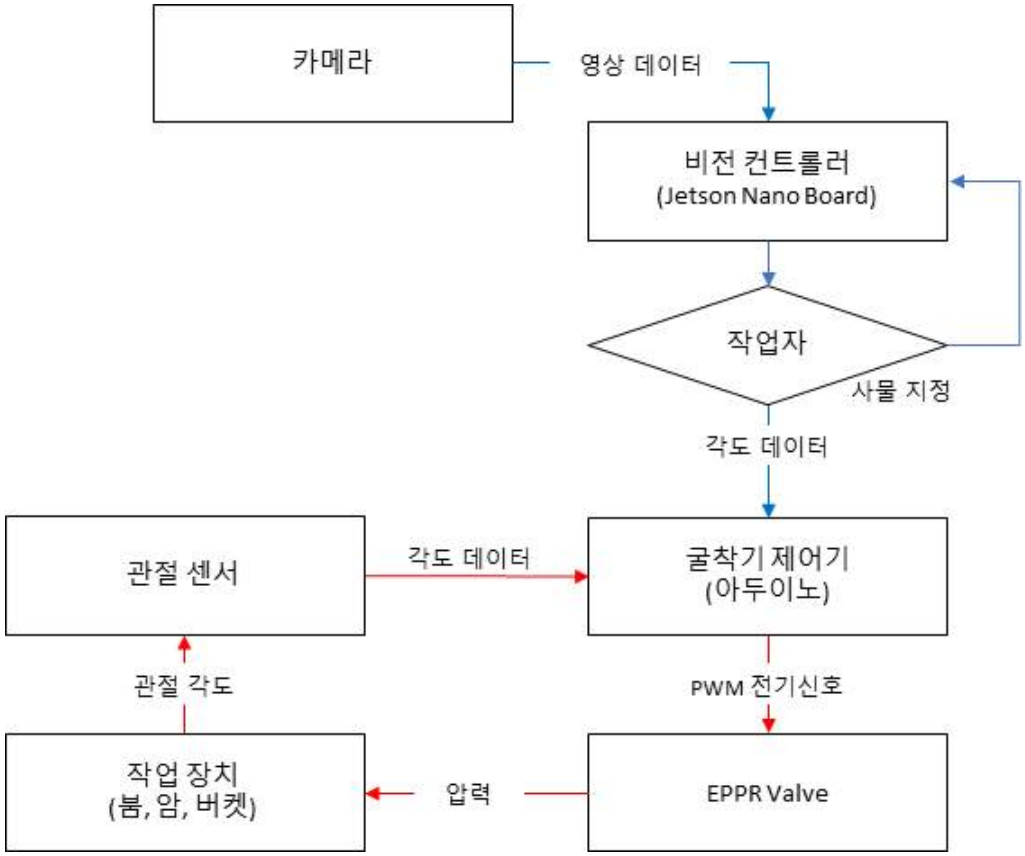


Fig. 2-4 자율 굴착 시스템

1) 관절 센서

포텐셔미터는 가변 저항으로 중간 회전자가 회전하며 이동한 만큼 저항 값이 변화한다. 변화하는 저항 값을 기준으로 회전하는 각도를 측정한다.

(a) 붐 (b) 암, 버켓

Fig. 2-5 굴착기 붐, 암, 버켓 각도센서

각 굴착기 관절에 개발된 관절 센서를 부착한 모양은 Fig. 2-5의 (a), (b)와 같다. 굴착기에 연결한 관절 센서는 포텐셔미터이다.

본 연구실에서 사용하는 굴착기에 관절센서를 연결하여 붐, 암, 버켓의 허용가능 각도와 그에 맞는 데이터 신호를 확인하였고 결과는 Table. 2-3과 같다.

Table. 2-3 붐, 암, 버켓 허용가능 각도와 신호 값

Parts	Min	Max
Boom(θ_2)	-57 ° (down)	66 ° (up)
Arm(θ_3)	-133 ° (out)	-24 ° (in)
Bucket(θ_4)	-140 ° (in)	24 ° (out)

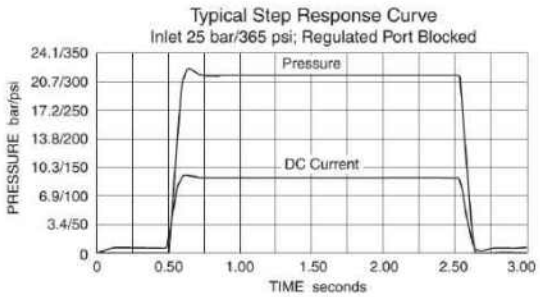
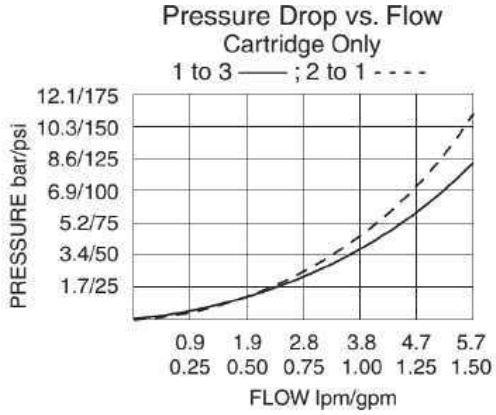
2) 전자 비례 감압밸브

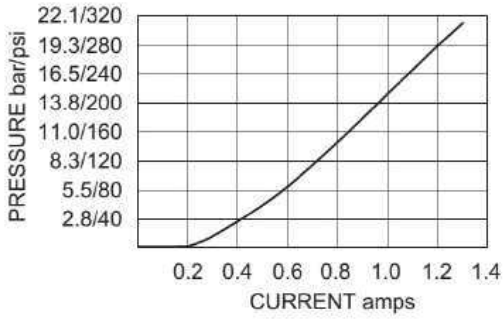
전자 비례 감압밸브(Electric Proportional Pressure Reducing Valve)로 입력 전류에 비례하여 압력을 제어하는 밸브다. 일반적으로 중장비의 메인 컨트롤 밸브에 사용되는 스풀을 전자적으로 제어 할 수 있는 핵심부품으로 건설 중장비, 선박, 제철, 광산 분야를 포함하여 전반적인 일반 산업분야 및 자동화 분야에 다양하게 응용된다.

본 시스템에 적용된 전자 비례 감압밸브는 EHPR98-T35이며 형상 및 제원은 Fig. 2-6 및 Table. 2-4와 같다.

Fig. 2-6 전자 비례 감압밸브

Table. 2-4 전자 비례 감압밸브의 제원

Model Name	EHPR98-T35
Max Inlet Pressure (bar)	103
Max Tank Port Pressure (bar)	34.5
Control Pressure at Max Control Current (A)	20.7
Max Control Current (A)	1.38 A for 10 VDC Coil 1.30 A for 12 VDC Coil 0.69 A for 20 VDC Coil 0.65 A for 24 VDC Coil
Resistance (ohm)	4.2 ohm (10V) 5.1 ohm (10V) 17 ohm (20V) 19.3 ohm (24V)
Inductance	80 mH (12V)
Hysteresis	less than 4% with 100 Hz PWM
Rated Flow (lpm)	5.7
Max Internal Leakage	De-energized : 75 ml/min at 25 bar Energized at I-Max : 125 ml/min at 25 bar
Temperature (degree)	-40 to 120
Ambient Air Temperature (degree)	-40 to 80
Electric Controllers	EVDR Series Controller
Performance	
Pressure Drop vs Flow Catirdge Only	

Reducing Pressure vs Current Inlet	

전자 비례 감압밸브를 제어하기 위해 Hydraforce사의 EVDR1 컨트롤러를 적용하였으며 형상은 Fig. 2-7과 같다. 컨트롤러로 입력 전압이 들어가면 전류를 내보낼 수 있으며 이를 활용하여 전자 비례 감압밸브를 제어할 수 있다. 또한 EVDR1 컨트롤러의 출력은 정방향 및 역방향 모두 가능하며 입력 전압 대비 출력 전류를 매핑한 모습은 Fig. 2-8과 같다.

Fig. 2-7 EVDR1 Controller

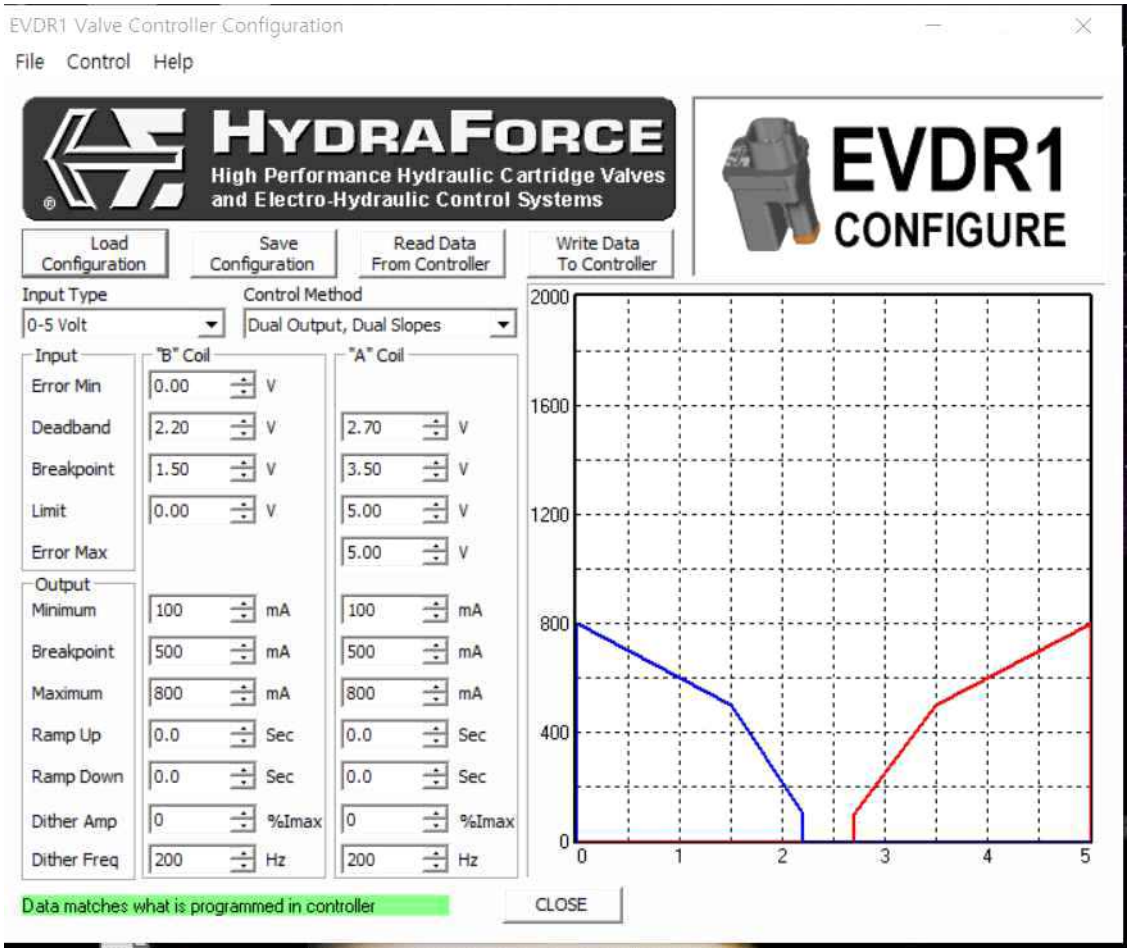


Fig. 2-8 EVDR1 매핑 화면(Boom)

본 시스템에서는 아두이노의 0 ~ 5 V 전기신호를 매핑하여 전자 비례 감압밸브로 보내야 한다. 그러나 아두이노에는 DAC(Digital to Analog Conveter)가 내장되어 있지 않으므로 0 ~ 5 V 아날로그 전기 신호가 아닌 PWM(Pulse With Modulation) 신호로 내보낸다.

