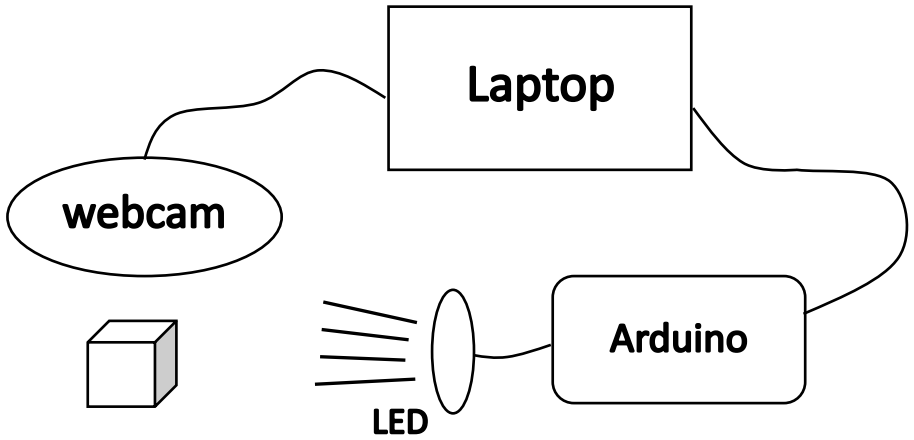
Fuzzy Logic Based Image Quality Control in Low Light Situations



Student: Chia Sheng Kuo

Adviser: Prof. Chi Cheng Cheng

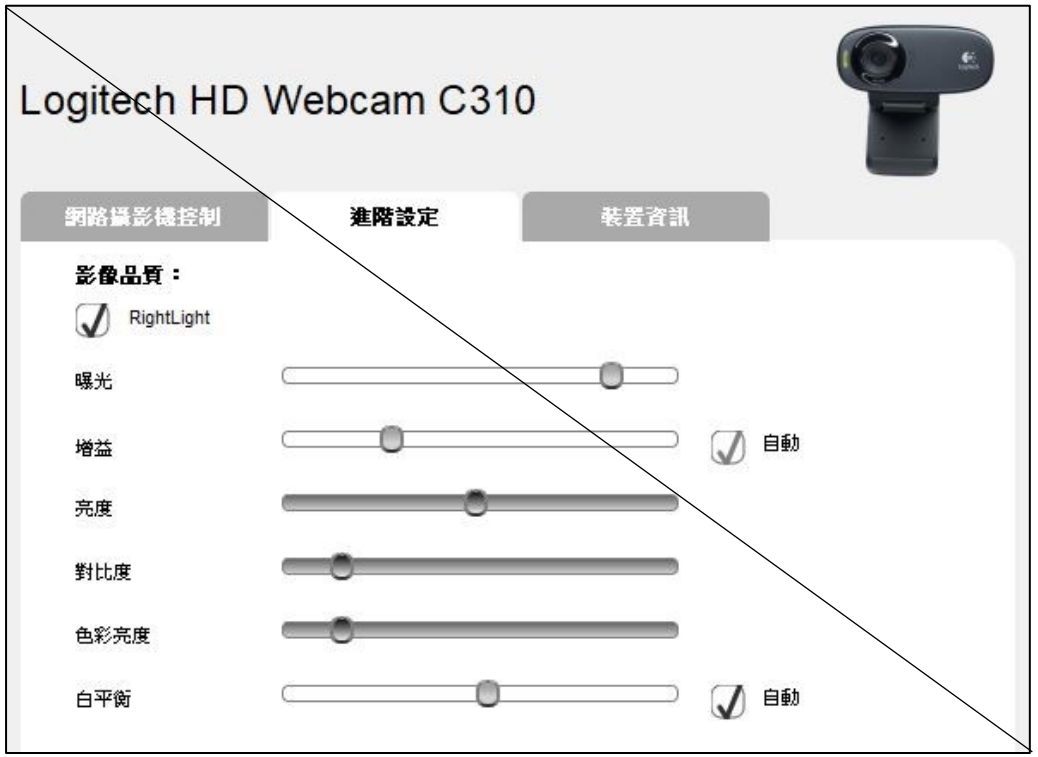
Dept. of Mechanical and Electro-Mechanical Engineering National Sun Yat-Sen University

OUTLINE

- Brief introduction to AE
- System Specification
- Input & Output
- Environment Test
- Fuzzification
- Fuzzy Inference
- Defuzzification
- Result & Conclusion

