學 士 學 位 論 文

SoC드론을 위한 영상처리 및 자율비행

충남대학교

공과대학 컴퓨터공학과

이 동 환

윤 신

안 희 준

지도교수 남 병 규

2020 年 2 月

SoC드론을 위한 영상처리 및 자율비행

지도교수 남 병 규

이 논문을 공학사학위
청구논문으로 제출함

2019 年 11 月

충 남 대 학 교

공과대학 컴퓨터공학과

201402388 이 동 환
201402384 윤 신
201402372 안 희 준

목 차

그림목차

표목차

Ⅰ. 서 론

1.1 연구의 배경과 목적

기술의 발전과 함께 드론 연구에 대한 관심 및 필요성이 증가하고 있다. 드론은 초기에 공격용 무기를 통해 적을 공격하는 용도와 접근하기 힘든 곳 정찰 등의 군사적 용도로 사용되었다. 최근에는 군사적인 용도를 넘어서 물건을 운송하거나 농업, 영화 촬영 등 다양한 용도로 사용되고 있다. 취미용으로도 개발되어 사업자뿐만 아니라 일반 사람들도 12kg 이하인 드론을 제어할 수 있다.

드론이 촬영한 영상을 기존 인식 방법보다 빠르게 할 수 있는 연구 또한 오랫동안 진행되어왔다. 영상처리는 20세기 중반까지 아날로그에 의한 방법을 사용하였고, 컴퓨터 처리 속도가 향상됨에 따라 정확하고, 구현하기 쉬운 디지털 처리 기법으로 대체되었다. 소프트웨어로 영상 처리를 구현 시 속도가 느려진다는 문제점을 보완하기 위해 하드웨어 언어인 Verilog를 사용하여 영상처리를 구현하는 방법을 연구 및 구현해보기로 하였다. 이것 뿐만 아니라 영상처리를 통해 얻은 데이터를 바탕으로, 제어 장치를 사용할 수 없는 상황에서 드론이 사물을 인식하여 사물의 이동 경로를 추적하도록 하는 방법을 연구 및 구현해보기로 하였다. 그리고 드론이 비행 시 고도를 유지하며 안정적으로 비행할 수 있는 방법도 연구하고 구현해보기로 하였다.

1.2 연구의 방법

카메라로 보여주는 영상을 활용하여 물체를 인식하기 위해서는 영상을 흑백으로 처리하거나 배경과 물체를 분리하는 과정 등 여러 과정이 필요하다. 우리가 사용하려는 영상 속 물체는 인식하기 쉬운 하얀 바탕에 흑백으로 되어있는 물체이기 때문에 그레이스케일(Grayscale) 영상으로 변환시켜주는 과정만 수행할 것이다. 그레이스케일(Grayscale)로 변환된 영상으로부터 물체를 인식하기 위해 영상에서 특징점을 추출한다. 특징점은 물체의 형태나 크기 및 위치가 변해도 물체를 쉽게 식별할 수 있게 도와주는 점이다. 특징점을 추출하는 여러 알고리즘 중 우리는 FAST알고리즘을 사용하였다. FAST알고리즘 중에서도 연산속도가 빠름과 동시에 신뢰성이 강한 FAST-9 알고리즘을 사용하였다. FAST-9알고리즘은 총 3개의 단계로 나누어져 있다. 먼저 Feature Detection은 사용자가 정의한 임계값을 이용하여 특징점 후보를 판별한다. Feature Score는 자동으로 픽셀의 최저 임계값을 계산하여 특징점 후보들의 점수로 사용한다. 그리고 Non-Maximal Suppression은 인접한 특징점들의 점수를 비교하여 가장 높은 점수의 특징점만을 유효 특징점으로 선택하고 어두운 부분과 밝은 부분을 따로 비교하여 최종 특징점을 선택한다. 위 과정을 통하여 물체를 인식할 수 있게 된다. 영상 처리 과정

을 통하여 소프트웨어로 구현할 수 있다. 하지만 소프트웨어언어로 구현 할 시 처리 속도가 느리다는 단점 때문에 드론이 실시간으로 영상을 처리하는데 있어서 어려움이 있을 것이라 생각하였다. 그래서 하드웨어인 FPGA(Field Programmable Gate Array)위에서 영상처리를 하도록 하였고, FPGA(Field Programmable Gate Array)에 올라갈 영상처리 모듈들은 Verilog언어를 통하여 구현하였다. 드론을 제어하기 위해서 PID Control을 드론에 접목시킬 것이다. PID Control은 단순히 비례(Proportional), 적분(Integral), 미분(Differential)를 의미한다. 제어하고자 하는 대상에 의해 정해지는 입력 값을 측정하여 이를 목표로 하는 설정 값과 비교하여 오차(error)를 계산한다. 이 때 오차(error) 값을 이용하여 제어에 필요한 제어 값을 계산하고, 이 제어 값을 다시 피드백(feedback) 구조로 대상의 입력으로 받는 구조를 가지고 있다. PID Control을 이용하여 드론이 고도를 유지하며 안전하게 비행할 수 있도록 할 것이다.

Ⅱ. 사용된 알고리즘

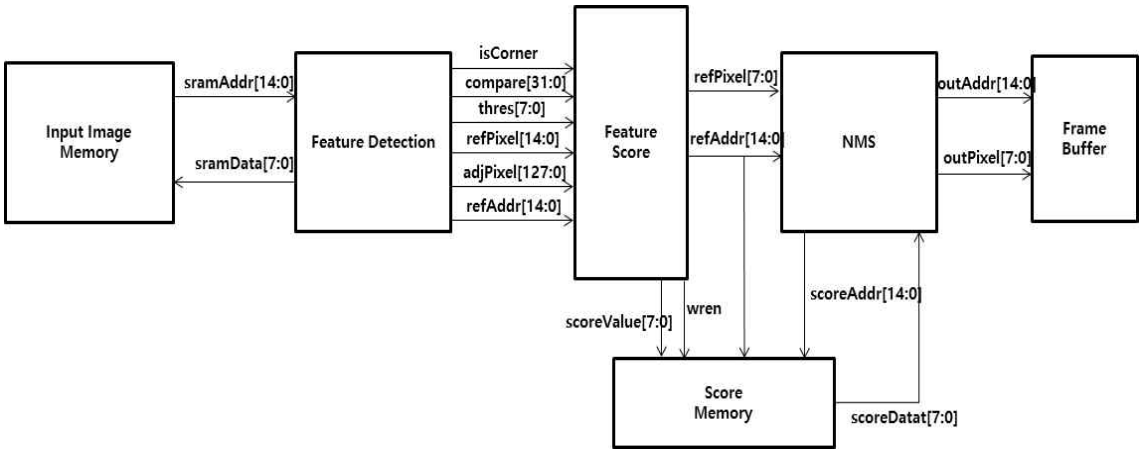
2.1 그레이스케일(Grayscale)로의 변환

카메라로부터 비디오 디코더(Video Decoder)는 720x480크기의 RGB565영상을 초당 30프레임의 속도로 전달받는다. 본격적인 영상처리/인식을 하기 전에 FPGA에서 우선 RGB565영상을 그레이스케일 영상으로 바꾸는 과정과 180x120크기로 영상 크기를 조절하는 과정이 필요하다. 여기서 RGB565는 Red, Green, Blue가 각각 5비트, 6비트, 5비트씩 차지하는 16비트로 한 컬러픽셀을 표현하는 방식이고, Green만 6비트인 이유는 눈에 초록색이 민감하기 때문이다.