Fig. 2-9 전자 비례 감압밸브

굴착기 메인 컨트롤 밸브의 스풀을 움직이는 조이스틱을 대신하여 전자 비례 감압밸브가 메인 컨트롤 밸브의 스풀을 움직이도록 구성하였고, 실험실에 사용한 1.5톤 굴착기에 연결되어있는 밸브는 Fig. 2-9와 같이 붐, 암, 버켓, 스윙, 주행 좌, 주행 우에 연결했다.

3) 제어기

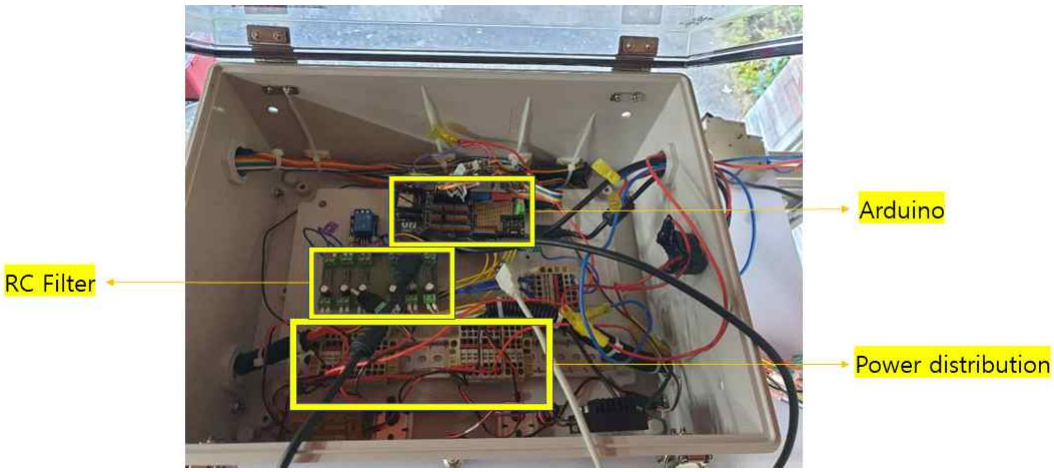


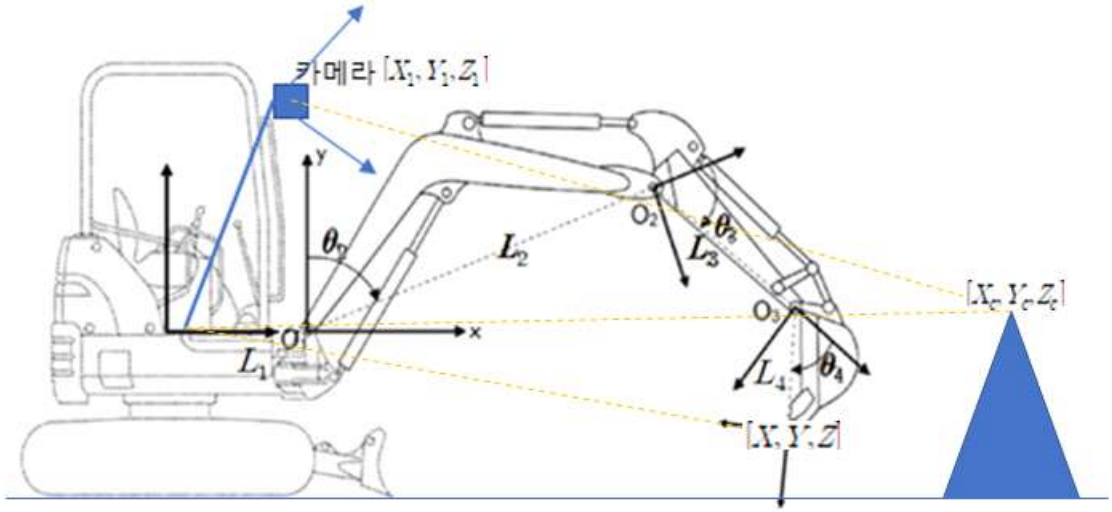
Fig. 2-10 제어기(아두이노)

굴착기의 제어기는 Fig. 2-10과 같이 구성했다. 제어기는 크게 아두이노와 PWM 신호를 매끄럽게 바꿔줄 RC 필터, 메인 컨트롤러와 제어기, 스테레오 카메라에 전력을 분배하는 전원 분배 장치로 구성했다.

3. 거리 측정 시스템 및 계측

3.1. 거리 측정 시스템

거리 측정 시스템이란 거리를 측정할 수 있는 센서를 통해 사물과의 거리를 측정하는 것을 말한다. [8]



($[X_c, Y_c, Z_c]$: 카메라를 통한 사물의 좌표, $[X, Y, Z]$: 월드좌표계, $[X_1, Y_1, Z_1]$: 카메라좌표계)

Fig. 3-1 거리 측정 시스템 개요

1) 거리 측정 계산식

거리를 측정하는 방식으로 스테레오 카메라를 이용하였다. 스테레오 카메라를 통해 거리를 측정하는 계산식은 Fig. 3-2와 같다.[9-11]

Fig. 3-2에서 Baseline은 좌, 우 렌즈의 거리 차이고, f는 Focal Length로 카메라 본연의 초점거리이다. 좌, 우 영상에서 동일하게 나타나는 물체에 대한 X축 위치($X-X'$) 차이 시차(Disparity)를 이용해 거리(Z)를 구할 수 있다. 식 (1)에서 시차는 스테레오 정합을 위한 두 이미지에서 객체 위상차이 이다.

카메라를 통해 거리측정을 하기 위해서는 카메라의 내부 파라미터(Q)를 얻어내는 작업인 카메라 캘리브레이션과 왜곡 보정을 통한 영상 처리 작업을 해야 한다.

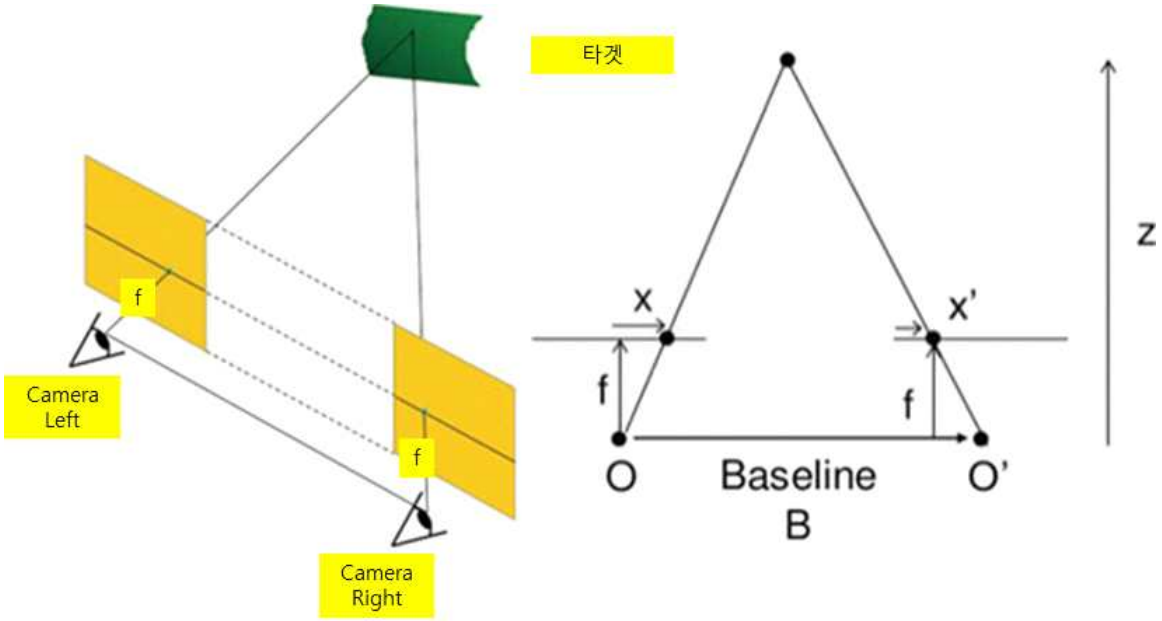


Fig. 3-2 스테레오 카메라 거리 측정 방식

$$disparity = X - X' = \frac{B * f}{Z} \quad (1)$$

(X, X' : 좌우 카메라 사물의 X좌표, B : 카메라 렌즈 거리, f : 초점거리, Z : 사물 거리)

3.2 스테레오 카메라 거리 측정 방식

스테레오 카메라를 동작하고 이미지 프로세싱과 같은 영상 처리 작업을 하기 위해 많이 사용하는 프로그램으로 파이썬을 사용하였고, 실시간 컴퓨터 비전을 목적으로 한 프로그래밍 라이브러리 OpenCV을 사용하여 영상 처리 작업의 정확도와 신속성을 향상시켰다.

카메라를 이용하여 거리를 측정하기 위해서는 필수적으로 해야 하는 작업이 있으며, 그 중 렌즈 왜곡을 보정해주는 작업과 카메라 캘리브레이션을 하는 작업을 본 논문에서 사용하여 거리 측정 결과의 정확도를 향상시켰다.[12-14]

1) 렌즈 왜곡 및 왜곡 보정

카메라 렌즈 왜곡 보정이란, 렌즈의 특성으로 생겨날 수밖에 없는 현상으로 영상에서 검출한 물체의 실제 위치를 알기 위해 영상좌표를 물리적인 좌표로 변환할 때, 영상 왜곡을 보정 하지 않으면 오차가 발생한다.

렌즈 왜곡은 방사왜곡(radial distortion)과 접선왜곡(tangential distortion)이 있다. 방사왜곡은 볼록렌즈의 굴절률에 의한 것으로 중심에서 멀어지고 가장자리로 갈수록 영상이 늘어나는 현상을 말한다. 접선왜곡의 경우, 카메라의 렌즈와 이미지 센서의 수평이 맞지 않거나 렌즈 자체의 중앙점이 맞지 않아서 발생하는 왜곡으로 렌즈의 중심점이 영상의 중심에 위치하지 않는 현상을 말한다.

일반적으로 렌즈 왜곡 보정을 하는 방법은 카메라 캘리브레이션을 통해 카메라 내부 파라미터를 얻어 낸 뒤, 이 데이터를 기반으로 하여 왜곡된 영상을 조정할 때 생기는 빈 픽셀을 채워주는 방식이다. 일반적으로 접선왜곡은 거의 발생하지 않기 때문에 방사왜곡을 보정해주는 작업으로 오차를 크게 줄일 수 있다.

