자율 주행 RC카의
곡률과 속도에 따른 모터 제어 방법

한국외국어대학교
공과대학
디지털정보공학과

김윤기

2018年 12月

자율 주행 RC카의
곡률과 속도에 따른 모터 제어 방법

위 논문 학사학위 논문으로 제출합니다.
지도교수: 윤 일 동

2018年 12月

대학 : 한국외국어대학교
학과 : 디지털정보공학과
학번 : 201300712
이름 : 김윤기

김윤기의 학사학위 논문을 심사하여
합격으로 판정합니다.

심사위원: (인)

자율 주행 RC카의 곡률과 속도에 따른 모터 제어 방법

요약

본 논문은 라즈베리파이 환경에서 OpenCV를 사용한 RC카의 모터 제어 방법에 대한 연구이다. 영상에서 차선을 검출하고 좌표값을 찾아낸다. 좌표값을 수식에 대입하여 모터의 PWM를 조절하고, 두 바퀴의 속도 차이로 방향을 제어한다. 실험은 곡률에 따라, 초기 속도에 따라 진행하였다. 방향제어가 가장 잘 되는 최선의 조건을 찾아내고 이 조건에서의 방향 제어 수식을 유도하였다. 또한 장애물 인식을 위해 RC카에 초음파센서를 설치하여 정지하는 기능을 추가하였다.

Abstract

This paper is a study on motor control method of RC car using OpenCV in Raspberry pi environment. Detect line in image and find coordinate value. By assigning the coordinate value to the formula, the PWM of the motor is controlled and the direction is controlled by the speed difference of two wheels. Experiments were carried out according to curvature and initial velocity. We find the best condition for best direction control and derive the direction control formula in this condition. In addition, an ultrasonic sensor is installed in the RC car to stop the obstacle.

목차

1. 서론

최근 자율 주행 자동차에 대한 관심이 높아지고 있다. 구글, 애플 등 IT 기업들이 자율 주행 제품 개발에 몰두하고 있다. 또한 현대, 기아 등 자동차 기업들도 무인 자율 주행 자동차 경진대회를 개최하는 등 국내외적으로 무인자동차 기술에 대한 관심이 높아지고 있다[1].

현재 학술 논문에도 자율 주행에 관련된 논문은 많이 나와 있다. 이는 대부분 영상처리와 딥 러닝 기반의 알고리즘 분야에 집중되어있다. 자율 주행을 위해 차선을 검출하는 다양한 알고리즘이 있다[2]. 알고리즘을 활용한 모형 자동차의 자율 주행에 관한 논문도 있지만 중점이 알고리즘에 맞춰져 있어서 RC카의 효과적인 모터 제어에 대한 방법은 나와 있지 않다[3].

캡스톤 설계를 진행하면서 중점을 둔 부분은 소프트웨어적으로 차선을 효과적이고 신속하게 검출하는 알고리즘을 설계하였고 하드웨어적으로 정확한 방향 전환을 목표로 하는 실험과 연구를 하였다. 본인의 역할은 통신 부분과 하드웨어를 중심으로 라즈베리파이와 RC카에서 효과적인 모터 제어 방법과 방향 제어에 영향을 주는 여러 가지 요소에 대하여 실험하였다. 본 논문은 앞서 말한 실험과 연구 과정과 결과를 소개하도록 한다.

2. 시스템 구성

시스템에 사용된 하드웨어는 [표 1]과 같다.

부품명	기능
Raspberry Pi 2	Alphabot과 연결되어 영상처리, 동작 등의 명령을 내림
Alphabot	자동차 차체. 모터드라이버와 라즈베리 인터페이스.
webcam	라즈베리파이와 연결되어 영상을 보냄.
dc모터	Alphabot과 연결되어 바퀴부분의 동작을 담당.
초음파센서	라즈베리파이와 연결되어 앞의 장애물을 감지.
와이파이동글	라즈베리파이가 와이파이에 접속할 수 있도록 함.

[표 1] 시스템 사용 하드웨어

2.1 라즈베리파이 (Raspberry Pi 2)

라즈베리파이는 영국 라즈베리 파이(Raspberry Pi) 재단에서 만든 초소형/초저가

PC이다. Raspberry pi 2는 Raspbian이라는 리눅스 커널 기반 운영 체제가 제공되고, 4개의 USB 포트와 10/100 Mbit/s 이더넷 컨트롤러가 장착되어 있다. 또한 사용자가 추가한 USB 이더넷이나 와이파이 어댑터로 네트워킹이 가능하다[4].