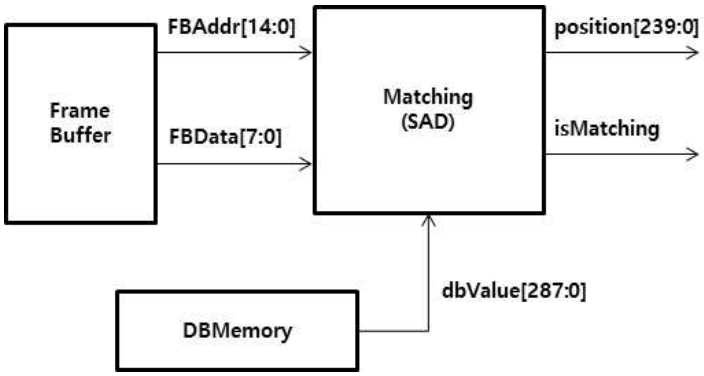
$$Grayscale = (R + G + B)/3 \qquad (식1)$$

이 RGB565를 (식1)을 이용하면 그레이스케일로 변환가능하다. 영상의 크기는 메모리의 용량을 고려하여 180x120으로 조절하였다.

2.2 FAST-9을 이용한 특징점 추출



<그림1> FAST-9 Module Block Diagram



<그림2> Matching Module Block Diagram

<그림1>에서의 FAST-9에서는 3개 Stage와 3개의 SRAM으로 구성한다. 3개의 Stage에서는 Feature Detection단계, Feature Score단계 그리고 Non-Maximal Suppresion단계가 있다.

3개의 Stage에 대해서 간단히 설명하자면 Feature Detection단계에서는 Input Image Memory로 통해서 받아들인 이미지를 가지고 사용자가 정의한 임계값을 이용하여 특징점 후보들을 판별한다. Feature Score단계에서는 Feature Detection단계를 통해서 알아낸 특징점 후보들의 score(점수)를 계산한다. 계산하는 방식은 아래 (식2)를 참고한다.

$$V = \max\left(\sum_{x \in S_{bright}} |I_{p \to x} - I_p| - t, \sum_{x \in S_{dark}} |I_p - I_{p \to x}| - t \right)$$

I_p : 기존 pixel값
t : 임계값 (threshold)
$I_{p \to x}$: 주변 pixel 값

(식2)

Non-Maximal Suppresion 단계에서는 Feature Score에서 계산한 score 값을 가지고 인접한 특징점들의 점수를 비교하여 가장 높은 점수의 특징점만을 유효 특징점으로 선택하고 어두운 부분과 밝은 부분을 따로 비교하여 최종 특징점을 선택한다. SRAM은 Input Image Memory, Score Memory 그리고 FrameBuffer Memory가 존재한다. Input Image Memory는 FPGA에 연결되어 있는 카메라를 통해서 영상을 받아들이고 받아들인 이미지를 저장하는 역할을 한다. Score Memory는 Feature Score단계에서 계산된 score값들을 저장하는 역할을 하고 있으며 Frame Buffer에서는 Non-Maximal Suppression 단계를 통해 계산한 corner결과를 반영한 픽셀 값들을 저장하는 역할을 하고 있다. 3개의 Stage와 3개의 SRAM을 가진 FAST-9 알고리즘을 통해서 카메라로부터 받아온 영상을 인식하여 특징점들을 추출하게 된다.

<그림2>는 <그림1>의 FAST-9알고리즘을 통해서 추출한 특징점 데이터 값들(Frame Buffer에 저장된 값)을 기존에 추출한 특징점 데이터 값들(DBMemory에 저장된 값)과 비교하여 matching이 되었는지 유무를 판단하는 과정에 대한 그림이다. 이를 위해 SAD알고리즘을 사용한다.

2.3 PID Controller

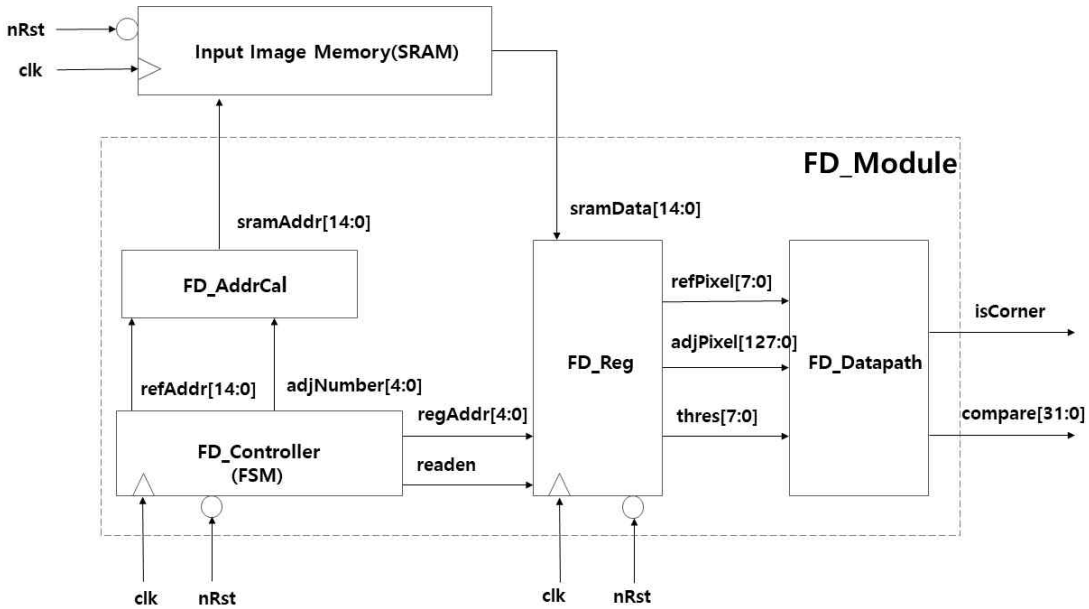
PID 제어는 기본적으로 피드백 제어기 형태이며, 제어하고자 하는 대상의 출력값과 참조값(또는 설정값)을 비교하고, 이 오차값을 이용하여 제어에 필요한 제어값 수치를 계산하는 구조이다. 표준적인 PID 제어기는 P(Proportional, 비례항), I(Integral, 적분항), D(Derivative, 미분항) 이렇게 세 개의 항을 더하여 제어값을 계산하도록 구성되어있다. 비례항은 현재 상태에서 오차값 크기에 비례한 제어작용을 한다. 적분항은 제어값이 참조값에 근접했을 때 발생하는 오차를 줄이는 작용을 한다. 미분항은 출력값의 급격한 변화가 있을 때 제동을 걸어 제어 안정성을 향상시킨다.

PID 제어기는 위와 같은 보통 표준식의 형태로 사용하지만, 경우에 따라서는 약간 변형된 형태로 사용하는 경우도 많다. 예를 들어, 비례항만을 가지거나, 혹은 비례-적분, 비례-미분 항만을 가진 제어기의 형태로 단순화하여 사용하기도 하는데, 이 때는 각각 P, PI, PD 제어기라 불린다.

Ⅲ. 설계한 영상처리 Module

3.1 Feature Detection Module

3.1.1 FD Block Diagram



<그림3> Feature_Detection Module Block Diagram

이미지 안에서 물체의 경계선 근처에 위치한 픽셀들은 밝기 차이가 크게 난다. 이 점을 이용하면 기준이 되는 픽셀과 인접 픽셀의 밝기 차이를 조사해서 비교하는 것만으로도 어떤 픽셀이 물체의 경계선이나 모서리에 위치할지 예상할 수 있다. 이렇게 선정하는 픽셀을 후보 특징점이라고 부르며, 이미지에서 후보 특징점들을 찾아내는 과정을 FD(Feature Detection)라고 한다.