2) 카메라 캘리브레이션(Camera Calibration)

카메라 캘리브레이션은 영상처리, 컴퓨터 비전 분야에서 꼭 필요한 과정중의 하나이다. 카메라로 찍은 이미지는 2차원 자료이기에 이를 3차원의 점들이 이미지 상에서 어디에 맺히는지는 기하학적으로 생각하면 영상을 찍을 당시의 카메라의 위치 및 방향에 의해 결정된다.

하지만 이 경우 사용된 렌즈, 렌즈와 이미지 센서 와의 거리, 렌즈와 이미지 센서가 이루는 각도 등 카메라 내부의 기구적인 부분에 의해서 크게 영향을 받는다. 따라서 3차원 점들이 영상에 투영된 위치를 구하거나 역으로 영상좌표로부터 3차원 공간좌표를 복원할 때에는

이러한 내부 요인을 제거해야만 정확한 거리 정보 계산이 가능하다. 이러한 내부 요인의 파라미터 값을 구하는 과정을 카메라 캘리브레이션 이라고 한다.

$$s\begin{bmatrix}x\\y\\1\end{bmatrix} = A\,[R|t]\begin{bmatrix}X\\Y\\Z\\1\end{bmatrix} \tag{2}$$

($[x,y,1]$: 이미지 평면 속 좌표, $[X, Y, Z, 1]$: 월드 좌표계에서 사물의 위치,
A : 카메라 내부 파라미터, $[R|t]$: 카메라 외부 파라미터)

식 (2)에서의 $[x, y, 1]$는 이미지 평면 속 좌표이고, A는 카메라 내부 파라미터로 초점거리(Focal length, f_x, f_y), 주점(principal point, c_x, c_y), 비대칭계수(skew coefficient)를 포함하고 있다. $[R|t]$는 카메라 외부 파라미터로 월드 좌표계 기준으로 카메라의 위치 좌표 값이고, $[X, Y, Z, 1]$는 월드 좌표계에서 사물의 위치이다.[15]

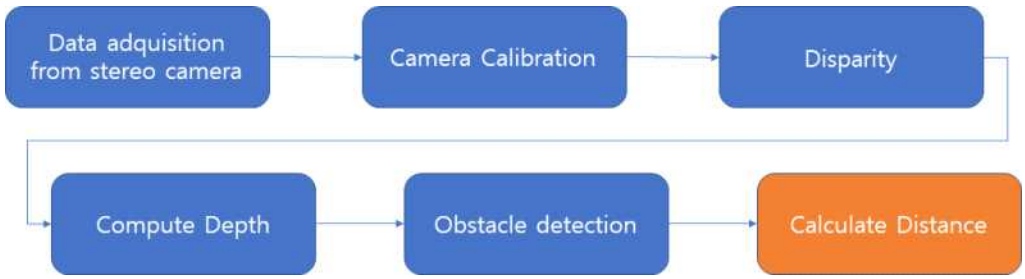


Fig. 3-3 거리 측정 과정

거리 측정을 하는 전체과정은 Fig. 3-3과 같다. 스테레오 카메라의 외부 파라미터를 받고, 카메라 캘리브레이션을 통해 카메라 내부 파라미터를 얻는다. 이들 모든 파라미터를 구하기 위해서는 정의된 패턴인 체스보드 이미지를 제공하는 것이다. 이미지 내부에 지정된 지점인 체스보드의 사각형 모서리를 찾으며, 이미 알고 있는 좌표의 이미지가 카메라에서 어디에 위치하는지 확인 할 수 있는 지점이다. 확인된 데이터는 식 (2)를 이용하여 왜곡 계수를 구할 수 있다.

Fig. 3-4 캘리브레이션 과정

정확한 결과를 얻기 위해서는 최소한 10개의 패턴 이미지가 필요하다. 카메라를 통해 인식된 체스보드의 모서리와 사각형의 형상을 실제 체스보드의 모습과 비교하여 카메라의 내부 파라미터를 파악하고, 확인된 파라미터를 역산하여 이미지 센서에 대입한다. 사각형의 형상을 비교한 것은 Fig. 3-4와 같다.

영상 데이터를 통해 왜곡을 줄인 영상을 작업자에게 보여주며, 작업자는 영상 정보를 통해 사물을 파악할 수 있다. 왜곡 처리된 영상 정보를 통해 사물을 파악하고, 작업자가 사물을 지정하면 좌, 우 화면에서의 사물 위상 차이가 발생하고, 이는 시차(Disparity)라고 한다. 시차를 통해 사물과의 거리 정보를 표현한 깊이를 계산할 수 있으며, 작업자가 지정한 사물의 깊이 값을 이용하여 카메라와 사물의 거리를 계산한다.

3) 사물 좌표 측정

카메라를 통해 사물과 카메라의 거리를 계산하였고, 계산한 거리를 통해 카메라 기준으로 사물의 좌표를 구해야 한다. Fig. 3-5와 같이 초점거리일 때 (x, y)값을 비례식을 통해 Z_c일 때 (X_c, Y_c, Z_c)값으로 구할 수 있다. 비례식을 풀면 식 (1)과 같다.

$$x = f * X_c / Z_c + C_x \quad (1)$$

$$y = f * Y_c / Z_c + C_y$$

((x, y):카메라 영상에서 사물 좌표,(X_c, Y_c, Z_c):실제 사물의 좌표,f:초점거리,Z_c:사물의 거리))

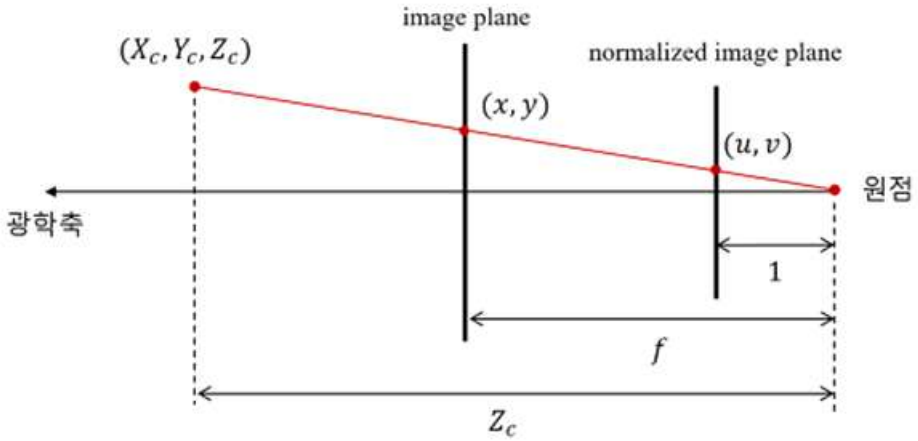


Fig. 3-5 사물 좌표 측정

3.3. 실험 장치 제작

실제 굴착기에 부착하여 거리를 측정하기에 앞서 스테레오 카메라를 통한 방식의 정확도를 알아보기 위하여 간단한 실험실 테스트벤치를 제작하여 Lab Test를 했다.

Lab Test를 위해 제작한 테스트벤치에 사용한 측정 부품은 다음과 같다.

1) 라이다

테스트벤치에서 사용한 라이다는 Table. 3-1을 사용하였다. 라이다를 이용하여 테스트벤치에서 사물의 위치를 정확히 구할 수 있으며, 라이다를 통한 거리와 스테레오 카메라를 통한 거리를 비교하여 스테레오 카메라를 통한 거리 측정 시스템의 타당성을 확인할 수 있다.

Table. 3-1 Lidar

Model Name	Garmin Lidar-Lite v3 HP
Power	40 mA
Range	5 cm ~ 40 m
Typical accuracy	± 2.5 cm

2) 스테레오 카메라

테스트벤치에서 사용한 카메라는 Table. 3-2를 사용하였다. 이 카메라를 2개 사용하여 스테레오 카메라로 사용하였으며, 이를 통해 거리를 측정할 수 있다.

Table. 3-2 Camera

Model Name	Logitech C270
Width x Height	1280 x 960 px
Frame rate	30 fps
Focal Length X x Y	1430 x 1430 px
Optical Center X x Y	480 x 620 px

1개의 카메라로는 2차원의 사물정보만 얻을 수 있다. Fig. 3-6과 같이 사물을 입체적으로 보기 위해서는 2개의 카메라를 동일 선상에 두고 두 카메라를 통해 사물을 봐야 한다. 좌, 우 화면에서 인식되는 사물을 각각 지정함으로써 좌, 우 화면에서 사물의 X좌표를 알 수 있다.

Fig. 3-6 스테레오 카메라

3) 메인 컨트롤 컴퓨터

젯슨 나노 보드는 인공지능 AI와 딥 러닝을 전문적으로 다루기 위해 개발된 소형 컴퓨터이다. 실시간 이미지 분류, 개체감지, 세분화, 음성 처리 기술을 지원하고, PyTorch와 TensorFlow 등 머신러닝 프레임워크를 사용할 수 있는 차세대 마이컴으로 많이 사용된다. 젯슨 나노 보드의 제원은 Table. 3-3과 같다.

파이썬과 OpenCV를 사용하여 스테레오 카메라를 작동하고, 카메라 캘리브레이션 작업과 렌즈 왜곡 보정 작업을 실시간으로 병행한다. 왜곡이 보정된 영상을 사용하면 좌우 카메라를 통해 사물의 깊이를 알 수 있는 Depth Map을 실행시킬 수 있다.

Table. 3-3 Main Computer

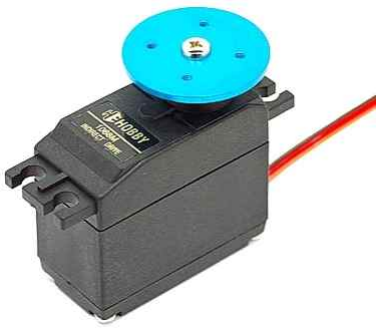
Model Name	NVIDIA Jetson nano board
GPU	128 코어 NVIDIA Maxwell
CPU	쿼드 코어 ARM A57 1.43 GHz
Memory	4 GB 64 비트 LPDDR4 25.6 GB/s
Size	100 mm x 80 mm x 29 mm

4) 카메라 구동 장치

원격에서 작업자가 원하는 사물을 찾기 위해 스테레오 카메라에 서보모터를 연결하여 좌우 회전(Pan)과 상하 회전(Tilt)을 할 수 있게 하였다. 상하좌우 회전을 통해 굴착기 전면부의 환경을 인식하고, 작업자가 원하는 사물의 거리를 측정할 수 있게 하였다. 사용한 서보 모터의 제원은 Table. 3-4와 같다.

차후 연구를 추가로 진행하여 인공지능과 빅데이터를 적용하여 실시간으로 사물을 인지하고 거리 정보를 작업자에게 보여줄 수 있도록 할 것이다.

Table. 3-4 서보 모터



Model Name	Jumbo dgs-1088m
Modulation	Digital
Torque	25 kg/cm in 6.0 V
Speed	0.11 sec/60° in 6.0 V
Size	58.9 mm x 29.0 mm x 55.9 mm

5) 테스트벤치

굴착기에 부착하여 거리를 측정하고 환경을 인식하기 이전에, 실험실에서 테스트벤치를 제작하여 Lab Test를 진행하였다. 이 실험을 통해 스테레오 카메라를 통한 거리 측정 시스템의 타당성을 확인하였으며, 차후 굴착기에 부착하기 위한 사전 준비과정으로 활용하였다.