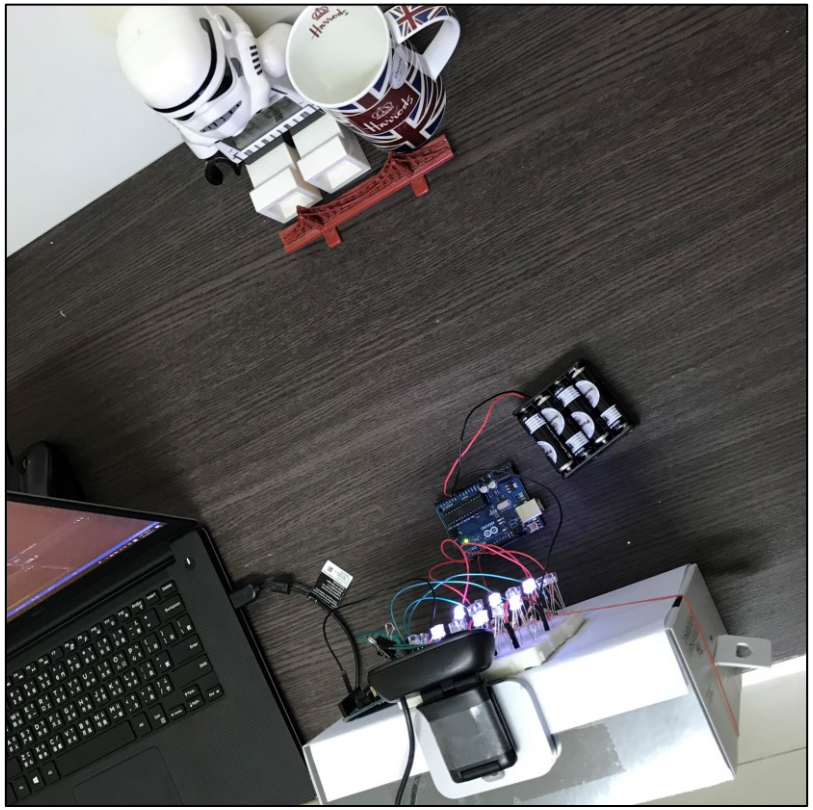
Auto Exposure

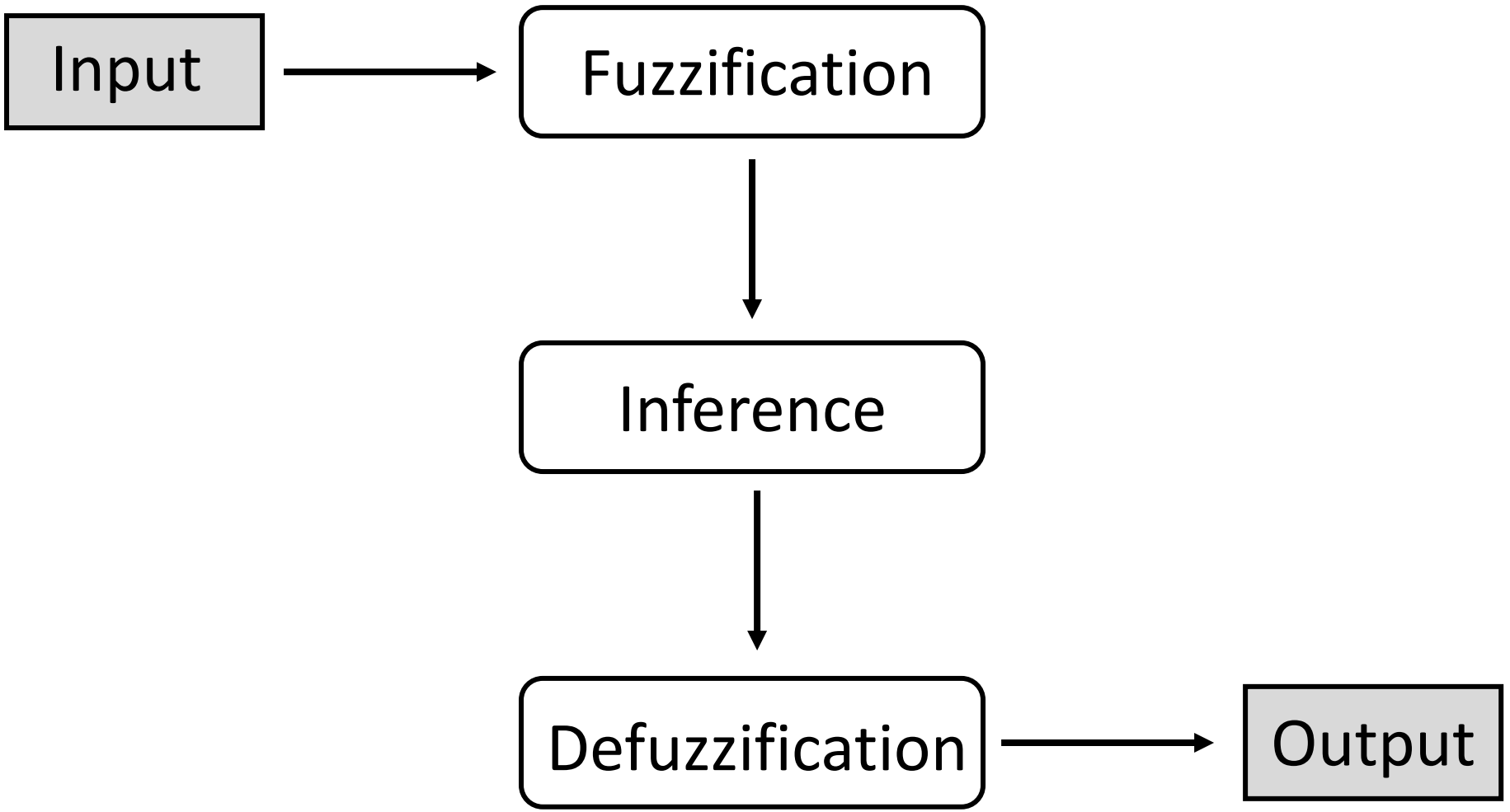
In low light situation, even AE is applied, image quality could be degraded.
That’s the reason why camera needs the flashlight.