Feature Detection의 간단한 알고리즘은 먼저 주변 16개의 픽셀 값과 기준 픽셀들을 Input Image Memory(SRAM)에서 read를 하고 이 값들을 Register file에 저장한다. 그리고 사용자 threshold값과 기준 픽셀값들을 read한 후 두 값의 합과 차(sum&differecne)를 계산한다. 윗 단계에서 계산한 합과 차를 주변 16개의 픽셀들과 비교한다. 이 때, 9개의 연속적인 인접 픽셀들의 값이 계산된 합보다 클 경우 기준 픽셀을 S_{bright}로 설정 및 계산된 차보다 작을 경우 기준 픽셀을 S_{dark}로 설정한다.

이러한 기준픽셀이 S_{bright} 또는 S_{dark}일 경우 Feature Detection으로 계산된 특징점으로 판별한다.

3.1.2 FD State Diagram

현재 상태	입력	다음 상태	출력
INIT	X	S0	adjNumber=5'bx, regAddr = 5'bx, readen=1'b0;
S0	X	S1	adjNumber=5'd0, regAddr = 5'bx, readen=1'b0;
S1	X	S2	adjNumber=5'd1, regAddr = 5'bx, readen=1'b0;
S2	X	S3	adjNumber=5'd2, regAddr = 5'd0, readen=1'b0;
S3	X	S4	adjNumber=5'd3, regAddr = 5'd1 readen=1'b0;
S4	X	S5	adjNumber=5'd4, regAddr = 5'd2, readen=1'b0;
S5	X	S6	adjNumber=5'd5, regAddr = 5'd3, readen=1'b0;
S6	X	S7	adjNumber=5'd6, regAddr = 5'd4, readen=1'b0
S7	X	S8	adjNumber=5'd7, regAddr = 5'd5, readen=1'b0
S8	X	S9	adjNumber=5'd8, regAddr = 5'd6, readen=1'b0
S9	X	S10	adjNumber=5'd9, regAddr = 5'd7; readen=1'b0;
S10	X	S11	adjNumber=5'd10, regAddr = 5'd8; readen=1'b0;
S11	X	S12	adjNumber=5'd11, regAddr = 5'd9, readen=1'b0;
S12	X	S13	adjNumber=5'd12, regAddr = 5'd10, readen=1'b0;
S13	X	S14	adjNumber=5'd13, regAddr = 5'd11, readen=1'b0;
S14	X	S15	adjNumber=5'd14, regAddr = 5'd12, readen=1'b0;

S15	X	S16	adjNumber=5'd15, regAddr = 5'd13, readen=1'b0;
S16	X	S17	adjNumber=5'd16, regAddr = 5'd14, readen=1'b0;
S17	X	S18	adjNumber=5'bx, regAddr = 5'd15, readen=1'b0;
S18	X	S19	adjNumber=5'bx, regAddr = 5'd16, readen=1'b0;
S19	X	INIT	adjNumber=5'bx, regAddr = 5'bx, readen=1'b1;

<표1> FD_FSM의 State Diagram

3.1.3 FD_ConXtroller submodule

입력신호	설명
clk	메인 clock
nRst	Negative Reset
출력신호	설명
refAddr[14:0]	기준 픽셀의 주소값 (180*120=21600)
adjNumber[4:0]	인접한 픽셀들의 번호 (1~16)
regAddr[4:0]	Register file의 주소값
readen	Register file read enable
기능	설명
FD_Reg module과 FD_AddrCal module을 제어하기 위한 신호를 생성하는 모듈이다.	

<표2> FD_Controller의 신호

INIT state에서 신호의 초기 값을 설정해주고, 다음 state에서는 각 인접한 픽셀들의 신호를 생성한다. regAddr[14:0]는 FD_Reg에 바로 들어가지만 adjNumber[4:0]는 FD_AddrCal을 거쳐서 FD_Reg module에 도달한다.

두 신호가 도착하는 시간 차이를 고려하여 regAddr와 adjNumber는 2cycle만큼 차이가 발생하도록 조절한다.

3.1.4 FD_AddrCal submodule

입력신호	설명
refAddr[14:0]	기준 픽셀의 주소값
adjNumber[4:0]	인접한 픽셀들의 번호
출력신호	설명
sramAddr[14:0]	기준 픽셀의 주소값을 이용하여 계산한 인접 픽셀의 주소값
기능	설명
기준 픽셀의 주소를 바탕으로 기준픽셀에 인접한 16개의 픽셀의 주소값을 계산 하는 모듈이다.	

<표3> FD_AddrCal의 신호

인접한 픽셀들의 번호(adjNumber[4:0])들에 대해 refAddr를 기준으로 인접 픽셀의 주소값을 계산하여 sramAddr[14:0]에 저장한다.

3.1.5 FD_Reg submodule

입력신호	설명
clk	메인 clock
nRst	Negative Reset
regAddr[4:0]	Register file의 주소값
readen	FD_Controller에서 전달받은 값들
sramData[14:0]	Input Image Memory로부터 받아온 픽셀 값 (0~255)
출력신호	설명
refPixel[7:0]	기준 픽셀 값
adjPixel[127:0]	주변 16개 픽셀들의 픽셀 값 (0~255) *1개 픽셀은 8비트필요
thres[7:0]	사용자 임계값 (0~255)
기능	설명
기준 픽셀과 기준 픽셀 주변 16개의 픽셀 값을 저장하는 모듈이다.	

<표4> FD_Reg의 신호

FD_AddrCal의 출력 신호인 sramAddr[14:0]값을 바탕으로 Input Image Memory에서 픽셀 값을 가져온다. 이 값을 sramData[14:0]에 저장한다. 이 sramData[14:0]를 이용하여 기준 픽셀(refPixel)값과 주변 픽셀(adjPixel)값을 저장한다.

3.1.6 FD_Datapath submodule

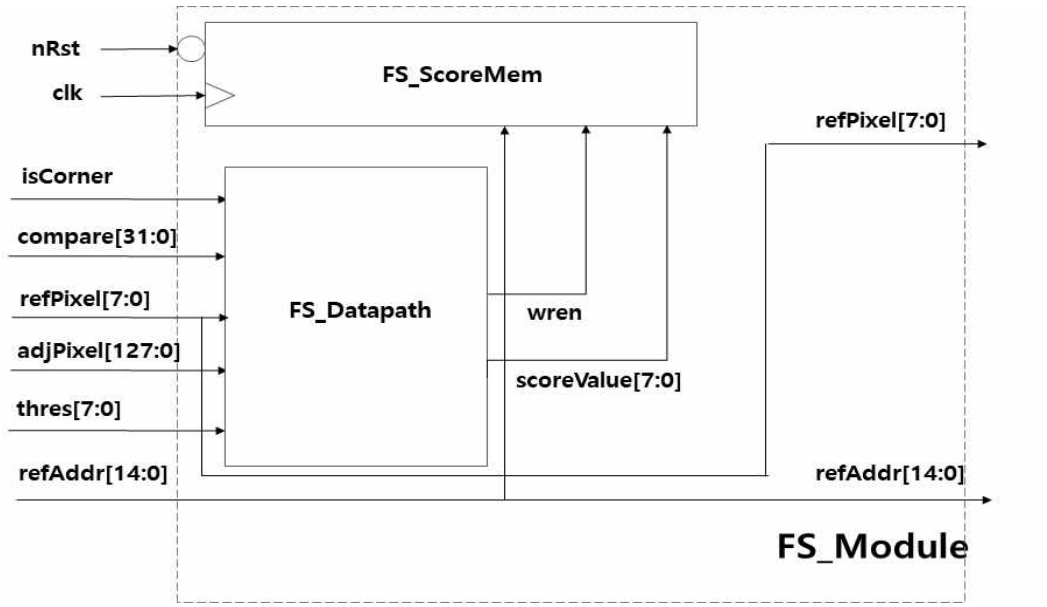
입력신호	설명
refPixel[7:0]	기준 픽셀 값
adjPixel[127:0]	주변 16개 픽셀들의 픽셀 값
thres[7:0]	사용자 임계값
출력신호	설명
isCorner	기준 픽셀이 corner(코너)인지를 판별하는 결과값
compare[31:0]	16개의 점에 대해서 각각 Dark, Brighter, Similar 값 저장
기능	설명
기준 픽셀과 주변 16개의 픽셀 값을 비교하여 corner을 판별하는 모듈이다.	