테스트벤치를 제작하기 전에 3D 설계 소프트웨어 SolidWorks를 통해 만들었으며, Fig. 3-7

과 같이 테스트벤치 설계하였고, 타겟의 허용 가능 범위를 지정하였고 필요한 물품을 구입할 때 이용하였다. 설계를 통한 테스트벤치의 제원은 다음 Table. 3-5와 같다.

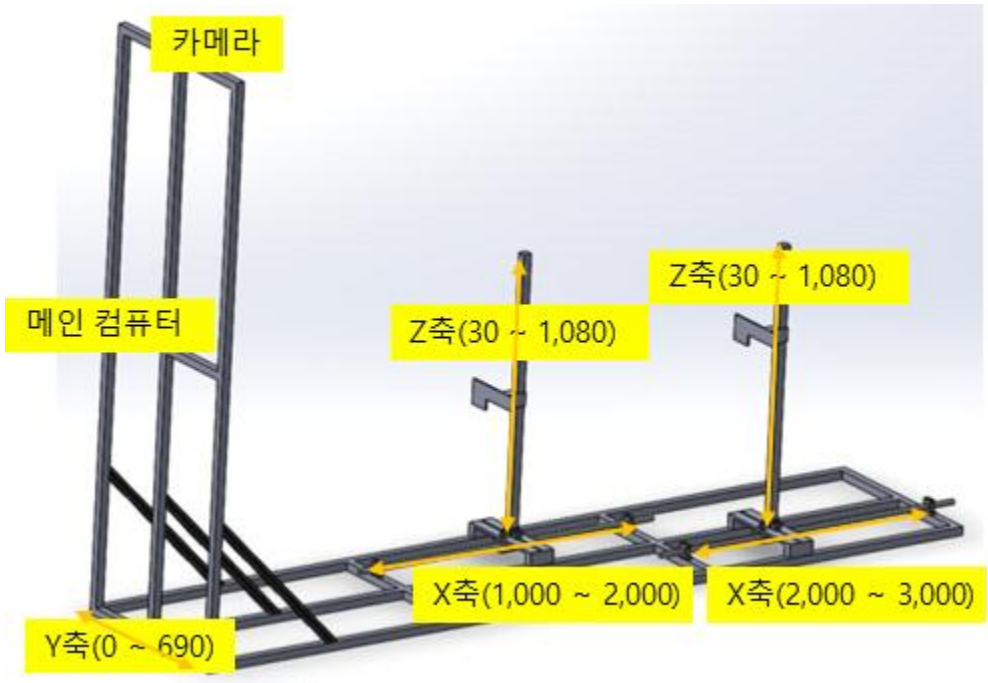


Fig. 3-7 테스트벤치 형상

Table. 3-5 테스트벤치 제원

	제원
측정 가능 거리	X축 거리 : 1,000 ~ 3,000 mm
	Y축 거리 : 0 ~ 690 mm
	Z축 거리 : 30 ~ 1,080 mm
카메라 외부 파라미터($[R\|t]$)	높이(H) : 2160 mm
	Tilt : 0 ~ 90 °
	Pan : 0 °
카메라 내부 파라미터(A)	f_x : 1,430 px / f_y : 1,430 px
	c_x : 480 px / c_y : 620 px
	k_1 : 0.918 / k_2 : -26.858
	p_1 : 0.0138 / p_2 : 0.0177

3.4. Lab Test

1) 시스템 구성

실차 시험 전, 테스트벤치를 구성하고 랩 테스트를 진행하였다. 랩 테스트를 통해 스테레오 카메라를 이용한 거리 측정 시스템을 확인하였다. 랩 테스트를 위한 테스트벤치는 Fig. 3-8과 같이 제작하였다.

서보 모터를 통해 상, 하로 움직이는 스테레오 카메라는 2,160 mm 높이에 위치하며, 유선으로 연결되어 있는 젯슨 나노 보드를 통해 구동하고 영상 처리 작업을 한다.

타겟은 앞, 뒤로 움직이는 Rail을 통해 1,000 ~ 3,000 mm까지 이동이 가능하며 위, 아래로는 30 ~ 1,080 mm 높이를 움직일 수 있다.

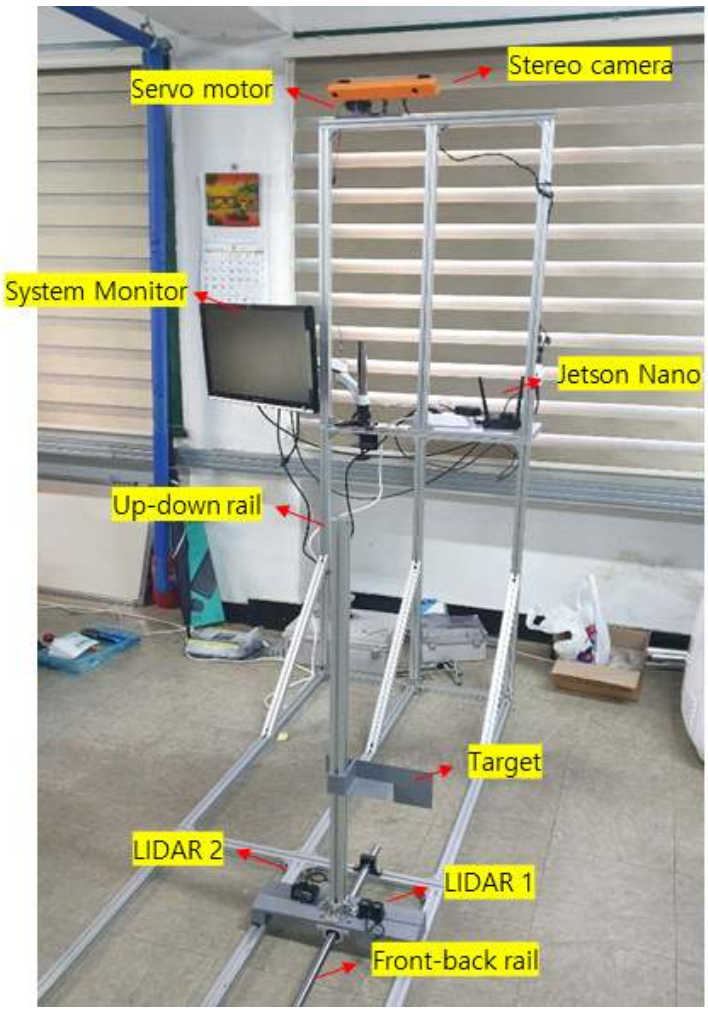


Fig. 3-8 Lab Test용 테스트벤치

2) 시험방법 및 조건

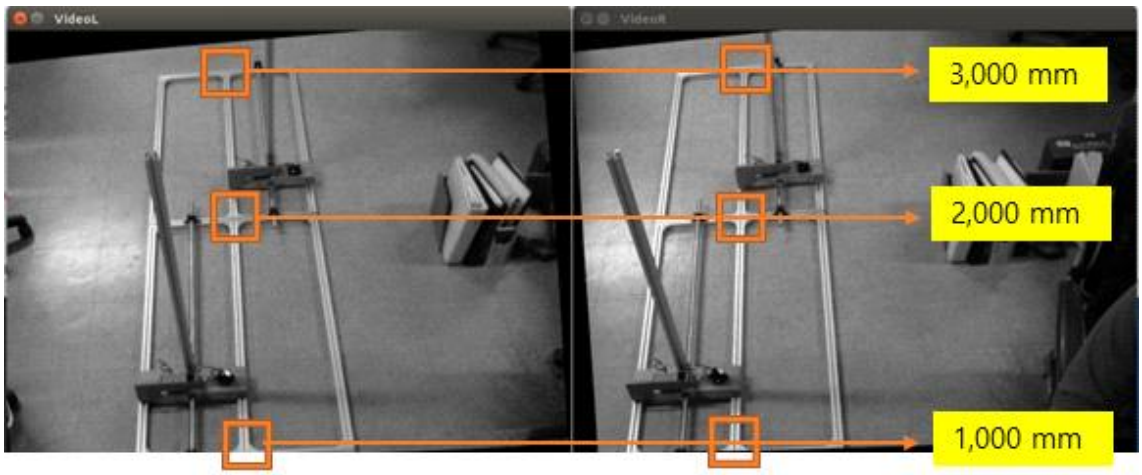
제작한 테스트벤치의 측정가능 영역은 Table. 3-6과 같다. 카메라의 각도를 45° 아래로 향하여 X축 1,000 ~ 3,000 mm, Y축 0 ~ 690 mm, Z축 0 ~ 1080 mm 영역을 측정할 수 있다.

Table. 3-6 Test bench operation distance

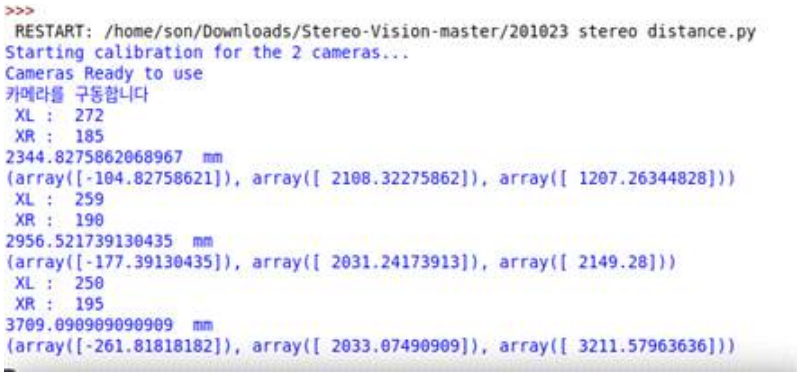
축	측정 거리
X-axis	1,000 mm, 2,000 mm, 3,000 mm
Y-axis	0 mm
Z-axis	200 mm

3) 시험 결과

제작한 테스트벤치를 이용하여 거리측정 및 영상 확인을 위해 Lab Test 결과는 Fig. 3-9와 Table. 3-8과 같다.



(a) 거리 측정 과정 화면



(b) 거리 측정 결과 화면

Fig. 3-9 테스트벤치 거리측정 결과

캡처한 사진은 카메라 거리 측정 시스템을 작동시킨 결과로 나타난 화면이며, Fig. 3-9 (a)와 같이 좌, 우 카메라를 통한 영상 정보를 작업자에게 보여준다. 이 영상을 통해 작업자는 사물을 지정하고, 좌, 우 X좌표의 차이를 통해 시차를 구하고 사물의 거리 값을 계산 할 수 있다.

스테레오 카메라 시스템을 통해 측정한 거리와 목표물 거리를 비교한 결과는 Table. 3-7과 같다. 비교 결과 타겟의 거리가 멀어짐에 따라 오차가 증가한다.