Specification

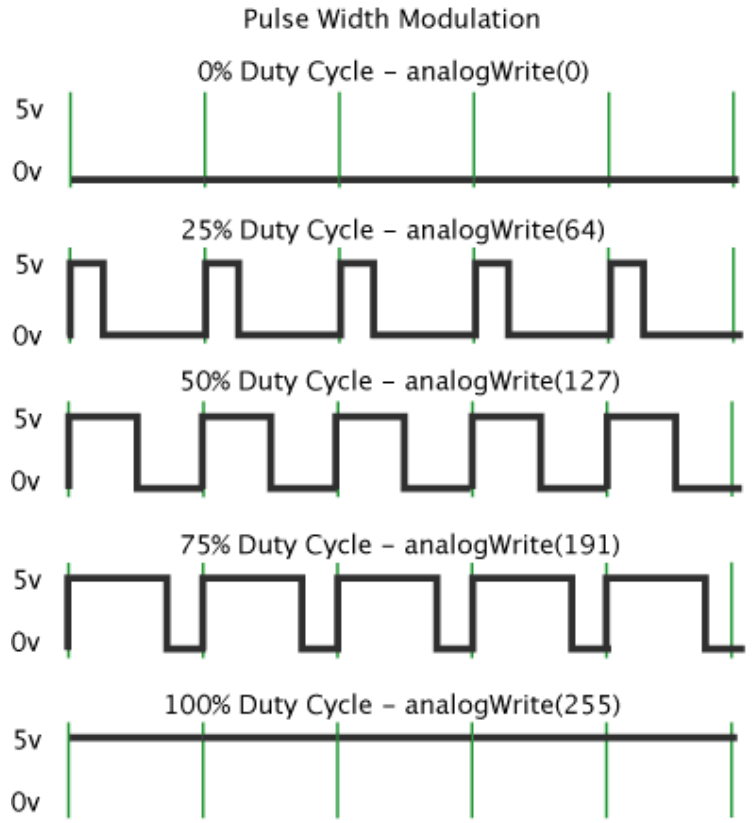
- Laptop (Visual Studio 2017 + OpenCV 3.4.1)
- fuzzylite 6.0
- Logitech HD Webcam C310
- Arduino UNO Board *2 (5V Output)
- LED *30 (Vf = 3.2V, If = 20mA)
- Resistor 30 Ω *6, 47 Ω *6