<표5> FD_Datapath의 신호

adjPixel[127:0]을 8bit씩 나눈 뒤 adjPixel이 기준점(refPixel[7:0]) - 임계값(thres[7:0])보다 작으면 Dark, adjPixel이 기준점(refPixel[7:0]) + 임계값(thres[7:0])보다 크면 Brighter, 그 외의 경우 Similar 값을 compare[31:0]에 2bit씩 저장한다. compare[31:0] 중 9개가 연속으로 Dark 또는 Brighter이면 corner가 된다.

3.2 FS(Feature Score) Module

3.2.1 FS Block Diagram



<그림4> Feature_Score Module Block Diagram

이미지에서 후보 특징점을 뽑았을 때, 몇몇 후보 특징점은 다른 후보 특징점에 비해 물체의 경계를 뚜렷이 나타낼 수 있다. 어떤 후보 특징점이 물체의 경계를 뚜렷하게 나타내는지 알 수 있도록 점수를 계산하고 매기는 과정이 필요하다. 이런 과정을 FS(Feature Score)라고 한다.

Feature Score의 간단한 알고리즘은 먼저 사용자가 지정한 threshold값과 픽셀값의 최대값인 255의 평균값을 구하여 저장한다. 이 구한 평균값과 픽셀값들을 read한 후 두 값의 합과 차를 구한다. 이러한 합과 차를 주변 16개의 픽셀값들과 비교한다. 이 때, 9개의 연속적인 인접 픽셀들의 값이 계산된 합보다 클 경우 기준 픽셀을 S_{bright}로 설정 및 계산된 차보다 작을 경우 기준 픽셀을 S_{dark}로 설정한다. 만약에 특징점이 아니라면 다시 픽셀값의 최대값인 255를 평균값으로 대체하고 특징점이라면 사용자의 threshold값을 평균값으로 대체한다. 그리고 전 단계의 두 값을 비교하여 같을 경우 score로 정의하지만 다를 경우에는 다시 첫 번째 단계로 올라가 다시 시작한다.

3.2.2 FS_Datapath submodule

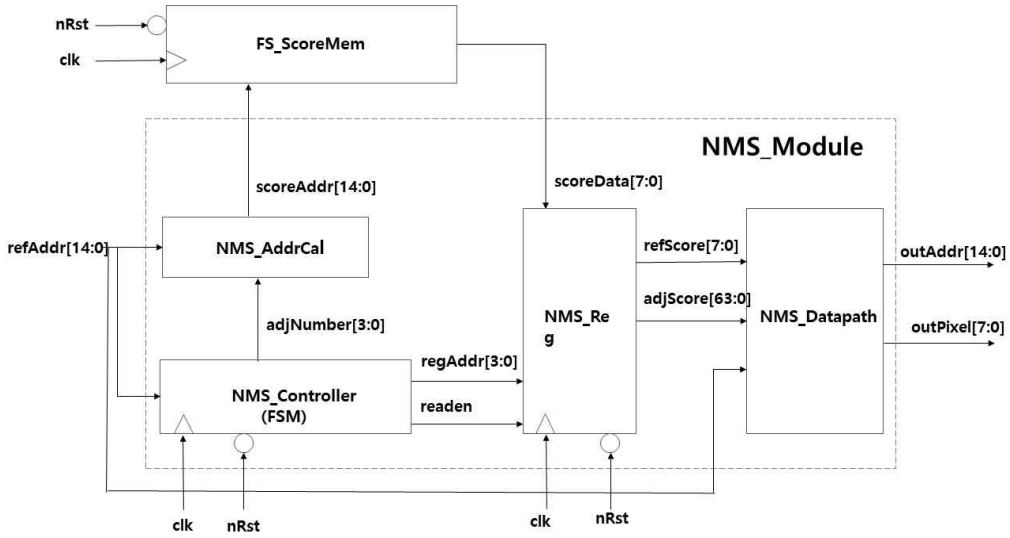
입력신호	설명
isCorner	기준 픽셀이 corner(코너)인지를 판별하는 결과값
compare[31:0]	16개의 점에 대해서 각각 Dark, Bright, Similar 값
refPixel[7:0]	기준 픽셀 값
adjPixel[127:0]	주변 16개 픽셀들의 픽셀 값
thres[7:0]	사용자 임계값
출력신호	설명
wren	write 신호
scoreValue[7:0]	corner(코너) 값
기능 설명	
후보 특징점에 점수를 매기는 역할을 한다	

<표6> FS_Datapath의 신호

입력 받은 값을 이용하여 기준 픽셀의 임계 점수를 설정하는 모듈이다. 점수를 설정하는 과정은 위에 설명되어 있는 Feature Score 알고리즘과 동일하다.

3.3 NMS(Non-Maximal Suppression) Module

3.3.1 NMS Block Diagram



<그림5> NMS Module Block Diagram

이미지의 변화가 심한 부분에서 많은 수의 특징점 후보들이 추출될 가능성이 매우 높다. 하지만 많은 수의 특징점 후보들에 대해 모두 매칭 과정을 수행하는 것보다 특정 부분을 대표하는 특징점 하나에 대해서만 매칭 과정을 수행하는 것이 효율적이다. 많은 수의 특징점 후보들 중 하나의 특징점 대표를 뽑는 과정을 NMS(Non-Maximal Suppression)라고 한다.

Non-Maximal Suppression의 간단한 알고리즘은 주변 8개의 인접 픽셀들의 FS_ScoreMem(SRAM)주소를 계산하고 FS_ScoreMem에서 특징점 후보들의 score을 read한 후 기준 픽셀들과 score들을 비교한다. 이 때 기준 픽셀에 인접한 특징점 후보들의 score가 기준 픽셀의 score보다 클 경우에는 기준 픽셀을 특징점 후보에서 제외를 하고 작을 경우에는 기준 픽셀을 특징점 후보로 저장한다.

3.3.2 NMS State Diagram

현재 상태	입력	다음 상태	조건	출력
INIT	X	INIT	if(refAddr<15'd906)	readen=1'b0;
		S0	else	
S0	X	S1		adjNumber=4'd0, regAddr=4'bx, readen=1'b0

S1	X	S2		adjNumber=4'd1, regAddr=4'bx, readen=1'b0
S2	X	S3		adjNumber=4'd2, regAddr=4'd0, readen=1'b0
S3	X	S4		adjNumber=4'd3, regAddr=4'd1, readen=1'b0
S4	X	S5		adjNumber=4'd4, regAddr=4'd2, readen=1'b0
S5	X	S6		adjNumber=4'd5, regAddr=4'd3, readen=1'b0
S6	X	S7		adjNumber=4'd6, regAddr=4'd4,
S7	X	S8		adjNumber=4'd7, regAddr=4'd5, readen=1'b0
S8	X	S9		adjNumber=4'd8, regAddr=4'd6, readen=1'b0
S9	X	S10		adjNumber=4'bx, regAddr=4'd7, readen=1'b0
S10	X	S11		adjNumber=4'bx, regAddr=4'd8, readen=1'b0
S11	X	S12		adjNumber=4'bx, regAddr=4'bx, readen=1'b0
S12	X	S13		adjNumber=4'bx, regAddr=4'bx, readen=1'b0
S13	X	S14		adjNumber=4'bx, regAddr=4'bx, readen=1'b0
S14	X	S15		adjNumber=4'bx, regAddr=4'bx, readen=1'b0
S15	X	S16		adjNumber=4'bx, regAddr=4'bx, readen=1'b0
S16	X	S17		adjNumber=4'bx, regAddr=4'bx, readen=1'b0
S17	X	S18		adjNumber=4'bx, regAddr=4'bx, readen=1'b0
S18	X	S19		adjNumber=4'bx, regAddr=4'bx, readen=1'b0
S19	X	INIT		adjNumber=4'd15, regAddr=4'bx, readen=1'b1

<표7> NMS_FSM의 State Diagram

3.3.3 NMS_Controller submodule

입력신호	설명
clock	메인 clock
nReset	Negative Reset
refAddr[14:0]	기준 픽셀의 주소값
출력신호	설명
adjNumber[3:0]	인접한 8개의 점의 번호
regAddr[3:0]	Register file의 주소값
readen	Register file read enable
기능 설명	
NMS_Reg, NMS_AddrCal을 제어하기 위한 FSM이다.	