Table. 3-7 테스트벤치 거리측정 결과

Case	Target Distance [mm]	Measure Distance [mm]	Error [mm]
1	1,000	0,983	-17
2	2,000	2,013	13
3	3,000	3,002	2

오차의 원인은 작업자가 좌, 우 영상을 보고 물체를 지정하기 때문에 좌, 우 영상에서의 사물의 위치가 정확히 일치할 수 없기에 오차가 발생하는 것을 볼 수 있고, 작업자가 사물을 지정할 때, 작업자의 선택을 도와줄 수 있는 시스템을 추가할 것이고 향후 실차 테스트를 할 때 사물을 지정함에 있어서 사물을 잘 지정할 수 있도록 프로그램 코드를 보강할 것이다.

이를 통해 본 연구의 알고리즘은 이전 연구의 장비에 비해 저가 장비를 사용하였고, 거리 측정 알고리즘을 간소화하여 응답속도를 빠르게 만들었음에도 거리 측정의 정확도가 유사함을 볼 수 있다.

4. 실차 실험 및 결과

4.1. 실차 실험

1) 시스템 구성

스테레오 카메라와 라이다의 거리 측정 시스템을 실제 차량에 장착하였고 형상은 Fig. 4-1과 같다. 원격에서 작업자가 사물을 인지하고 지정하면 사물에 대한 거리정보와 좌표를 얻을 수 있다.

획득한 거리와 좌표정보를 역방향 기구학적 모델링에 입력하여 굴착기의 작업기 끝단이 사물에 도달하기 위한 붐, 암, 버켓의 각도 값을 구한다. 현재 굴착기의 각 관절 데이터 값은 굴착기 작업기의 관절에 부착된 포텐셔미터를 통해 측정하며 굴착기의 현재 각도와 목표물에 도달하기 위해 구한 각도 값을 비교하여 아두이노를 통해 동작 각도 값을 계산한다. 계산된 값은 전자 비례 감압밸브로 전달하여 굴착기를 동작한다.

스테레오 카메라를 통해 안전 콘을 인식하고 거리 측정 계산식을 통하여 거리와 좌표 값을 계산하고, 역기구학을 통해 사물에 도달할 때 필요한 각도 값을 아두이노로 전송한다. Seiral data로 전송받은 각도 값과 현재 굴착기의 붐, 암, 버켓 각도 값을 비교하여 아두이노와 연결되어있는 전자 비례 감암밸브에 PWM(Pulse With Modulation) 신호를 보낸다.

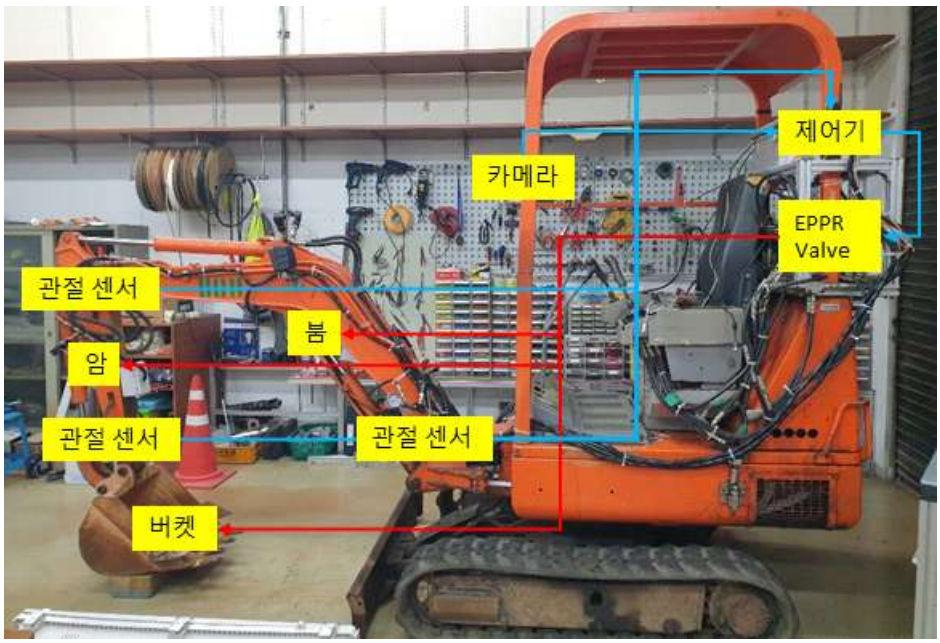


Fig. 4-1 전체 거리 측정 시스템

Fig. 4-2와 같이 제작한 카메라 시스템을 굴착기 붐, 암, 버켓의 영향을 받지 않고 작업 영역을 볼 수 있는 위치에 카메라 시스템을 장착했다.

장착 위치는 작업자가 굴착기에 탑승하였을 때, 눈높이에 맞추어 설치하였고, 부착된 서보 모터를 통해 상, 하로 카메라가 움직이며 굴착기 전방을 확인할 수 있다.

Fig. 4-2 굴착기의 카메라 시스템

2) 좌표 변환

Fig. 4-3과 같이 스테레오 카메라의 좌표계가 굴착기 좌표계와 떨어져 따로 존재하기 때문에, 스테레오 카메라를 통해 얻은 사물의 좌표를 굴착기의 초기 위치로 변환하는 공식을 사용하여 사물의 좌표 값을 변환해 주어야 굴착기가 사물에 도달하기 위한 작업기 각도를 얻을 수 있다. 카메라 좌표계에서 굴착기 좌표계로 좌표를 변환하는 식은 식 (1)과 같다.

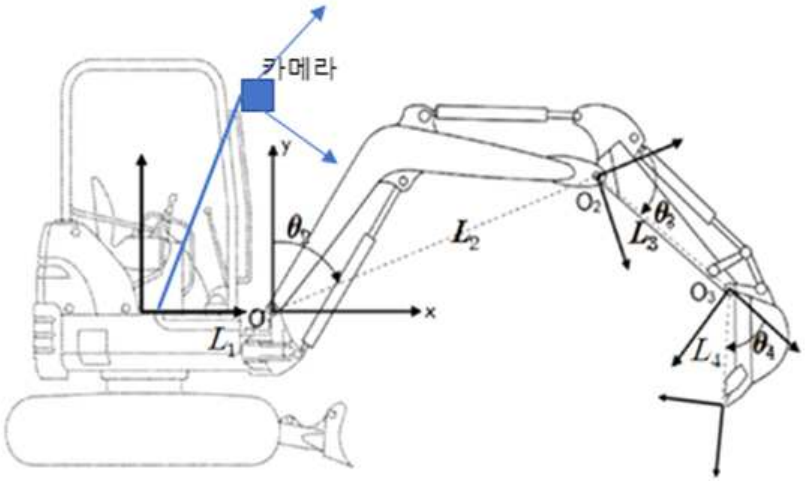


Fig. 4-3 카메라 및 굴착기 좌표계

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \begin{bmatrix} \cos(p) & \sin(p) & 0 \\ -\sin(p)\sin(t) & \cos(p)\sin(t) & -\cos(t) \\ -\sin(p)\cos(t) & \cos(p)\cos(t) & \sin(t) \end{bmatrix} \left(\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} - \begin{bmatrix} X_1 \\ Y_1 \\ Z_1 \end{bmatrix} \right) \quad (1)$$

식 (1)에서 $[X_c, Y_c, Z_c]$는 카메라를 통한 사물의 좌표이고 $[X, Y, Z]$는 월드좌표계, $[X_1, Y_1, Z_1]$는 카메라좌표계이다.

월드좌표계와 카메라좌표계의 좌표 이동 값을 보정한 후, 회전변환 공식에 따라 좌표 값을 변환하면 카메라를 통한 사물의 좌표 값을 굴착기의 초기위치 기준 월드 좌표 값을 얻을 수 있다.

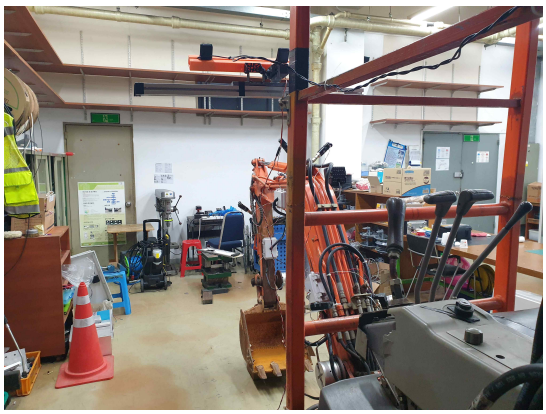
3) 시험방법 및 조건

건설현장에서 많이 사용하는 안전 콘을 이용하여 거리를 측정하는 실험을 진행하였다. 안전 콘을 1,400 mm, 1,500 mm, 1,900 mm, 2,300 mm에 두고 측정하였을 때, 안전 콘을 인지하고 거리 값과 좌표 값을 측정하였다. 측정된 거리 값의 오차가 50 mm이내이고, 계산된

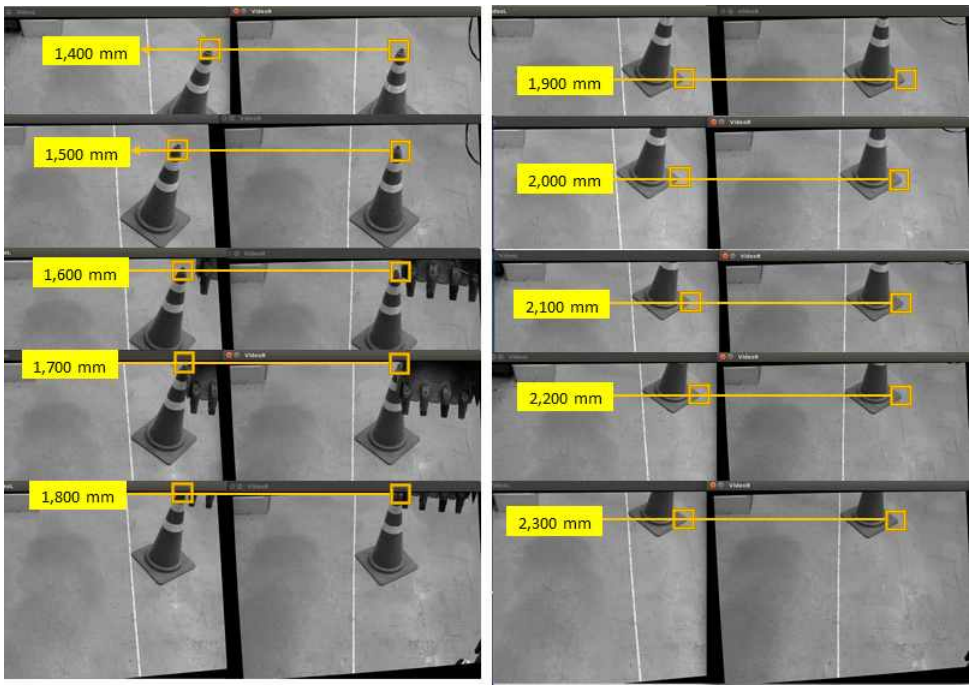
각도로 구동하면 굴착기의 끝단이 안전 콘에 도달함을 실험한다.

4) 시험 결과

제작한 거리 측정 시스템을 이용하여 거리측정 및 영상 확인한 결과는 Fig. 4-6과 Table. 4-1 과 같다.