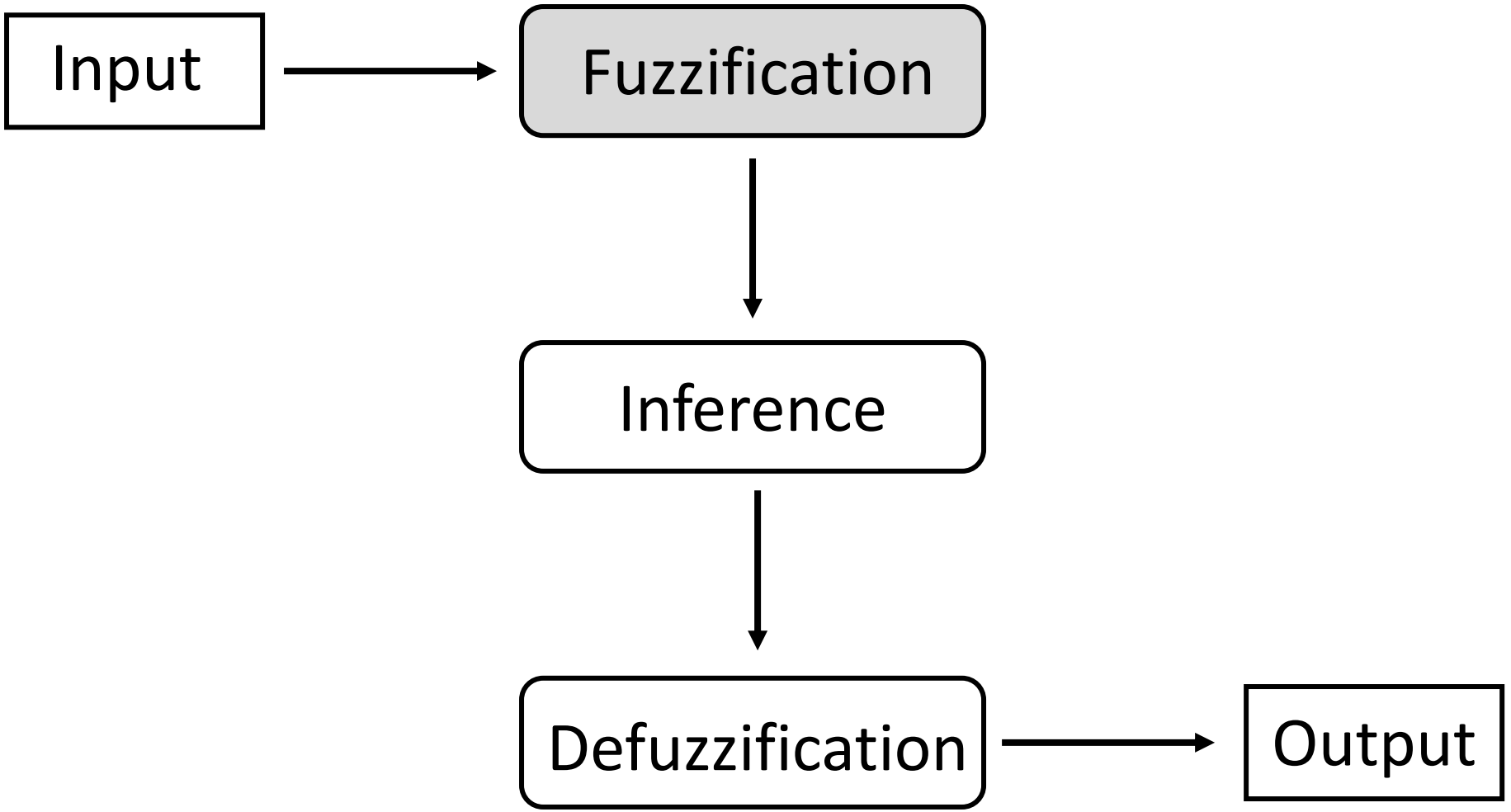

Input
Fuzzification
Inference
Defuzzification
Output

Input & Output of the System

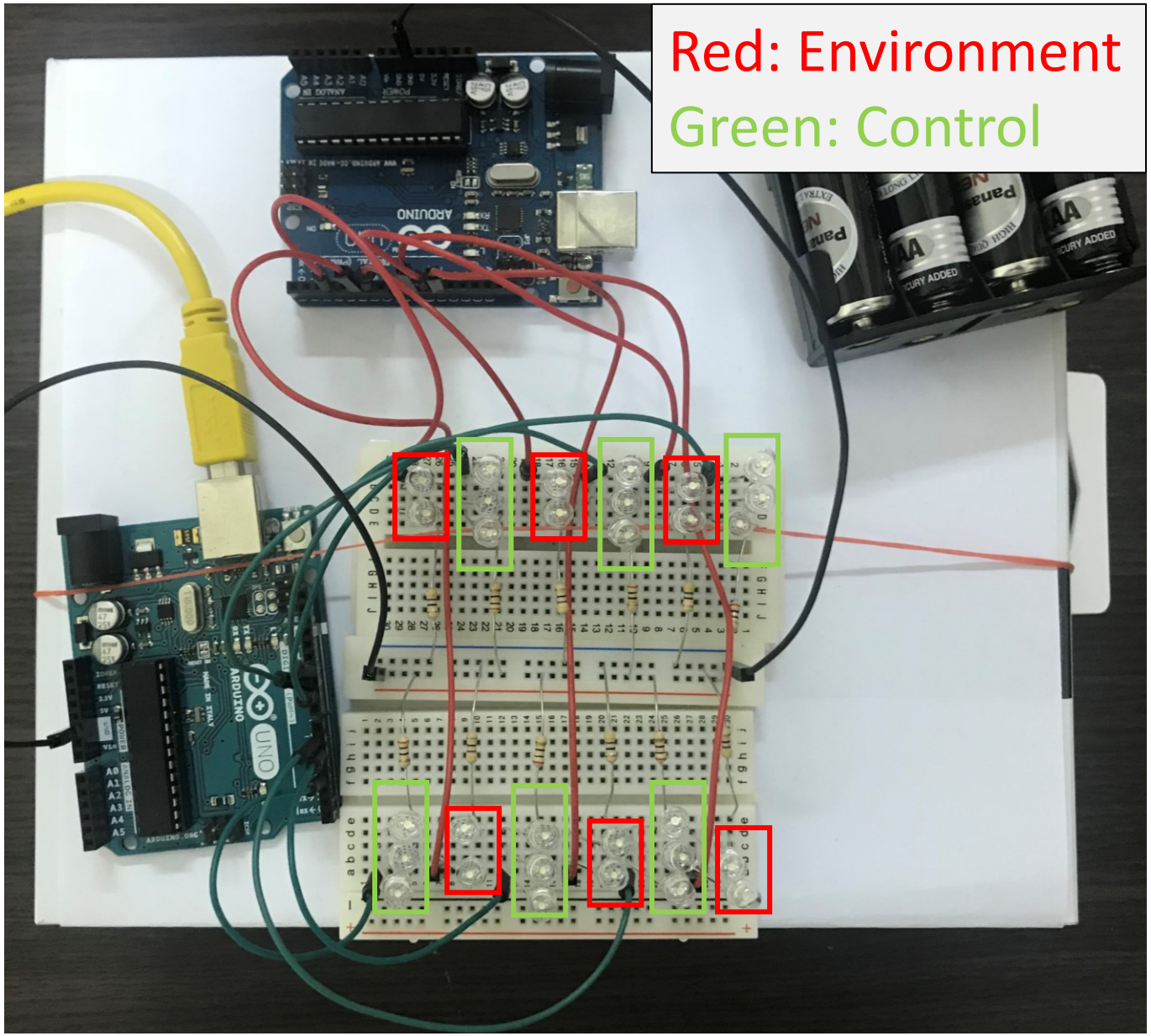
Input	Output
Mean gray value = $\frac{\sum_{j=1}^{m}\sum_{i=1}^{n} x_{ij}}{m*n}$	PWM Signal Resolution 0 – 255
$Contrast(f) = \frac{L\max - L\min}{L\max + L\min} * 100$	
$Entropy = -\sum_{i=0}^{255} P(i)\log_2 P(i)$	
$SNR(dB) = 20 * \log_{10}\left(S_{pixel} \Big/ \sqrt{\frac{1}{m*n}\sum_{j=1}^{m}\sum_{i=1}^{n}[x_{ij} - \bar{x}]^2} \right)$	



Before fuzzification, the possible range of the input and output should be acquired.