<표8> NMS_Controller의 신호

NMS_Reg, NMS_AddrCal을 제어하기 위해 state machine의 형태를 취하고 있다. INIT state에서 신호의 초기 값을 설정해주고, 다음 state에서는 각 인접한 픽셀들의 신호를 생성한다. regAddr[3:0]는 NMS_Reg에 바로 들어가지만 adjNumber[3:0]는 NMS_AddrCal을 거쳐서 NMS_Reg module에 도달한다.

두 신호가 도착하는 시간 차이를 고려하여 regAddr와 adjNumber는 2cycle만큼 차이가 발생하도록 조절한다.

3.3.4 NMS_AddrCal submodule

입력신호	설명
refAddr[14:0]	기준 픽셀의 주소값
adjNumber[3:0]	인접한 8개의 점의 번호
출력신호	설명
scoreAddr[14:0]	FS_ScoreMem의 input 주소
기능 설명	
기준 픽셀 주변 8개 점의 주소값을 계산한다	

<표9> NMS_AddrCal의 신호

기준 픽셀의 주소 refAddr를 바탕으로 이미지가 180 x 120 사이즈라는 점을 이용해서 기준 픽셀 주변 인접 픽셀 8개의 주소값을 계산한다.

3.3.5 NMS_Reg submodule

입력신호	설명
clock	메인 clock
nReset	Negative Reset
readen	Register file read enable
regAddr[3:0]	Register file의 주소값
scoreData[7:0]	FS_ScoreMem으로부터 읽어온 값
출력신호	설명
refScore[7:0]	기준 픽셀의 score값
adjScore[63:0]	인접한 8개 픽셀들의 score 값
기능 설명	
기준 픽셀과 주변 픽셀 8개의 score을 저장한다	

<표10> NMS_Reg의 신호

NMS_AddrCal의 출력 신호인 scoreAddr[14:0]값을 FS_ScoreMem에서 픽셀 값을 가져온다. 이 값을 scoreData[7:0]에 저장한다. 이 scoreData[7:0]를 이용하여 기준 픽셀(refPixel)값과 주변 픽셀(adjPixel)값의 score들을 저장한다.

3.3.6 NMS_Datapath submodule

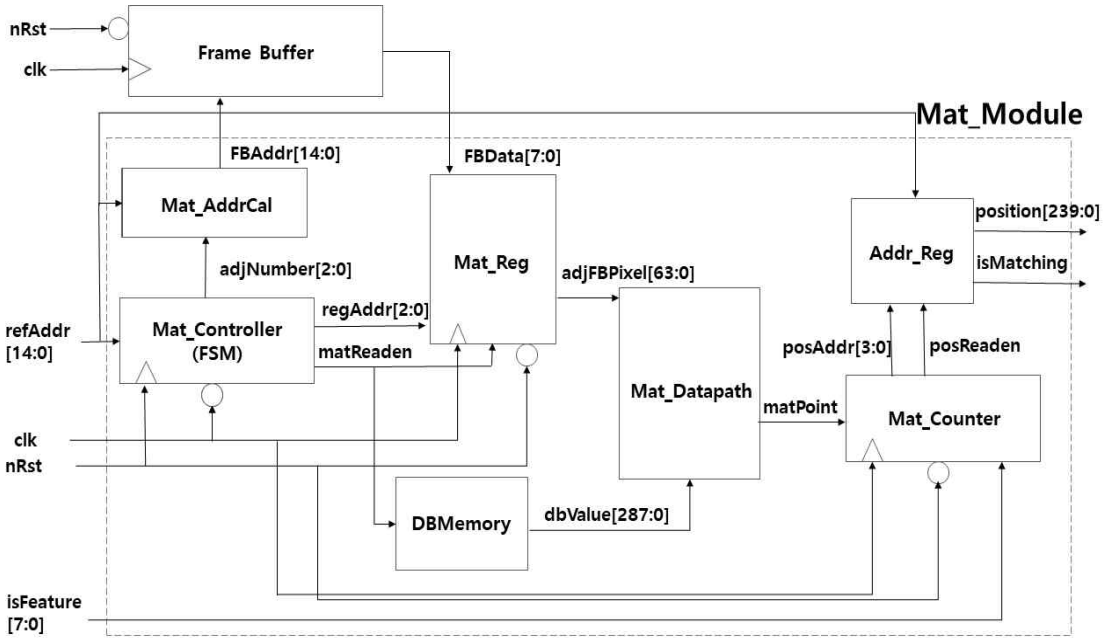
입력신호	설명
refScore[7:0]	기준 픽셀의 score값
adjScore[63:0]	인접한 8개 픽셀들의 score 값
refAddr[14:0]	기준 픽셀의 주소값
출력신호	설명
outAddr[14:0]	최종 corner(코너) 주소값
outPixel[7:0]	최종 corner(코너) 값
기능 설명	
기준 픽셀과 근처 픽셀들 중 최종 특징점을 판별한다	

<표11> NMS_Datapath의 신호

기준 픽셀과 기준 픽셀 주변 8개의 score가 담긴 refScore와 adjScore의 수치를 전부 비교하여 그 중 가장 높은 score를 가진 점을 최종 특징점으로 판별한다

3.4 Matching Module

3.4.1 Matching Block Diagram



<그림6> Matching Module Block Diagram

Matching은 대상 이미지와 DBMemory에 있는 주변 픽셀 값을 이용하는 것이다. 카메라로 보여주는 이미지의 특징점 주변 픽셀들의 평균값과 DBMemory안에 있는 값의 차이를 이용하여 Matching이 되었는지 여부를 판별하게 된다.

이 때 Matching module에서 사용하는 알고리즘은 SAD알고리즘이다. SAD알고리즘은 먼저 특징점 주변 3x3픽셀들을 read한다. 이 때 특징점을 제외한 3x3픽셀들의 평균값을 구하고 DBMemory에 미리 계산된 3x3픽셀들의 평균값을 read한다. 그리고 대상 특징점의 평균값과 DBMemory에 있는 모든 특징점 값들을 뺀 결과 값을 사용자 threshold값 이하일 경우 해당 특징점은 유효한 특징점으로 판단하고 이 특징점을 Register file에 저장한다. 이 과정들을 계속 반복하게 진행하여 사용자가 임의로 지정한 개수 이상일 경우 해당 이미지는 matching 되었다고 판단한다.