실험에 사용된 RC카 모형에는 라즈베리파이와 아두이노 인터페이스가 포함되어 있고, 라즈베리파이 2를 장착하여 진행하였다. 사용 언어는 Python 3.5이고 openCV 라이브러리를 사용하였다.

라즈베리파이에 영상을 보내기 위해 webcam을 연결하였고, 장애물 감지를 위해 초음파센서, 그리고 무선 네트워크 사용을 위한 와이파이 어댑터가 사용되었다. 사용 사례는 뒤에서 자세히 설명한다.

2.2 Alphabot 구성

Alphabot은 waveshare에서 개발한 Raspberry Pi / Arduino와 호환되는 모바일 로봇 플랫폼이다. 컨트롤러 보드, Raspberry Pi / Arduino를 연결하고 라인 추적 오픈 소스 예제 코드와 결합하면 이제 로봇 추적을 시작할 수 있다. 하지만 새로운 영상처리 알고리즘과 결합하기 위해 추가적인 하드웨어 번경도 진행하였다.

[그림 1] 실험에 사용된 RC카 모형 Alphabot[5]

[그림 1]에서 보이는 2개의 뒷바퀴와 전면 하단의 무게추로 균형을 잡고 움직인다. 리튬 18650 배터리로 전원을 공급하고, 모터 드라이버 인터페이스가 포함되어 있다. 2개의 DC모터를 납땜하여 모터드라이버와 연결하였고, 바퀴의 이음새도 헐거워서 납땜하여 고정시켰다.

3. 기능 및 구현

3.1 SSH 통신

원활한 실험을 위해 putty를 이용하여 PC와 라즈베리파이의 SSH 서버 통신을 구현하였다. 라즈베리파이는 라즈비안의 리눅스 기반의 프로그램이고, 라즈베리파이와 통신할 노트북 PC는 윈도우 기반이므로 Xming[6] 프로그램을 사용하여 윈도우 pc에서 영상 뷰어와 같은 라즈베리파이의 GUI 프로그램을 사용하였다. 현재 통신 조건은 노트북 PC와 라즈베리파이는 같은 WIFI 망에 연결되어 있고 암호화가 되지 않은 WIFI 여야 한다는 단점이 있다.

3.2 모터제어

실험 환경은 다음과 같다.

- 카메라 : Microsoft Lifecam VX-2000
- 영상처리 보드 : Raspberry Pi 2
- 사용 언어 : Python Ver. 3.52
- 영상 정보 : 30프레임, 640X480 해상도
- 모터 정보 : 직경 6.5cm
- RC카 모형 정보 : Waveshare Alphabot

3.2.1 On-Off 방식 바퀴 제어

먼저 방향을 제어하기 위한 방법으로 On-Off 방식 모터 제어를 실험하였다.

모터 드라이버	라즈베리파이
IN1	P12
IN2	P13
ENA	P6
IN3	P20
IN4	P21
ENB	P26

[표 2] 모터드라이버 –
라즈베리파이 인터페이스

[표 2]에서 IN1, IN2, ENA는 좌측 모터의 핀넘버이고, IN3, IN4, ENB는 우측 모

터의 핀넘버이다. 각각 HIGH신호, LOW신호, 모터의 사용을 가능케 하는 ENABLE 신호를 받는 3개의 핀이 있다.