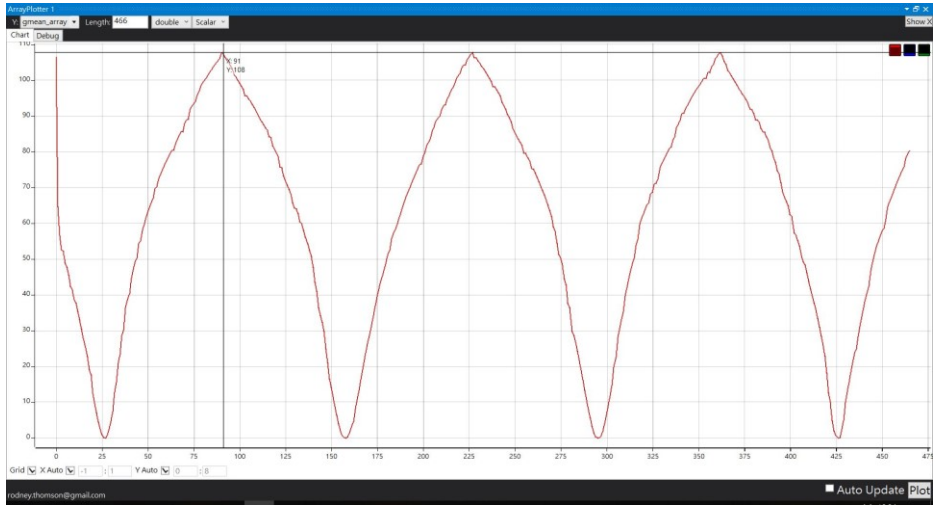


Environment test



Environment:
PWM goes back and forth from 0 to 255.

For example:

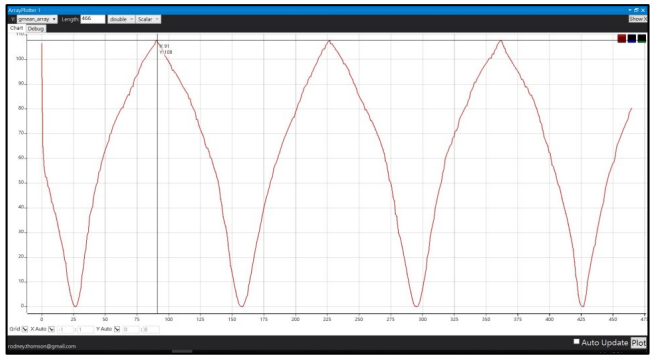


Mean gray value

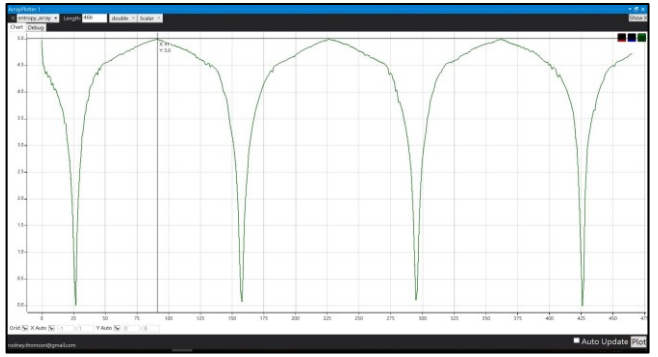
For better understanding

Environment:
PWM goes back and forth from 0 to 255.