3.4.2 Matching State Diagram

현재 상태	입력	다음 상태	조건	출력
INIT	X	S0	if(refAddr)	adjNumber=3'bx, regAddr=3'bx, matReaden=1'b0
		INIT	else	
S0	X	S1		adjNumber=3'd0, regAddr=3'bx, matReaden=1'b0
S1	X	S2		adjNumber=3'd1, regAddr=3'bx, matReaden=1'b0
S2	X	S3		adjNumber=3'd2, regAddr=3'd0, matReaden=1'b0
S3	X	S4		adjNumber=3'd3, regAddr=3'd1, matReaden=1'b0
S4	X	S5		adjNumber=3'd4, regAddr=3'd2, matReaden=1'b0
S5	X	S6		adjNumber=3'd5, regAddr=3'd3, matReaden=1'b0
S6	X	S7		adjNumber=3'd6, regAddr=3'd4, matReaden=1'b0
S7	X	S8		adjNumber=3'd7, regAddr=3'd5, matReaden=1'b0
S8	X	S9		adjNumber=3'bx, regAddr=3'd6, matReaden=1'b0
S9	X	S10		adjNumber=3'bx, regAddr=3'd7, matReaden=1'b0
S10	X	S11		adjNumber=3'bx, regAddr=3'bx, matReaden=1'b1

<표12> Mat_FSM의 State Diagram

3.4.3 Mat_Controller submodule

입력신호	설명
clock	메인 clock
nReset	Negative Reset
refAddr[14:0]	기준 픽셀의 주소값
출력신호	설명
adjNumber[2:0]	인접한 점들의 번호
regAddr[2:0]	Register file의 주소값
matReaden	DB 및 Matching Register file read
기능 설명	
Mat_Reg module과 Mat_AddrCal module을 제어하기 위한 신호를 생성하는 모듈이다.	

<표13> Mat_Controller의 신호

INIT state에서 신호의 초기 값을 설정해주고, 다음 state에서는 각 인접한 픽셀들의 신호를 생성한다. regAddr[2:0]는 Mat_Reg에 바로 들어가지만 refAddr[14:0]는 Mat_AddrCal을 거쳐서 Mat_Reg module에 도달한다.

두 신호가 도착하는 시간 차이를 고려하여 regAddr와 refAddr는 2cycle만큼 차이가 발생하도록 조절한다.

3.4.4 Mat_AddrCal submodule

입력신호	설명
refAddr[14:0]	기준 픽셀의 주소값
adjNumber[2:0]	인접한 점들의 번호
출력신호	설명
FBAddr[14:0]	기준 픽셀의 주소를 이용하여 계산된 이웃 픽셀들의 주소
기능 설명	
기준 픽셀의 주소를 바탕으로 기준픽셀에 인접한 16개의 픽셀의 주소값을 계산 하는 모듈이다.	

<표14> Mat_AddrCal의 신호

인접한 픽셀들의 번호(adjNumber[2:0])들에 대해 refAddr를 기준으로 인접 픽셀의 주소값을 계산하여 FBAddr[14:0]에 저장한다.

3.4.5 Mat_Reg submodule

입력신호	설명
clock	메인 clock
nReset	Negative Reset
matReaden	DB 및 Matching Register file read
regAddr[2:0]	Register file의 주소값
FBData[7:0]	기준 픽셀의 주소를 이용하여 계산된 이웃 픽셀들의 주소
출력신호	설명
adjFBPixel[63:0]	SAD연산을 위한 주변 8개의 픽셀 값
기능 설명	
기준 픽셀과 기준 픽셀 주변 8개의 픽셀 값을 저장하는 모듈이다.	

<표15> Mat_Reg의 신호

Mat_AddrCal의 출력 신호인 FBAddr[14:0]값을 바탕으로 Frame Buffer에서 픽셀 값을 가져온다. 이 값을 FBData[7:0]에 저장한다. 이 FBData[7:0]를 이용하여 기준 픽셀(refPixel)값과 주변 픽셀(adjPixel)값을 저장한다.

3.4.6 Mat_Datapath submodule

입력신호	설명
adjFBPixel[63:0]	SAD연산을 위한 주변 8개의 픽셀
dbValue[287:0]	DB 이미지의 모든 특징점 계산 값
출력신호	설명
matPoint	기준 픽셀의 Matching 결과
기능 설명	
1개의 특징점의 주변 평균값을 계산하고 해당 평균 값과 DBMEM에 저장된 모든 값을 비교하는 모듈이다.	

<표16> Mat_Datapath의 신호

adjFBPixel[63:0]을 8비트씩 나눠 모두 더한 다음 8로 나누면 평균 값인 refAvg를 구할 수 있다. dbValue값을 8bit씩 나눈 뒤 dbValue-refAvg 값이 임계값(threshold) 이하이면 matPoint가 1이 된다.

3.4.7 Mat_Counter submodule

입력신호	설명
clock	메인 clock
nReset	Negative Reset
matPoint	기준 픽셀의 Matching 결과
isFeature[7:0]	특징점 확인 변수
출력신호	설명
posAddr[3:0]	Register file의 주소값
posReaden	Matching Register의 데이터를 readen
기능 설명	
Matching이 된 특징점의 개수를 count하는 모듈이다.	

<표17> Mat_Counter의 신호

matPoint가 1이고 isFeature이 7'bx가 아닌 경우에만 다음 레지스터에 매칭된 특징점 주소를 저장하게 한다. 16개의 픽셀이 매칭 되었으면 posReaden값이 1이 된다.

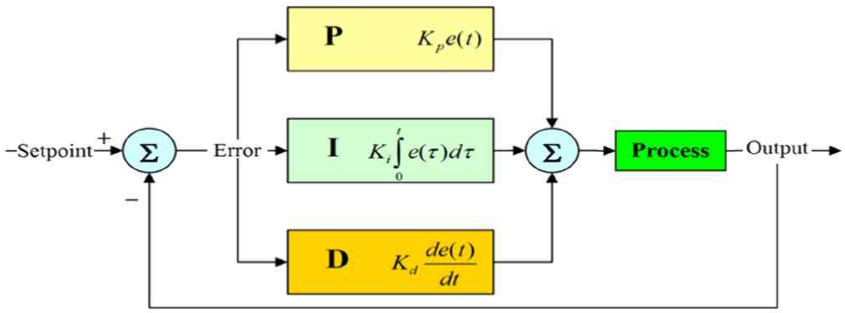
3.4.8 Addr_Reg submodule

입력신호	설명
refAddr[14:0]	기준 픽셀의 주소값
posAddr[3:0]	Register file의 주소값
posReaden	Matching Register의 데이터를 readen
출력신호	설명
position[239:0]	유효 특징점 주소
isMatching	매칭 여부 판별
기능 설명	
매칭된 픽셀 주소를 저장하는 모듈이다.	

<표18> Addr_Reg의 신호

16개의 특징점 주소를 레지스터에 저장하고, 16개의 픽셀 주소가 모두 저장되면 주소를 한번에 묶어서(position[239:0]) 출력한다.

Ⅳ. 설계한 드론 제어



<그림7> PID 피드백 구조

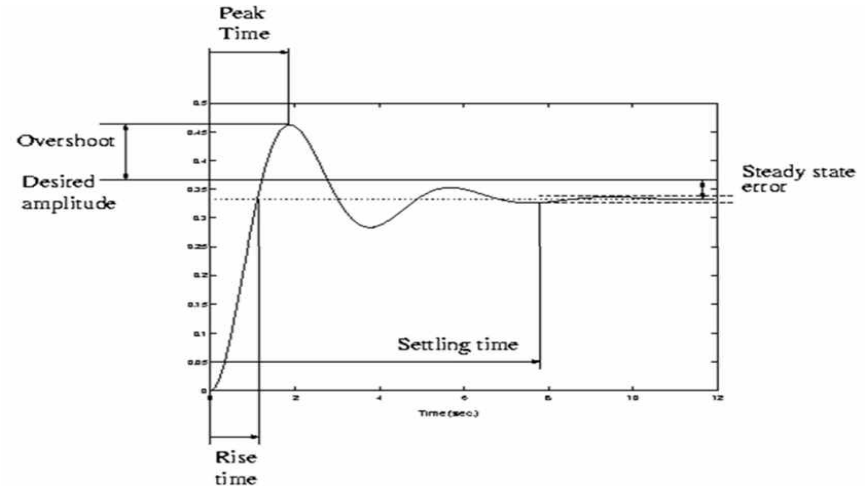
$$MV(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de(t)}{dt} \qquad \text{(식3)}$$

PID제어는 P(Proportional Controller), I(Integral Controller) 그리고 D(Derivative Controller)인 세 가지를 이용하여 해당 목표에 도달하기 위해 도와준다. 이 때 사용하는 공식은 (식3)을 사용한다. 또한 PID는 <그림7>과 같이 피드백 구조를 이루고 있기 때문에 계속하여 측정한 목표가 해당 목표보다 작을 경우에는 그 목표에 도달하기 위하여 상승시켜주고 해당 목표보다 작을 경우에는 하강시켜주도록 도와주는 역할을 수행한다.