IN1	IN2	IN3	IN4	동작
0	0	0	0	정지
1	0	0	1	전진
0	1	1	0	후진
0	0	0	1	좌회전
1	0	0	0	우회전

[표 3] 모터드라이버 – 자동차의 동작 인터페이스

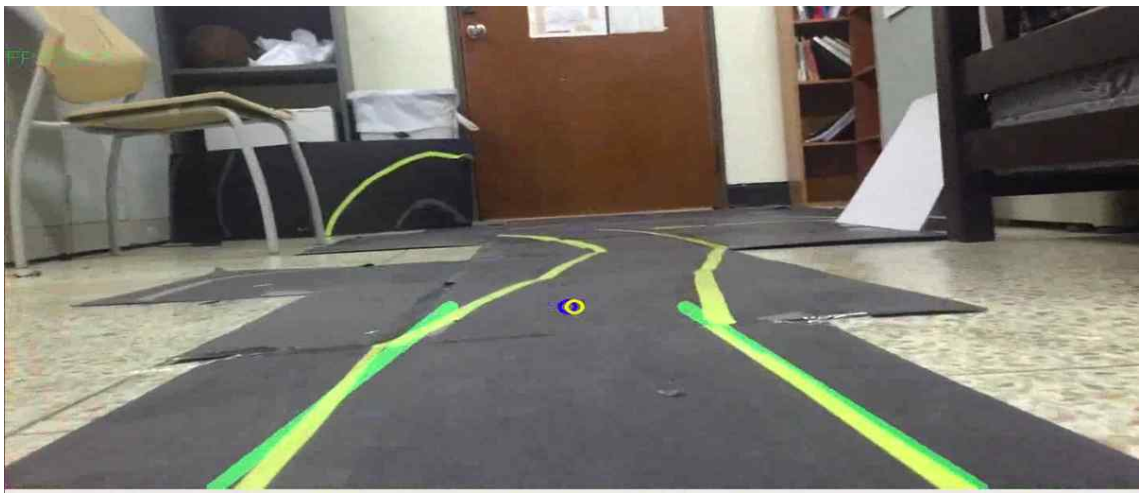
[표 3]에서 IN1은 좌측 모터의 HIGH신호, IN2는 좌측 모터의 LOW신호, IN3는 우측 모터의 LOW신호, IN4는 우측 모터의 HIGH신호이다. 예를 들어 IN1에만 1을 주고 나머지에 0을 준다면 좌측 모터의 HIGH신호가 들어가고, 좌측 바퀴만 전진을 하게 되어 자동차는 우회전을 하게 된다.

3.2.2 PWM 제어를 이용한 방향 제어

앞서 소개한 On-Off 방식으로는 프레임마다 좌회전, 우회전, 전진, 후진만 할 수 있었다. 여기서 좌회전, 우회전은 일반적인 좌회전, 우회전이 아니라 차체만 좌, 우로 회전하는 것이기 때문에 매끄러운 주행이 불가능했다. 따라서 두 바퀴의 속도를 다르게 하여 방향 제어를 실험하였다. 예를 들어 왼쪽 바퀴의 속도가 오른쪽 바퀴의 속도보다 느리다면, 오른쪽 바퀴가 더 많은 거리를 가기 위해서 차체는 왼쪽으로 회전하면서 전진할 것이다.

모터의 속도는 DC 전압을 제어하면 속도를 바꿀 수 있다. 하지만 DC 전압을 손쉽게 제어하기 힘들기 때문에 라즈베리파이의 RPI.GPIO 라이브러리에서 제공하는 함수를 사용하여 PWM 제어를 하였다. dutycycle의 평균값에 해당하는 전류가 DC 모터에 흐르게 되는 것이 PWM으로 속도 제어를 하는 원리이다[7].

[그림 2] 오른쪽으로 꺾인 곡선의 영상처리 결과



[그림 3] 웹캠에서 받아온 영상의 영상처리 결과

[그림 2]에서 시점의 중앙값이 노란 점이고 차선의 중앙값이 파란 점이다. 이 두 좌표의 x값 차이로부터 곡률을 계산한다. 두 좌표의 차이가 많이 날수록 곡률이 심한 곡선이다. 즉, 두 좌표의 차이와 곡률 사이에는 어느 정도 비례 관계가 존재한다.

다음은 곡률이 30도 이하인 완만한 커브에서 PWM 변화를 나타낸 식이다. 방향은 왼쪽으로 완만하게 꺾이는 곡선이다.

$$curve = 1-(|centercircle-viewcenter|\times k)$$
$$PWMA = PWMA\times curve \quad \text{-(1)}$$
$$PWMB = PWMB$$

만약 곡률이 심한 곡선일수록 두 바퀴의 속도 차이는 커져야 하므로 1보다 작은 curve의 값은 작아져야 한다. 즉, 좌표값의 차이가 클수록 curve의 값은 작아져야 한다. 위와 같은 관계에서 수식 (1)을 도출하였다.