(a) 실차 거리 측정



(b) 거리 측정 영상

Fig. 4-4 굴착기 거리측정 결과

캡처한 사진은 카메라 거리 측정 시스템을 작동시킨 결과로 나타난 화면이며, Fig. 4-4 (b)와 같이 좌, 우 카메라를 통한 영상 정보를 작업자에게 보여준다. 이 영상을 통해 작업자는

사물을 지정하고, 좌, 우 좌표의 차이를 통해 시차를 구하고 사물의 거리 값을 계산 할 수 있다. 안전 콘의 위치에 따른 결과 값은 Table 4-1과 같다.

Table. 4-1 굴착기 거리 측정 결과

Case	Target Distance [mm]	Measure Distance [mm]	Error [mm]
1	1,400	1,500	100
	1,400	1,499	99
	1,400	1,495	95
	1,400	1,528	128
2	1,500	1,571	71
	1,500	1,598	98
	1,500	1,604	104
	1,500	1,598	98
3	1,600	1,691	91
	1,600	1,660	60
	1,600	1,614	14
	1,600	1,645	45
4	1,700	1,732	32
	1,700	1,752	52
	1,700	1,785	85
	1,700	1,756	56
5	1,800	1,815	15
	1,800	1,819	19
	1,800	1,890	90
	1,800	1,862	62
6	1,900	1,902	02
	1,900	1,978	78
	1,900	1,923	23
	1,900	1,996	96
7	2,000	2,138	138
	2,000	2,100	100
	2,000	2,079	79
	2,000	2,020	20
8	2,100	2,212	112
	2,100	2,157	57
	2,100	2,217	17
	2,100	2,159	59
9	2,200	2,299	99
	2,200	2,339	139
	2,200	2,260	60
	2,200	2,400	200
10	2,300	2,397	97
	2,300	2,474	174
	2,300	2,425	125
	2,300	2,334	34

Fig. 4-5와 같이 굴착기 정해진 위치에 도달하였으며, 동작 결과는 (a), (b), (c), (d)와 같고, 측정 좌표 값은 Table. 4-2와 같다.

(a) X : 1400 Y : 10 Z : 0

(b) X : 1500 Y : 10 Z : 300

(c) X : 1900 Y : 10 Z : 300

(d) X : 2300 Y : 10 Z : 0

Fig. 4-5 굴착기 시스템 동작 결과

측정된 사물의 좌표를 굴착기 동작 시스템에 대입하여 구동한 결과 Table 4-2와 같이 지정된 사물에 굴착기가 도달한 모습을 볼 수 있다.

Table. 4-2 굴착기 거리 측정 결과

Case		결과 값	오차 값
(a)	*X*: 1,400 *Y*: 10 *Z*: 0	*X*: 1,432 *Y*: 20 *Z*: 118	*X*: 32 *Y*: 10 *Z*: 118
(b)	*X*: 1,500 *Y*: 10 *Z*: 300	*X*: 1,539 *Y*: 31 *Z*: 131	*X*: 39 *Y*: 21 *Z*: -169
(c)	*X*: 1,900 *Y*: 10 *Z*: 300	*X*: 1,958 *Y*: 76 *Z*: 233	*X*: 58 *Y*: 66 *Z*: -67
(d)	*X*: 2,300 *Y*: 10 *Z*: 0	*X*: 2,268 *Y*: 582 *Z*: 95	*X*: -32 *Y*: 572 *Z*: 95

하지만 Case (b)와 Case (c)를 보았을 때, 두 경우 모두 안전 콘의 상단부를 지정하였고 두 가지의 각도 값이 도출되었다. 한 가지는 (b)는 버켓이 접혀 들어간 상태로 안전 콘의 상단부에 도달하였고, (c)는 버켓이 펴져서 안전 콘의 상단부에 도달함을 볼 수 있다. 그 이유는 굴착기 버켓이 지표면과 이루는 각도를 지정하지 않고 역방향 기구학적 모델링을 진행하였기에 역방향 기구학적 모델링이 가능한 모든 경우의 수를 계산하는 것이고, 그 결과로 같은 지점을 선택했지만 두 가지의 각도 결과 값이 나온 것이다.

이런 상황을 줄이기 위해서 일반적으로 사용하는 방식은 버켓의 끝단이 지표면과 이루는 각도를 미리 지정해두고 계산식에 미리 대입한다. 그러면 버켓의 각도가 정해져 있으며 굴착기가 사물에 도달하기 위한 붐, 암, 버켓의 관절 각도 계산에 시간이 줄어드는 장점이 있다. 이를 통해 작업자의 조종 없이 카메라를 통해 사물을 지정하면 굴착기가 자동으로 사물에 도달하는 시스템을 확인했다.

5. 결론 및 향후 계획

본 논문에서는 굴착기를 작업자의 조종 없이 작업기 붐, 암, 버켓이 움직이는 시스템을 제안하였다. 굴착기의 순방향 기구학적 모델링과 역방향 기구학적 모델링을 계산하여 작업자가 굴착기에 원하는 각도를 지정하면 굴착기가 지정된 각도로 붐, 암, 버켓을 움직였으며 작업자가 원하는 좌표를 지정하면 굴착기가 지정된 좌표로 붐, 암, 버켓을 움직이는 결과를 확인하였다.

그리고 본 논문에서는 자동굴착을 위한 물체 거리 측정 시스템을 제안하였다. 스테레오 카메라를 이용하여 원격에서 굴착기의 주변 환경을 인식하고 물체의 거리를 측정하여 굴착기를 원격 조종하는 시스템을 구현하였다. 스테레오 카메라를 통해 거리를 측정하기 위해 카메라 내부와 외부의 정보를 얻는 작업인 카메라 캘리브레이션을 진행하고, 측정된 거리 값의 정확도를 향상시키는 결과를 확인하였다.

거리 측정 시스템을 통해 측정된 거리 정보와 좌표정보를 굴착기 역기구학 계산을 통하여 작업자가 지정한 사물에 도달하기 위한 굴착기 붐, 암, 버켓의 필요 관절 각도를 구하고 이를 통해 굴착기를 무인 자동화 할 수 있는 시스템을 구축하였다. 원격 조종 시스템을 통해 작업자가 사물을 지정하면 굴착기가 사물에 도달하였고, 간단한 작업은 자동으로 할 수 있는 결과를 확인하였다.

향후 연구를 계속하여 거리 측정 센서인 라이다와 초음파 센서를 추가로 설치하여, 거리 측정의 정확도를 향상시킬 것이고, 인공지능과 빅데이터를 적용하여 작업자가 인지하고 지정한 사물의 거리 뿐 아니라 환경인식과 동시에 모든 사물의 거리 값을 실시간으로 계산하는 시스템을 추가할 것이다.

Reference

[1] S.W. Choe et al. “A Study on the Technology Trend of Construction Machinery Intelligence by the Fourth Industrial Revolution”, KSFC spring conference, pp. 54-57. 2018

[2] S. W. Choi et al. “A Review on the Sensor System of the Next Generation Construction Machinery”, Journal of Drive and control, Vol. 15 No.3, pp. 80-88, 2018

[3] S. J. Na .et al. “ A Study on Remote System of Excavator Using IoT”. KSFC Autumn conference, pp. 93-98, 2019

[4] T. G. Son et al. “A Study on the Object Distance Measuring System for Automatic Excavation”. KSFC spring conference, pp. 92-93. 2020

[5] T. H.. Lim et al. “A Study on Development of Automatic Excavator system and Veryfy the System “. KSFC conference, pp. 45-49. 2011

[6] Q. H. Le et al. “A Study on Trajectory Tracking Control of Field Robot”. KSFC conference, pp. 119-123, 2016

[7] Q. H. Le et al. “Remote control of excavator using head tracking and flexible monitoring method”, Automation in Construction, Vol.81, pp. 99-111, (2017)

[8] Abdelmoghit Zaarane, Ibtissam Slimani, Wahban Al Okaishi, Issam Atouf, Abdellatif Hamdoun “Distance measurement system for autonomous vehicles using stereo camera”, Array 5, pp. 100016, 2020

[9] 김치승, 유제연, 염승훈, 강민정, 이충호, 김진환, 허욱열 “Distance Measurement using Stereo Vision”, 대한전기학회 학술대회 논문집, pp. 472-472, 2008

[10] 김치승, 유제연, 염승훈, 강민정, 이충호, 김진환, 허욱열, “스테레오 비전을 이용한 거리 측정”, 대한전기학회 학술대회 논문집, pp. 1843-1844, 2008

[11] 김승태, 이종훈, 김도성, 이명호, “카메라 영상에 의한 물체와의 거리 측정에 관한 연구” 대한전기학회 학술대회 논문집, pp. 415-420, (1989)

[12] 조재현, 이영진, 유준, “스테레오 비전을 이용한 거리측정 알고리즘 구현” 제어로봇시스템학회 국내학술대회 논문집, pp. 633-636, 2007

[13] 이강민, 나산마로, 박재훈, 안기원, 이치범 “스테레오비전과 6축 매니퓰레이터를 이용한 양팔 서비스 로봇 개발”, 한국생산제조학회지, 27(3), pp. 271-277, 2018

[14] Mingchang Zhao, Abdelhamid Mammeri and Azzedine Boukerche, “Distance Measurement System for Smart Vehicles”, Department of Electrical Engineering and Computer science, 2015

[15] C. S. Kim et al. “Distance Measurement using Stereo Vision”, 2008 Summer conference of KIEE, pp. 1843-1844, 2008

부 록

Ⅰ. 논문 발표 실적

- 손태곤, 최성웅, 레쾅호안, 김경동, 장고기, 양순용, “원격 거리 측정 시스템 모듈 개발에 대한 연구”, 유공압건설기계학회 학술대회논문집, 투고 중

- 손태곤, 레쾅호안, 염영진, 양순용, “자동 굴착을 위한 물체 거리측정 시스템에 대한 연구”, 유공압건설기계학회 학술대회논문집, pp. 92-93, 2020

- 김영재, 김태성, 손태곤, 양순용, “6축 굴삭기의 3D 시뮬레이터 개발에 관한 연구”, 유공압건설기계학회 학술대회논문집, pp. 87-88, 2019

- 손태곤, 나선준, 서보문, 양순용, “무선 통신을 이용한 건설기계용 데이터 수집 장치 개발에 관한 연구”,대한기계학회 춘추학술대회, pp. 15-15, 2019