P(Proportional Controller)는 비례 제어기이다. 이 비례 제어기의 역할은 현재 상태에서 오차 값의 크기에 비례한 제어 작용을 한다. 이에 따라 1)Steady state error가 발생한다.
I(Integral Controller)는 적분 제어기이다. 이 적분 제어기의 역할은 위의 비례 제어기에서 발생한 Steady state error를 제거하도록 도와주는 역할을 한다. 하지만 2)Overshoot가 발생할 뿐만 아니라 3)Settling time이 전보다 더 증가하게 된다. 그리고 마지막으로 D(Derivative Controller)는 미분 제어기이다. 미분 제어기의 역할은 비례 제어기로부터 발생한 Overshoot를 줄이도록 하는 역할을 할 뿐만 아니라 안정성을 더욱 향상시켜준다. 하지만 이러한 안전성이 향상되는 반면에 주변 noise에 매우 민감하게 반응한다.

---

1) 정상 상태와 현재 상태와 비교한 차이
2) 최고 peak값과 목표값과의 차이
3) 현재 상태가 최종 상태의 일정비율(보통5%) 이내로 되는 데 필요한 시간



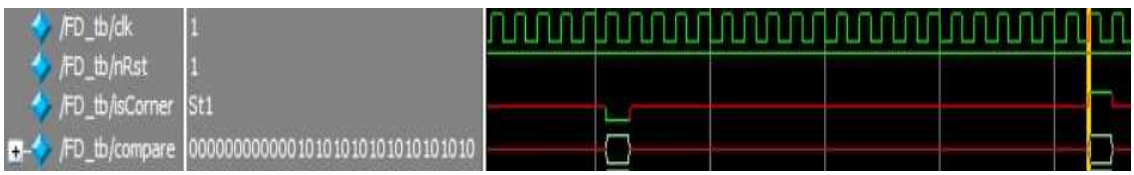
<그림8> PID 시간에 따른 상태 변화

Ⅴ. 실험 결과

5.1 영상처리 결과

<그림9> 알파벳 H를 C++로 실행한 결과

C++언어를 이용하여 FAST-9 알고리즘을 구현하였다. 그리고 실제 사용하는 'H' 이미지를 이용하여 프로젝트를 진행하였다. 그 결과 해당 이미지에 각 모서리에 빨간점으로 이루어진 특징점들을 발견하였다. 그리고 Verilog언어를 이용하여 FAST-9의 각 단계 module별로 테스트를 해 보았다. 아래 <그림10>은 Feature_Detection Module의 입출력 신호의 파형 사진이다.



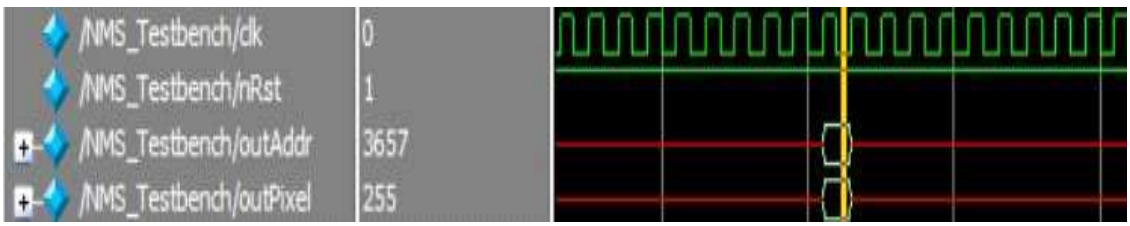
<그림10> Feature_Detection Module 결과

Feature_Detection의 결과 신호인 isCorner 출력 신호가 초록색으로 반응한 부분이 두 구간이 존재한다. 두 구간 중 오른쪽 구간만 후보 특징점이다. 그리고 compare이라는 출력 신호

는 2진수 32bit인 00000000000010101010101010101010 인 출력 신호가 생성되어졌다. 이 출력 신호를 00은 SIMILAR(유사한 밝기)로 10은 BRIGHT(더 밝은 상태)로 적용시킨다. 그리고 compare출력신호를 S 또는 B라는 글자로 해석한다면 SSSSSSBBBBBBBBBB이라는 결과값이 나오게 된다. 그 결과 B가 연속적으로 10개가 존재하기 때문에 Feature_Detection의 모듈을 통해서 해당 픽셀은 후보 특징점으로 분류된다. 아래 <그림11>은 Feature_Detection Module의 다음 단계인 Feature_Score Module의 입출력 신호 결과인 파형 사진이다. Feature_Score단계에서는 Feature_Detection으로부터 도출된 후보 특징점들의 score를 계산하는 단계이다.

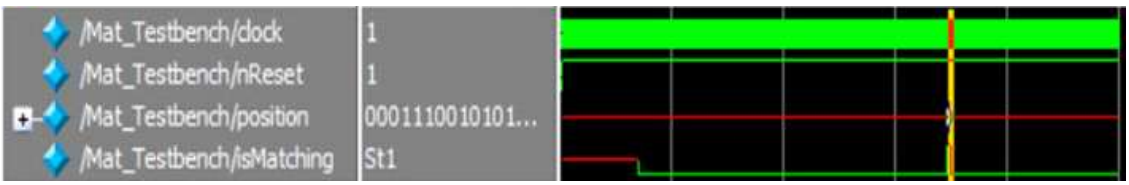
<그림11> Feature_Score Module 결과

이에 따라 <그림11>에서 알 수 있듯이 3656번째 픽셀 주소값에 대한 score를 구하였으며 결과값은 202이라는 값이 나온다는 것을 알 수 있었다. 그리고 계속하여 Feature_Detection에서 도출된 후보 특징점들의 score들이 신호를 따라 계속해서 결과값이 나온다는 것을 알 수 있다. 아래 <그림12>는 FAST-9알고리즘 단계 중에서 마지막 단계인 Non-Maximal Suppression의 입출력 신호 결과인 파형 사진이다. 이 단계를 거치면 특징점 후보들의 score가 기준 픽셀의 score보다 클 경우에는 기준 픽셀을 특징점 후보에서 제외를 하고 작을 경우에는 기준 픽셀을 특징점 후보로 저장하게 해주는 역할을 한다.



<그림12> Non-Maximal Suppression Module 결과

위의 <그림12>을 보면 outAddr과 outPixel이라는 출력신호의 결과값들이 전에 FAST-9알고리즘을 C++로 구현하였을 때의 결과값들과 비교를 하면 서로 같다는 것을 알 수 있었다. 아래 <그림13>은 Matching 단계를 거친 최종적으로 나온 입출력 신호 결과인 파형 사진이다.