수식 (1)에서 centercircle(파란 점)은 두 차선의 중앙 좌표값이고 viewcenter(노란 점)는 시점(영상)의 중앙 좌표값이다. 실험 조건은 영상이 8~9frame/s, 초기 PWM을 30, k=0.003으로 설정한다. 프레임마다 라즈베리파이에서 영상처리하여 받아온 좌표값들을 대입하여 curve의 값을 구하고, 이를 회전하는 쪽의 모터의 PWM 값에 곱해준다. 실험 결과 curve의 값은 대부분 0.7~0.9 사이의 값을 가지게 되어 왼쪽으로 약하게 회전하면서 전진할 수 있다. 이 식의 원리는 차선의 중앙값과 영상의 중앙값을 비교하여, 곡률에 비례한 회전 각도를 조정하는 데 있다.

[그림 3]과 같이 두 점의 좌표값의 차이가 근소하게 나는 경우가 있다. 좌표값은 해상도에 따라 유동적으로 변하는데, 실험에 사용한 웹캠에서 그 차이가 10 이하로 측정되는 경우는 직진으로 처리하였다. 좁은 차선에서는 두 바퀴의 속도 차가 조금만 나도 차선을 벗어나기 때문이다.

다음은 곡률이 30도 이상의 다소 급한 커브에서 PWM 변화를 나타낸 식이다. 급한 곡선에서는 방향은 왼쪽으로 다소 급하게 꺾이는 곡선이다. 한쪽 Line이 영상에서 detection 되지 않을 수 있기 때문에 전 프레임의 좌표값 정보를 사용한다.

$$newcenter = x0 \pm \left| \frac{distance}{2} \right|$$
$$curve = 1-(|newcenter-viewcenter|\times k) \quad \text{-(2)}$$
$$PWMA = PWMA\times curve$$
$$PWMB = PWMB$$

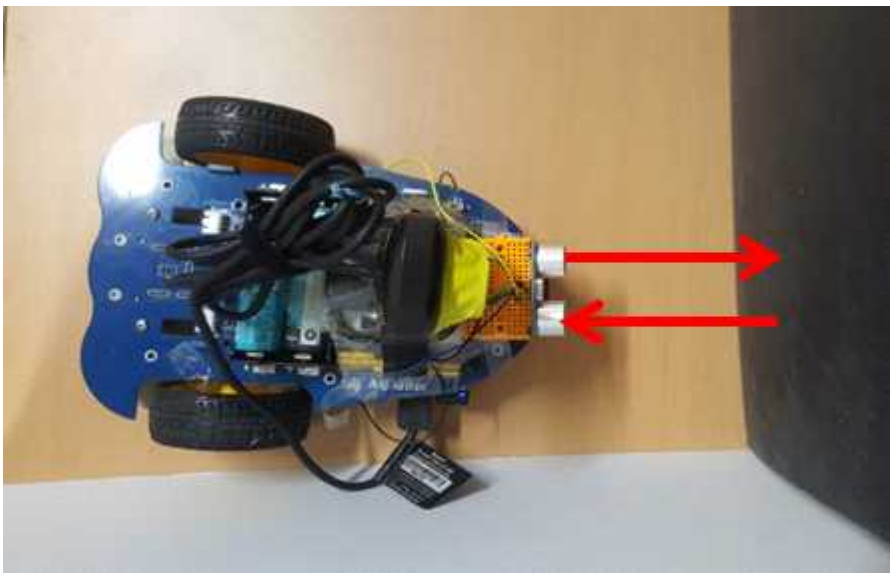
수식 (2)에서 x0는 검출된 한쪽 Line의 x좌표이고, distance는 전 프레임의 두 Line의 좌표값 차이이다. 검출된 차선이 왼쪽 차선이라면 distance의 절반을 더하고, 오른쪽 차선이라면 distance의 절반을 뺀다. 그리하여 새로운 차선의 중앙점 newcenter를 만든다. 그리고 수식 (1)과 같은 원리로 curve 값을 계산한다.