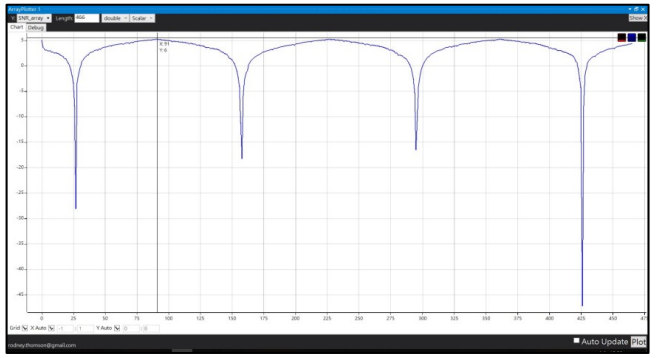
Mean gray value



Entropy

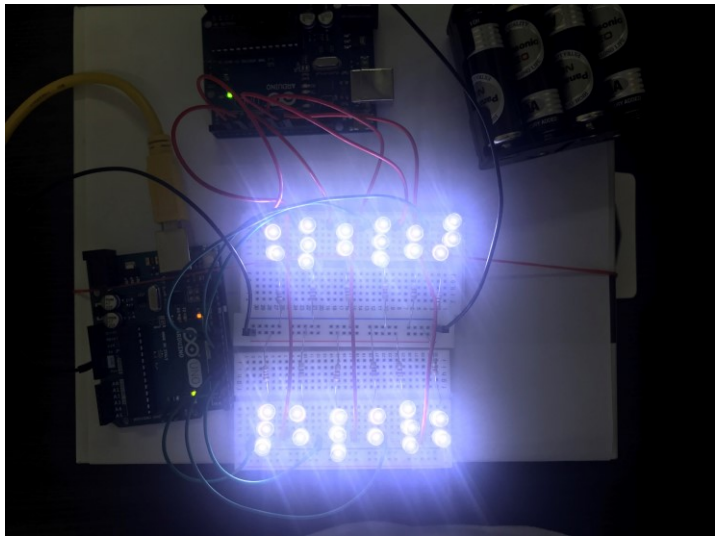


SNR

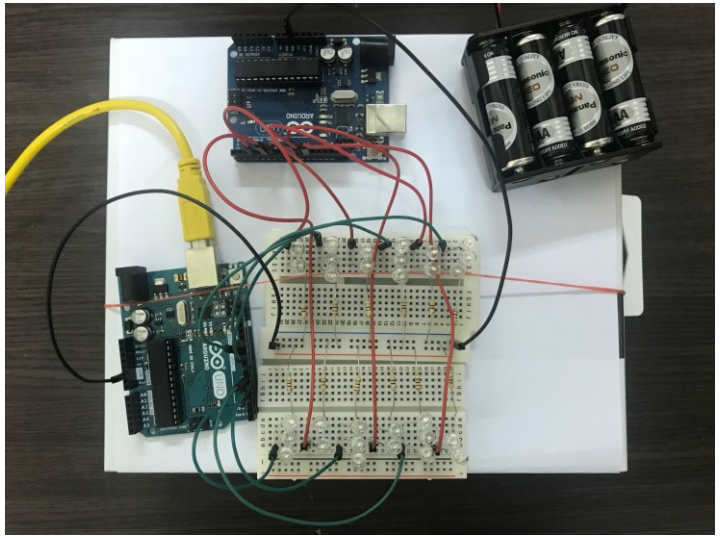


The goal is to make the Input signal smooth even though environment is changing non-stop.

Upper boundary and lower boundary



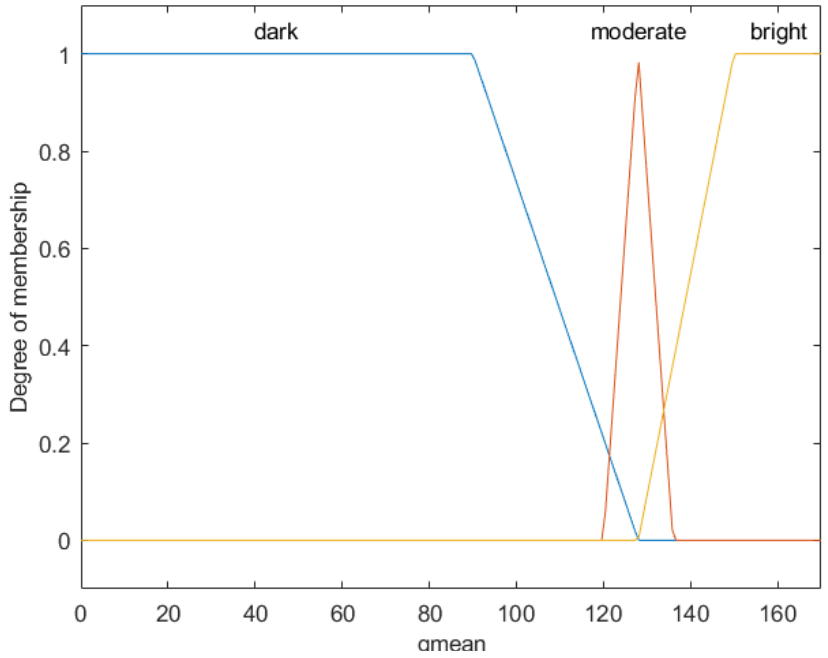
Upper boundary:
Environment(When PWM=255) + Control(PWM=255)



Lower boundary:
Environment(When PWM=0) + Control(PWM=0)