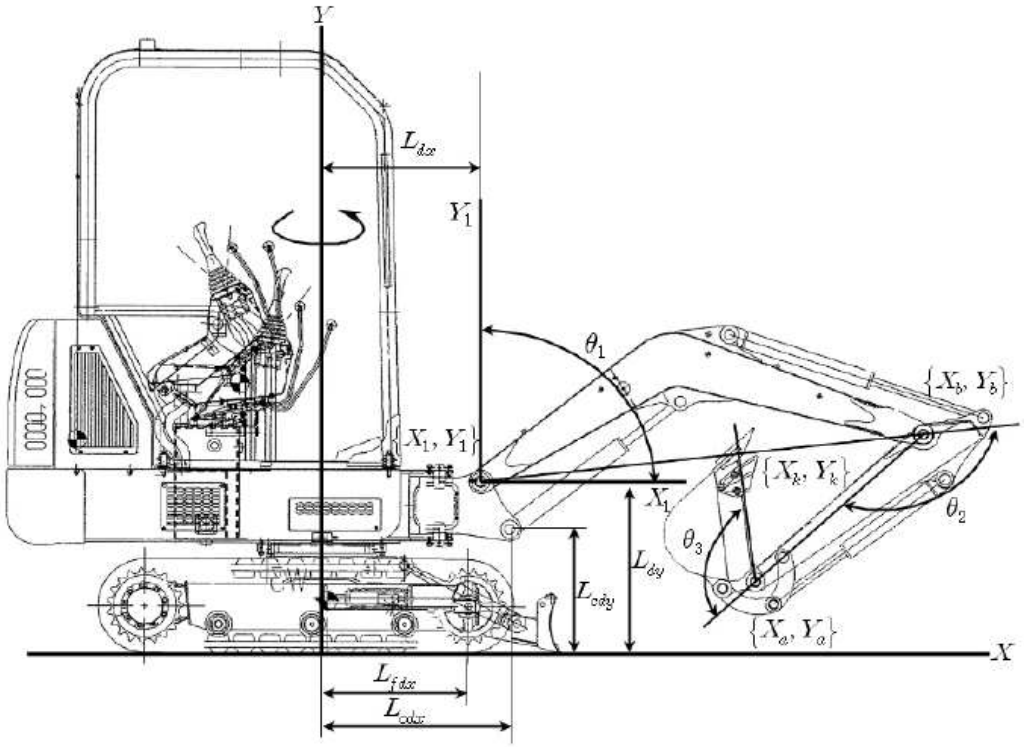
- 나선준, 서보문, 손태곤, 양순용, “RF 통신을 이용한 저가형 건설기계 맞춤 데이터 수집 장치 개발에 관한 연구”, 유공압건설기계학회 학술대회논문집, pp. 287-288, 2019

- G. Q. Zhang, S. L. Wang, T. G. Son, Y. S. Kim, T. S. Li, Y. J. Yun, Y. H. Gwak, S. Y.Yang, “A Study on the Simulation Analysis for the Orbital Riveting Process”, 한국정밀공학회 학술발표대회 논문집, pp. 102-102, 2019

- 손태곤, 최성웅, 레쾅호안, 양순용, “소형 굴착기의 원격 물체 거리 측정 및 자동 굴착 시스템에 관한 연구”, 드라이브・컨트롤, 투고 중

- 최성웅, 레쾅호안, 손태곤, 양순용, “소형 필드로봇의 무선 원격 제어를 위한 조종시스템 구축에 관한 연구”, 드라이브・컨트롤, 투고 중

Ⅱ. 1.5ton 굴착기 제원



명칭	부위	기호	크기	명칭	부위	기호	크기
본체	엔진커버부	W_1	85[kg]	본체		L_{frg}	-0.198[m]
	상부회전체	W_2	645[kg]			L_{btg}	0.606[m]
	트랙	W_3	610[kg]			L_{rlg}	0.0395[m]
		L_1	-0.925[m]	Boom	무게	m_b	
		L_2	-0.312[m]		끝단	L_b	42.865[kg]
		L_3	0.024[m]		무게중심	l_{bg}	0.823[m]
		L_4	0.082[m]		무게중심각	δ_{bg}	9[°]
		L_5	0.925[m]			X_b	-
		h_1	0.817[m]			Y_b	-
		h_2	1.037[m]			X_{bg}	-
		h_3	0.122[m]			Y_{bg}	-
		L_{dx}	0.596[m]	Bucket	무게	m_k	45.0[kg]
		L_{dy}	0.66[m]		끝단	L_k	0.478[m]

Bucket	무게중심	l_{kg}	0.247[m]	Arm Cyl		l_{a2}	0.248[m]
	무게중심각	δ_{kg}	40.21[°]		계산값	l_{a3}	-
		X_k	-			δ_{a1}	-
		Y_k	-			δ_{a2}	-
		X_{kg}	-		계산값	β_a	-
		Y_{kg}	-		계산값	α_a	-
Arm	무게	m_a	15.079[kg]		계산값	α_{a1}	-
	끝단	L_a	0.851[m]		계산값	γ_a	-
	무게중심	l_{ag}	0.253[m]			X_{acg}	-
	무게중심각	δ_{ag}	13.8[°]			Y_{acg}	-
		X_k	-		무게	m_{kcy}	9.57[kg]
		Y_k	-		압축	L_{kcy}	0.52[m]
		X_{kg}	-		인장	L_{kcy}	0.82[m]
		Y_{kg}	-		압축	l_{kcg}	0.265[m]
Boom Cyl	무게	m_{bcy}	10.722[kg]	Bucket Cyl	인장	l_{kcg}	0.381[m]
	압축	L_{bcy}	0.62[m]			l_{k1}	0.684[m]
	인장	L_{bcy}	1.0[m]			l_{k2}	0.18[m]
	압축	l_{bcg}	0.317[m]			l_{k3}	0.20[m]
	인장	l_{bcg}	0.469[m]			l_{k4}	0.138[m]
		l_{b1}	0.825[m]			l_{k5}	0.13[m]
		l_{b2}	0.216[m]			l_{k6}	0.821[m]
		δ_{b1}	16[°]			l_{k7}	-
		δ_{b2}	32.15[°]			l_{dx}	-
	계산값	β	-			l_{dy}	-
	계산값	α	-			δ_{k1}	7.24[°]
	계산값	θ_{bcg}	-			δ_{k2}	171.9[°]
		X_{bcg}	-			δ_{k3}	120.66[°]
		Y_{bcg}	-			δ_{k4}	-
Arm Cyl	무게	m_{acy}	6.05[kg]			α_{k1}	-
	압축	L_{acy}	0.62[m]			β_{k1}	-
	인장	L_{acy}	1.01[m]			β_{k11}	-
	압축	l_{acg}	0.31[m]			β_{k12}	-
	인장	l_{acg}	0.49[m]			β_{k2}	-
		l_{a1}	0.852[m]			γ_k	-

Bucket Cyl		γ_{k1}	-	Bucket Link		α_{l2}	-
		γ_{k2}	-			β_{k31}	-
		γ_{k3}	-			θ_{kl1}	-
		γ_{k4}	-			θ_{kl2}	-
		γ_{k5}	-			X_{klg1}	-
		θ_{kcg}	-			Y_{klg1}	-
		X_{kcg}	-			X_{klg2}	-
		Y_{kcg}	-			Y_{klg2}	-
Bucket Link	링크 1	m_{kl1}	2.326[kg]		붐 각도	θ_1(min)	147.95[°]
	링크 2	m_{kl2}	6.958[kg]			θ_1(max)	23.968[°]
	링크 1	l_{klg1}	0.092[m]		암 각도	θ_2(min)	24.489[°]
	링크 2	l_{klg2}	0.1017[m]			θ_2(max)	133.44[°]
		δ_{k5}	11.31[°]		버켓 각도	θ_3(min)	-10.69[°]
		δ_{k6}	10.44[°]			θ_3(max)	127.54[°]
		l_{kl1}	-			L_{fdx}	0.568[m]
		l_{kl2}	-			L_{cdx}	0.728[m]
		α_{1g1}	-			L_{cdy}	0.483[m]

Ⅲ. 작동 코드

1) Matlab - Forward Kinematics 코드

```
syms the1 the2 the3 the4 l1 l2 l3 l4 l5 nx ny nz px ox oy oz py ax ay az pz;

T01=[cos(the1) 0 sin(the1) l1*cos(the1);
     sin(the1) 0 -cos(the1) l1*sin(the1);
     0 1 0 0;
     0 0 0 1]
T12=[cos(the2) -sin(the2) 0 l2*cos(the2);
     sin(the2) cos(the2) 0 l2*sin(the2);
     0 0 1 0;
     0 0 0 1];
T23=[cos(the3) -sin(the3) 0 l3*cos(the3);
     sin(the3) cos(the3) 0 l3*sin(the3);
     0 0 1 0;
     0 0 0 1];
T34=[cos(the4) -sin(the4) 0 l4*cos(the4);
     sin(the4) cos(the4) 0 l4*sin(the4);
     0 0 1 0;
     0 0 0 1];

T02= T01*T12
T03= T01*T12*T23;
T03= simplify(T03)
T04= T01*T12*T23*T34;
T04= simplify(T04)
P= [nx ox ax px;
     ny oy ay py;
     nz oz az pz;
     0 0 0 1];
TP= T01\P;
TP= simplify(TP)
T14= T12*T23*T34;
T14= simplify(T14)
```

2) Matlab - Inverse Kinematics 코드

```
l1= 500;
l2= 1705;
l3= 851;
l4= 300;

the1= atan2(py,px)

s234= -ox*cos(the1)-oy*sin(the1);
c234= nx*cos(the1)+ny*sin(the1);
the234= atan2(s234,c234);

X1= px*cos(the1)-l1+py*sin(the1)-l4*cos(the234);
X2= pz-l4*sin(the234);

c3= (X1^2+X2^2-l2^2-l3^2)/(2*l2*l3);
s3= -sqrt(1-c3^2);
%s3= X2*cos(the2)-X1*sin(the2)/l3;
the3= atan2(s3,c3)

k1= l2+l3*cos(the3);
k2= l3*sin(the3);
the2= -atan2(k2,k1)+atan2(X2,X1)

the4= the234-the2-the3
```

3) Arduino - 굴착기 구동 코드

```
int boomAngSenPin = A6;
int armAngSenPin = A7;
int bucketAngSenPin = A8;

int boomAngSen, armAngSen, bucketAngSen;
float boomAng, armAng, bucketAng;
double pwmBoomPin = 3;
double pwmArmPin = 6;
double pwmBucketPin = 5;

String sig;
char Ard1[4] = {0};
char Ard2[4] = {0};
char Ard3[4] = {0};
char Ard4[4] = {0};
char check{1};
int the1 = 0;
int the2 = 0;
int the3 = 0;
int the4 = 0;

void setup() {
// put your setup code here, to run once:
Serial.begin(9600);
//   pinMode(A6, INPUT);
//   pinMode(A7, INPUT);
//   pinMode(A8, INPUT);
//   pinMode(10, OUTPUT);
//   pinMode(9, OUTPUT);
//   pinMode(8, OUTPUT);

}

void loop() {

//while(Serial.available()){
//   char ch= Serial.read();
```

```
//    sig.concat(ch);
//
//    sig.substring(0,1).toCharArray(check,2);
//    if(check[0]== 'Q')
//    {
//       if(sig.length()==21)
//       {
//          sig.substring(1,4).toCharArray(Ard1,4);
//          sig.substring(6,9).toCharArray(Ard2,4);
//          sig.substring(11,14).toCharArray(Ard3,4);
//          sig.substring(16,19).toCharArray(Ard4,4);
//          the1= atoi(Ard1)-100;
//          the2= atoi(Ard2);
//          the3= atoi(Ard3);
//          the4= atoi(Ard4)-100;
//          sig= "";
//       }
//          else if(sig.length() > 21)
//          {
//             sig= "";
//          }
//    }
//}

boomAngSen = analogRead(boomAngSenPin);
 // max angle 353 (up); zero anle value  at 560 (down)
armAngSen = analogRead(armAngSenPin);
 // max angle at 932 (Open); min angle at 605 (close)
bucketAngSen = analogRead(bucketAngSenPin);
 // max value  at 215 (open), min close at 531 (close)

#boomAng = 530; //353-572
#armAng =  616; //604-932
#bucketAng = 273; //220-530

Serial.print("BoomAng : ");
Serial.print(boomAngSen);
Serial.print("/");
Serial.print(boomAng);
```

```
Serial.print("      ArmAng : ");
Serial.print(armAngSen);
Serial.print("/");
Serial.print(armAng);
Serial.print("      BuckAng : ");
Serial.print(bucketAngSen);
Serial.print("/");
Serial.println(bucketAng);
//boomErr=boomAng-boomAngSen;

if (boomAngSen - boomAng> 3) {
  analogWrite(pwmBoomPin, 80);
}
else if (boomAngSen - boomAng< -3) {
  analogWrite(pwmBoomPin, 150);
}
else analogWrite(pwmBoomPin, 127);

if (armAngSen - armAng> 3) {
  analogWrite(pwmArmPin, 80);
}
else if (armAngSen - armAng< -3) {
  analogWrite(pwmArmPin, 150);
}
else analogWrite(pwmArmPin, 127);

if (bucketAngSen - bucketAng> 3) {
  analogWrite(pwmBucketPin, 150);
}
else if (bucketAngSen - bucketAng< -3) {
  analogWrite(pwmBucketPin, 80);
}
else analogWrite(pwmBucketPin, 127);
}
```