곡률이 높은 곡선에서는 초기 속도를 직선보다 더 낮은 속도로 설정한다. 그 이유는 직선만큼 빠른 속도로 주행하게 되면 라즈베리파이의 영상처리 속도가 따라가질 못해 프레임 지연 현상이 생기고, Real-time으로 처리하기 어렵기 때문이다. 실험 조건은 영상을 8~9frame/s, 초기 PWM을 20, k=0.0025로 설정한다. 프레임마다 라즈베리파이에서 영상 처리하여 받아온 좌표값들을 대입하여 curve의 값을

구하고, 이를 회전하는 쪽의 모터의 PWM 값에 곱해준다. 실험 결과 curve의 값은 대부분 0.7 이하의 값을 가지게 되어 급한 커브도 회전하면서 전진할 수 있다. 이 식의 원리는 현재의 영상이 바로 전 프레임과 큰 변화가 없다고 가정하고 distance 값을 받아와서 새로운 중앙점을 만들고 이를 이용하여 회전 각도를 계산하는 것이다.

3.3 장애물 인식

도로에서 사람이나 동물이 갑자기 튀어나오는 등 돌발 상황은 언제나 존재하기 때문에 이를 대비한 컨트롤러는 필요하다[7]. RC카 앞에 장애물이 접근하면 추가적인 동작을 할 수 있도록 한다. 실험에서는 장애물과의 거리가 20cm가 되면 자동차가 정지하도록 설계하였다.



[그림 4] RC카 – 장애물 인식

[그림 4]과 같이 RC카의 앞쪽에 초음파센서를 부착하였다. 실험에서 사용된 HC-SR04는 약 40KHz 정도의 주파수를 생성하며, 최대 4~5m 정도까지의 거리를 측정할 수 있다. 동작 원리는 송신부에서 일정한 시간의 간격을 둔 짧은, 초음파 펄스를 방사하고, 대상물에 부딪혀 돌아온 에코 신호를 수신부에서 받아, 이에 대한 시간차를 기반으로 거리를 산출한다[9]. 그림에서 빨간 화살표가 초음파 펄스의 경로이다.

$$Udistance = \frac{t}{2} \times (34000) \qquad \text{-(3)}$$

수식 (3)은 RC카 앞의 초음파센서와 장애물까지의 거리를 계산하는 식이다. 초음파는 공기 중에서 속도가 34000cm/s이고, t는 초음파가 송신되고 다시 수신될

때까지의 시간이다. t를 반으로 나누면 초음파가 센서에서 장애물까지 도달하는 데 걸리는 시간을 뜻한다. 거리 = 시간 X 속도라는 개념을 이용하여 거리를 계산한다. 측정한 거리 Udistance가 20 이하가 되면 정지하도록 설계하였다.

실험구간	초기속도(PWM)	프레임(frame/s)	결과
곡률 30도 이하	20	5	속도가 느려서 프레임 지연에 영향을 덜 받지만 매끄러운 주행 불가
		8	초기 속도가 너무 느려서 매끄러운 주행 불가 (일정 속도를 넘지 못하면 차체 전진 불가)
	30	5	프레임 지연이 길어서 실시간 처리에 어려움
		8	자율 주행하기에 최적의 조건
	40	5	프레임 지연이 길어서 실시간 처리에 매우 어려움
		8	1초의 8번 처리를 하기엔 너무 빠른 차량의 속도
곡률 30도 이상	20	5	속도가 느려서 프레임 지연에 영향을 덜 받지만 매끄러운 주행 불가
		8	자율 주행하기에 최적의 조건
	30	5	프레임 지연이 길어서 실시간 처리에 어려움
		8	곡률이 심한 경우에는 높은 속도에서의 방향 전환의 어려움
장애물 인식	40 미만	5	프레임 지연이 길어서 장애물을 인식하는 동안 차체가 전진
		8	장애물 인식하기에 최적의 조건

[표 4] 실험 구간 별 초기 속도, 프레임 조건에 대한 실험 결과

[표 4]는 초기 속도, 초당 프레임 수에 따라 나타낸 실험 결과이다. 곡률 30도 이하의 완만한 곡선에서는 초기 PWM 30, 8frame/s가 가장 적절하고, 곡률 30도 이상의 급한 곡선에서는 초기 PWM 20, 8frame/s가 가장 적절하다. 장애물 인식

구간에서는 초기 PWM은 40 미만의 속도에서는 모두 8frame/s가 가장 적절한 조건이다.