<그림13> Matching Module 결과

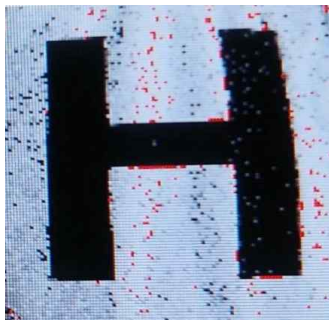
Non-Maximal Suppression단계에서 추출한 최종 특징점들과 DBMemory에 저장되어있는 특징점들의 값들과 비교하여 카메라로 보여진 이미지의 매칭 여부를 판단하고 만약 매칭이 되었다면 해당 이미지의 특징점들의 주소 16개를 출력하게 된다. <그림13>을 본다면 position출력 신호가 생성되었다는 것을 알 수 있고 결과값이 2진수로 총 240bit로 이루어져있다는 것을 알 수 있다.

Verilog로 작성된 module들을 FPGA에 적재하기 전에 그레이스케일(Grayscale)로 변환시켰다. 그리고 해당 영상의 크기를 메모리의 용량을 고려하여 180x120으로 조절하여 적재하였다.

<그림14> Grayscale 변환 전

<그림15> Grayscale 변환 후

<그림16> 180x120 화면모습



<그림17> FPGA칩의 모니터 실행화면

지금까지 작성한 Verilog코드들을 메인 MCU인 AMAZONE2 Processor에 적재하여 실행하였다. 실제로는 위의 <그림9>처럼 그림 속 'H 이미지'의 각 모서리 근처에만 붉은 점이 위치하는 것이 이상적이지만, 실제로 촬영하였을 때에는 특징점들(붉은 점)이 과하게 많이 발견되었다. 다른 부분보다 'H 이미지'의 테두리 근처에 특징점이 많이 찍히긴 하였으나, 그럼에도 불구하고 아무것도 없는 흰 화면에서 특징점들이 찍혔다는 점은 예상했던 것과 다른 결과를 보였다.

5.2 드론비행 결과

PID공식(식3)을 이용하여 드론의 고도를 40cm로 유지하며 안정적으로 비행시키고자 하였다. 처음 실험을 할 때에는 Kp:Ki:Kd를 2:5:1비율로 조절하고 드론을 동작시켰지만, 드론이 목표 고도를 제대로 유지하지 못 하였다. 그래서 각 Kp:Ki:Kd 비율들을 차례로 변경시켜 드론이 안정되게 [4]호버링을 할 수 있도록 수치를 조금씩 변경해주었다. 실험 결과 Kp를 증가시킬수록 Steady-state error은 줄어들었지만, Overshoot은 증가되었다. Kp의 계수를 2로 조절하여 최적의 결과를 얻을 수 있었다. 그리고 이번에는 Ki를 증가시켰다. Steady-state error는 제거되었지만 Overshoot와 settling time은 증가되었다. Ki의 계수를 7로 하는 것이 최적의 결과 값이었다. 그리고 Kd의 계수를 증가시켰다. 목표 지점에 드론이 빨리 도달하였지만 noise가 있을 경우 드론이 늦게 목표 지점에 도달하였다. Kd의 계수를 1로 하는 것이 최적의 결과 값이었다. 결론적으로 Kp:Ki:Kd를 2:7:1비율로 조절하였더니 최적의 결과값을 얻을 수 있었다.

그리고 드론이 비행을 하였을 때 필요한 Throttle, Roll, Pitch, Yaw이라는 변수들이 존재한다. 이때 처음 디폴트값으로 설정하여 진행하였을 때 드론이 무게 중심을 잡지 못하는 경우가 발생하였다. 그래서 수동조작을 하면서 각 변수들이 하는 역할에 따른 그의 값들을 점진적으로 값을 증감시켰다. 그 결과 Throttle의 값을 1500, Roll의 값을 1497, Pitch의 값을 1498

4) 제자리에서 비행을 유지하는 것

그리고 Yaw의값을 1500으로 변경시켜 더욱 안정되게 비행을 할 수 있게 해주었다.

〈그림18〉 호버링 실패

〈그림19〉 호버링 성공

Ⅵ. 결론 및 기대효과

기존에 영상을 인식하는 소프트웨어 프로그램은 속도가 느리다는 단점이 존재한다. 이러한 단점을 개선하기 위하여 하드웨어 언어로 프로그램을 만들어 기존 영상 인식 속도보다 빠르게 처리할 수 있도록 할 것이다. 그리고 해당 물체를 추적할 수 있는 알고리즘을 만들어 실생활 및 다양한 분야에 접목시킬 수 있도록 할 것이다.

프로젝트를 4차 산업 분야 중 자율 주행 분야에 활용 할 수 있다. 그리고 사물 및 영상을 인식하는 속도를 빠르게 하여 영상인식 분야를 발전시킬 수 있다. 자율 주행 드론은 특정한 사물을 카메라로 인식하고 해당 사물의 움직임에 따라서 비행이 가능하다. 이 기술을 군사용 및 치안 분야에 접목시킬 수 있다. 군사용으로 사용하는 경우, 드론을 국가 간의 경계 초소에 설치하여 적 군인이 나타날시 카메라로 아군 및 적군을 판별할 수 있다. 그리고 적군들을 상공에서 적군의 눈에 보이지 않게 추적할 수 있다. 이를 통해 국가 안보에 많은 도움이 될 수 있다는 기대효과가 존재한다. 치안용으로 사용하는 경우, 성 범죄자들을 감시 및 추적하는 용도로 활용할 수 있다. 성범죄자들의 경우 전자발찌가 있지만 전자발찌를 자르고 도주할 경우가 있다. 그렇기 때문에 해당 성범죄자들이 주로 사는 곳에 해당 드론을 배치하여 평상시에는 특정 위치에서 비행하지 않고 카메라로 CCTV 역할을 하다가 만약 성범죄자가 나타나면 카메라로 인식하여 해당 성 범죄자를 추적 및 관찰하여 성범죄를 예방하는데 사용할 수 있는 기대효과들이 발생할 것이다.

참고문헌

[1] 오일석, “컴퓨터비전”, 한빛아카데미, 2014

[2] R. Edward and T. Drummond, IEEE ICCV, 2005

[3] Taek-kyu Kim, “An Embedded FAST Hardware Accelerator for Image Feature Detection“, Journal of The Institute of Electronics Engineers, Vol. 49, No. 2, pp. 28-34, Mar. 2012

[4] Taek-Kyu Kim, Yong-Suk Suh. (2014). A Threshold Controller for FAST Hardware Accelerator. The Institute of Electronics and Information Engineers, 51(11), 187-192.

[5] V. Bonato, “A Parallel Hardware Architecture for Scale and Rotation Invariant Feature Detection“,IEEE Transactions on Circuits and Systems for Video Technology, Vol. 18, No. 12, pp. 1703-1712, Dec. 2008.

[6] Hoon Heo and Kwang-Yeob Lee, “FPGA based Implementation of FAST and BRIEF algorithm for Object Recognition“, Journal of IKEEE, Vol. 17,pp. 202-207, June 2013

[7] Seon-Jong Kim, Joo-Man Kim, Hyeog-Soong Kwon,(2014) “Design of Software Platform with FPGAs for Video Processing“. Korean Institute of Information Technology, 12(11), 101-109

[8] 유승호, 김황남. (2018). 드론 자율 비행 기술의 소개. 정보와 통신 열린강좌, 35(1(별책7호)), 36-43.

[9] 윤단비, 이규열, 한상기, 김용훈, 이승대. “자이로 센서와 PID 제어를 이용한 드론 비행 안정화에 관한 연구.“ 한국전자통신학회 논문지, (2017): 591-598

[10] 심이삭, 홍승관, 정준희, 차경현, 김진영. (2015)“드론의 안정화 비행을 위한 방법 및 알고리즘에 관한 연구“