4) Python - 카메라 작동 및 사물 인식 코드

```
import numpy as np
import cv2
import math
import serial

ser = serial.Serial(port='/dev/ttyACM0',baudrate=9600)

def map(x,input_min,input_max,output_min,output_max):
    return (x-input_min)*(output_max-output_min)/(input_max-input_min)+output_min
#map()함수 정의.

def click_eventL(event, xl, y, flags, param):
    global XL, YL
    if event == cv2.EVENT_LBUTTONDOWN:
        XL= xl
        YL= y
        print(" XL : ",xl)

def click_eventR(event, xr, y, flags, param):
    global XR
    if event == cv2.EVENT_LBUTTONDOWN:
        XR= xr
        print(" XR : ",xr)

def showVideo():
    try:
        print('카메라를 구동합니다')
        capR= cv2.VideoCapture(1)
        capL= cv2.VideoCapture(0)

    except:
        print('카메라 구동 실패')
        return
    capR.set(3,640)
    capR.set(4,480)
    capL.set(3,640)
    capL.set(4,480)
```

```
    while True:
        retR, frameR= capR.read()
        retL, frameL= capL.read()

        if not retR :
            print('비디오 읽기 오류')
            break

 # Rectify the images on rotation and alignement
        Left_nice= cv2.remap(frameL,Left_Stereo_Map[0],Left_Stereo_Map[1],
cv2.INTER_LANCZOS4, cv2.BORDER_CONSTANT, 0)
 # Rectify the image using the kalibration parameters founds during the initialisation
        Right_nice= cv2.remap(frameR,Right_Stereo_Map[0],Right_Stereo_Map[1],
cv2.INTER_LANCZOS4, cv2.BORDER_CONSTANT, 0)

        grayR= cv2.cvtColor(Right_nice,cv2.COLOR_BGR2GRAY)
        grayL= cv2.cvtColor(Left_nice,cv2.COLOR_BGR2GRAY)

        # Compute the 2 images for the Depth_image
        #disp= stereo.compute(grayL,grayR)#.astype(np.float32)/ 16

        #disp= ((disp.astype(np.float32)/ 16)-min_disp)/num_disp
        #Calculation allowing us to have 0 for the most distant object able to detect
        #dispR= cv2.resize(disp,None,fx=0.7, fy=0.7, interpolation = cv2.INTER_AREA)

        #cv2.imshow('Depth map',disp)
        #cv2.imshow('VideoR',frameR)
        #cv2.imshow('VideoL',frameL)
        cv2.imshow('VideoR',grayR)
        cv2.imshow('VideoL',grayL)

        cv2.setMouseCallback('VideoL', click_eventL)
        cv2.setMouseCallback('VideoR', click_eventR)

        k= cv2.waitKey(1) & 0xFF
        if k ==27:
            break
        elif k == ord('s'):
```

```
global A,B,C,D,op

disparity= XL - XR
distance= 240*850/disparity
print(distance,“ mm“)
(RX,RY)= ((XL-310)*distance/850,(YL-240)*distance/850)
RZ= distance
R= np.array([[1,0,0],[0,0.707,0.707],[0,-0.707,0.707]],np.float)
XYZ= np.array([[RX],[RY],[RZ]])
(RRX,RRY,RRZ)= R.dot(XYZ)
#print(R)
print((RRX-500,RRY-1100,RRZ))

px= RRZ
py= RRY-1100
pz= RRX-500

l1= 500
l2= 1705
l3= 851
l4= 300

the1= math.atan2(py,px)

ox= math.cos(the1)
oy= math.sin(the1)
nx= 0
ny= 0
s234= -ox*math.cos(the1)-oy*math.sin(the1)
c234= nx*math.cos(the1)-ny*math.sin(the1)
the234= math.atan2(s234,c234)

X1= px*math.cos(the1)-l1+py*math.sin(the1)-l4*math.cos(the234)
X2= pz-l4*math.sin(the234)

c3= (X1**2+X2**2-l2**2-l3**2)/(2*l2*l3)
s3= -math.sqrt(abs(1-c3**2))
the3= math.atan2(s3,c3)
```

```
            k1= l2+l3*math.cos(the3)
            k2= l3*math.sin(the3)
            the2= math.atan2(X2,X1)-math.atan2(k2,k1)

            the4= the234-the2-the3

            the1= the1*180/math.pi
            the3= the3*180/math.pi
            the2= the2*180/math.pi
            the4= the4*180/math.pi
            print(the1,+the2,+the3,+the4)

            tthe1= int(map(the1,-180,180,100,460))
            tthe2= int(map(the2,-57,66,351,572))
            tthe3= int(map(the3,-133,-24,604,932))
            tthe4= int(map(the4,-127,10,100,411))

            print(tthe1-100)
            print(tthe2)
            print(tthe3)
            print(tthe4-100)

            A= str(tthe1)
            B= str(tthe2)
            C= str(tthe3)
            D= str(tthe4)

            op = "Q"+A+B+C+D

        elif k == ord('t'):
            ser.write(op.encode())
            print(op)

    cv2.destroyAllWindows()

# Termination criteria
criteria =(cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 30, 0.001)
criteria_stereo= (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 30, 0.001)
```

```
# Prepare object points
objp = np.zeros((9*6,3), np.float32)
objp[:,:2] = np.mgrid[0:9,0:6].T.reshape(-1,2)

# Arrays to store object points and image points from all images
objpoints= []    # 3d points in real world space
imgpointsR= []    # 2d points in image plane
imgpointsL= []

# Start calibration from the camera
print('Starting calibration for the 2 cameras... ')
# Call all saved images
for i in range(0,35):    # Put the amount of pictures you have taken for the
calibration inbetween range(0,?) wenn starting from the image number 0
    t= str(i)
    ChessImaR= cv2.imread('chessboard-R'+t+'.png',0)      # Right side
    ChessImaL= cv2.imread('chessboard-L'+t+'.png',0)      # Left side

    retR, cornersR = cv2.findChessboardCorners(ChessImaR,(9,6),None)
# Define the number of chees corners we are looking for
    retL, cornersL = cv2.findChessboardCorners(ChessImaL,(9,6),None)
# Left side

    if (True == retR) & (True == retL):
        objpoints.append(objp)
        cv2.cornerSubPix(ChessImaR,cornersR,(11,11),(-1,-1),criteria)
        cv2.cornerSubPix(ChessImaL,cornersL,(11,11),(-1,-1),criteria)
        imgpointsR.append(cornersR)
        imgpointsL.append(cornersL)

# Determine the new values for different parameters
# Right Side
retR, mtxR, distR, rvecsR, tvecsR = cv2.calibrateCamera(objpoints, imgpointsR,
                                                        ChessImaR.shape[::-1],None,None)
hR,wR= ChessImaR.shape[:2]
OmtxR, roiR= cv2.getOptimalNewCameraMatrix(mtxR,distR, (wR,hR),1,(wR,hR))

# Left Side
retL, mtxL, distL, rvecsL, tvecsL = cv2.calibrateCamera(objpoints, imgpointsL,
```

```
                                                    ChessImaL.shape[::-1],None,None)
hL,wL= ChessImaL.shape[:2]
OmtxL, roiL= cv2.getOptimalNewCameraMatrix(mtxL,distL,(wL,hL),1,(wL,hL))

print('Cameras Ready to use')

#********************************************
#***** Calibrate the Cameras for Stereo *****
#********************************************

# StereoCalibrate function
flags = 0
flags |= cv2.CALIB_FIX_INTRINSIC
#flags |= cv2.CALIB_FIX_PRINCIPAL_POINT
#flags |= cv2.CALIB_USE_INTRINSIC_GUESS
#flags |= cv2.CALIB_FIX_FOCAL_LENGTH
#flags |= cv2.CALIB_FIX_ASPECT_RATIO
#flags |= cv2.CALIB_ZERO_TANGENT_DIST
#flags |= cv2.CALIB_RATIONAL_MODEL
#flags |= cv2.CALIB_SAME_FOCAL_LENGTH
#flags |= cv2.CALIB_FIX_K3
#flags |= cv2.CALIB_FIX_K4
#flags |= cv2.CALIB_FIX_K5
retS, MLS, dLS, MRS, dRS, R, T, E, F= cv2.stereoCalibrate(objpoints,
                                                          imgpointsL,
                                                          imgpointsR,
                                                          mtxL,
                                                          distL,
                                                          mtxR,
                                                          distR,
                                                          ChessImaR.shape[::-1],
                                                          criteria_stereo,
                                                          flags)
# StereoRectify function
rectify_scale= 0
# if 0 image croped, if 1 image nor croped
RL, RR, PL, PR, Q, roiL, roiR= cv2.stereoRectify(MLS, dLS, MRS, dRS,
                                                 ChessImaR.shape[::-1], R, T, rectify_scale,(0,0))
```

```
# last paramater is alpha, if 0= croped, if 1= not croped
# initUndistortRectifyMap function
Left_Stereo_Map= cv2.initUndistortRectifyMap(MLS, dLS, RL, PL,
                                             ChessImaR.shape[::-1], cv2.CV_16SC2)

# cv2.CV_16SC2 this format enables us the programme to work faster
Right_Stereo_Map= cv2.initUndistortRectifyMap(MRS, dRS, RR, PR,
                                             ChessImaR.shape[::-1], cv2.CV_16SC2)

#*************************************
#***** Parameters for the StereoVision *****
#*************************************

# Create StereoSGBM and prepare all parameters
window_size = 3
min_disp = 0
num_disp = 96-min_disp
stereo = cv2.StereoSGBM_create(minDisparity = min_disp,
    numDisparities = num_disp,
    blockSize = window_size,
    uniquenessRatio = 10,
    speckleWindowSize = 100,
    speckleRange = 2,
    disp12MaxDiff = -1,
    P1 = 8*3*window_size**2,
    P2 = 80*3*window_size**2)

showVideo()
```