4. 문제점 및 해결방안

4.1 통신 및 연산 속도 한계

라즈베리파이의 성능으로 실시간 영상 전송 및 처리는 어려웠다. 처음 완성된 영상처리 알고리즘은 초당 1frame 도 넘지 못하는 성능을 보여주었다. 그래서 알고리즘에서 불필요한 연산을 지워내고, ROI를 설정하여 최소한의 영역만 영상처리하여 속도를 높였다. 통신 속도를 증가하기 위해 Giga WIFI 망에서 실험을 진행하였다. 그래도 0.1초정도의 delay를 가지기 때문에 동작이 끝나고 0.1초 정도의 time sleep을 추가하였다. 위와 같은 방법으로 동작이 완료되는 동안 연산을 하고 다음 프레임의 정보를 받을 수 있도록 설계하였다.

4.2 Line Detection 한계

실험에서 사용한 주행 코스는 모두 수작업으로 만들었다. 실험 시간과 장소에 따라 빛의 잡음이 차선 검출에 다소 영향을 주었다. curve 값이 한 프레임에서라도 급격하게 튀면 차선 밖으로 나가는 오작동을 일으켰다. 오작동을 최소한으로 죽이기 위해 카메라의 필터를 흑백으로 하여 빛의 잡음을 최소화하여 효과적인 차선 검출을 도출하고, 광각렌즈를 사용하여 카메라가 받는 영상의 화각을 넓혀서 조금 더 넓은 폭의 차선도 검출할 수 있도록 하였다. 먼 차선이 검출되지 않거나 curve 값이 급격히 변하는 경우 현재 연산을 정지하고 전 프레임 차선의 정보를 처리하도록 설계하였다.

5. 결론 및 향후 연구 방향

본 논문에서는 영상처리가 가능한 RC카 차량의 모터 제어 방법에 대하여 알아보았다. 곡률과 각 바퀴의 속도가 어떤 관계를 가지고 변화하는지를 알아보기 위해 우리가 가진 line detection 알고리즘에 대입하고 반복적인 실험을 통해 수식을 유도하였다.

제안된 방법은 라즈베리파이와 WIFI 환경에서 실험하여 연산의 속도가 빠르지 않

아 완전한 real-time 처리는 어렵다. 이는 성능이 더 좋은 임베디드 시스템으로 설계한다면 해결할 수 있다. real-time 처리가 가능하다면 초기 속도를 더 높여서 주행하는 것도 가능하다. 현재 같은 WIFI 망 내에서 원격 자율 주행이 가능한 상태기 때문에 통신망에 구애받지 않는 통신 환경에 대한 추가적인 보완과 연구가 필요하다.

<참고문헌>

[1] 이병윤. "국내외 자율주행자동차 기술개발 동향과 전망." 한국통신학회지(정보와 통신), 33.4 10-16 / 2016.3

[2] 안수진, 한민홍. "자율주행차량을 위한 차선인식에 관한 연구." 한국정보기술학회 논문지, 5.1 136-142 / 2007.3

[3] 김운철, 이상원, 최용윤, 김광림, 이영일. "영상처리를 이용한 모형자동차의 자율주행." 제어로봇시스템학회 국내학술대회 논문집 660-663 / (2011.5)

[4] 위키백과, https://ko.wikipedia.org/wiki/%EB%9D%BC%EC%A6%88%EB%B2%A0%EB%A6%AC_%ED%8C%8C%EC%9D%B4, 2018-11-19

[5] Waveshare wiki, https://www.waveshare.com/wiki/AlphaBot, 2018-11-19

[6] tistory blog, http://klero.tistory.com/entry/Xming%EC%9D%84-PuTTY%EC%99%80-%EC%97%B0%EA%B2%B0%ED%95%B4%EC%84%9C-%EA%B0%84%EB%8B%A8%ED%95%98%EA%B2%8C-%EC%82%AC%EC%9A%A9%ED%95%98%EB%8A%94-%EB%B0%A9%EB%B2%95, 2018-11-28

[7] 네이버 블로그, http://blog.naver.com/PostView.nhn?blogId=lagrange0115&logNo=220625887280&parentCategoryNo=&categoryNo=&viewDate=&isShowPopularPosts=false&from=postView, 2018-11-19

[8] 김현지, 정의철. "자율주행환경 차량탑승자 직관에 의해 돌발 상황 대처가 가능한 Control UI 디자인 컨셉 제안." 한국자동차공학회 추계학술대회 및 전시회 1422-1429 / 2017.11

[9] kocoafab, https://kocoafab.cc/tutorial/view/357, 2018-11-18