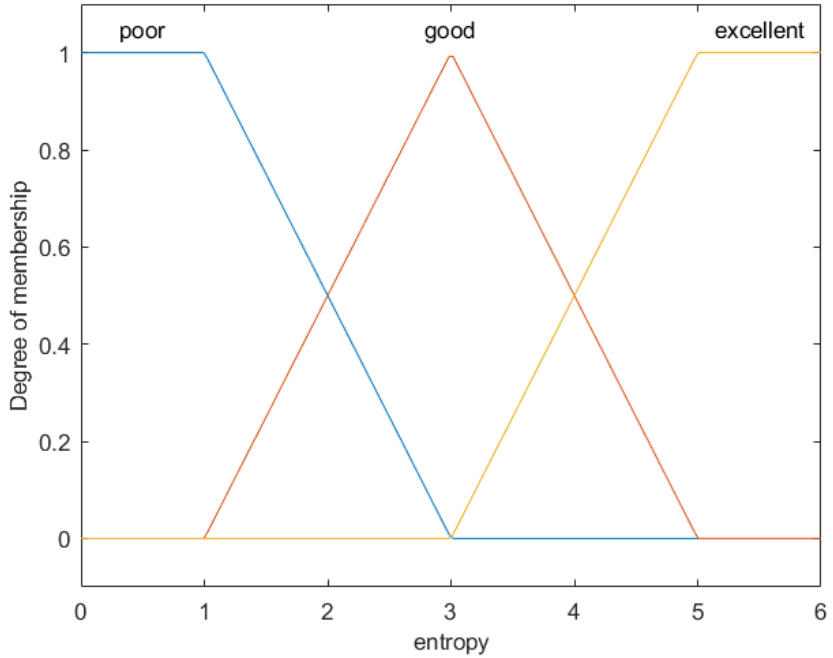
Mathematically, The range of the boundary would be larger the possible.

	Mean gray value	Entropy	SNR	Error	Error rate	PWM
Upper boundary	164	5.22	6.53	?	?	255
Lower boundary	0	0	-48	?	?	0



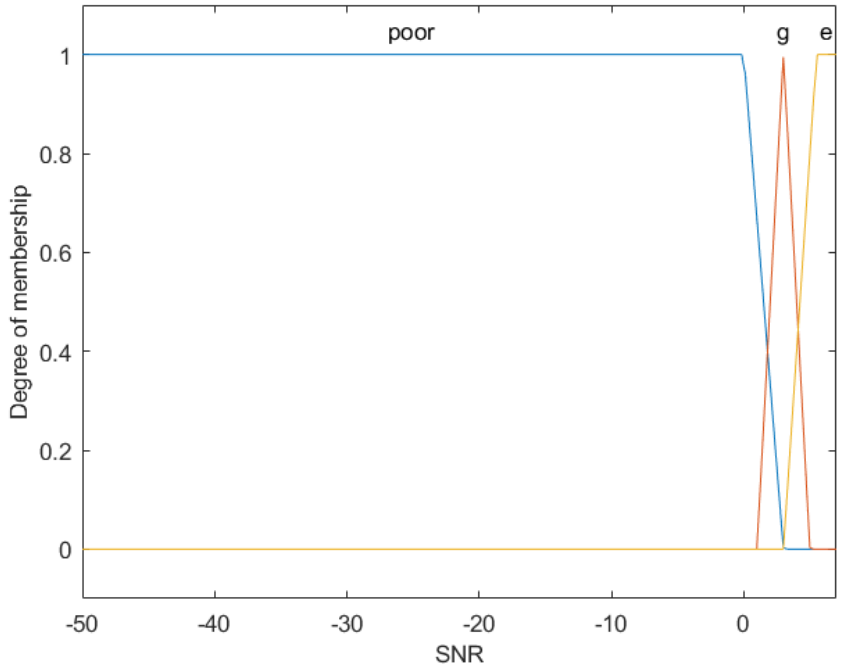
Mean gray value

```
InputVariable* gmean = new InputVariable;
gmean->setName("gmean");
gmean->setDescription("Image grayvalue mean");
gmean->setEnabled(true);
gmean->setRange(0.000, 170.000);
gmean->setLockValueInRange(true);
gmean->addTerm(new Trapezoid("dark", 0.000, 0.000, 90.000, 128.000));
gmean->addTerm(new Triangle("moderate", 120.000, 128.000, 136.000));
gmean->addTerm(new Trapezoid("bright", 128.000, 150.000, 170.000, 170.000));
engine->addInputVariable(gmean);
```



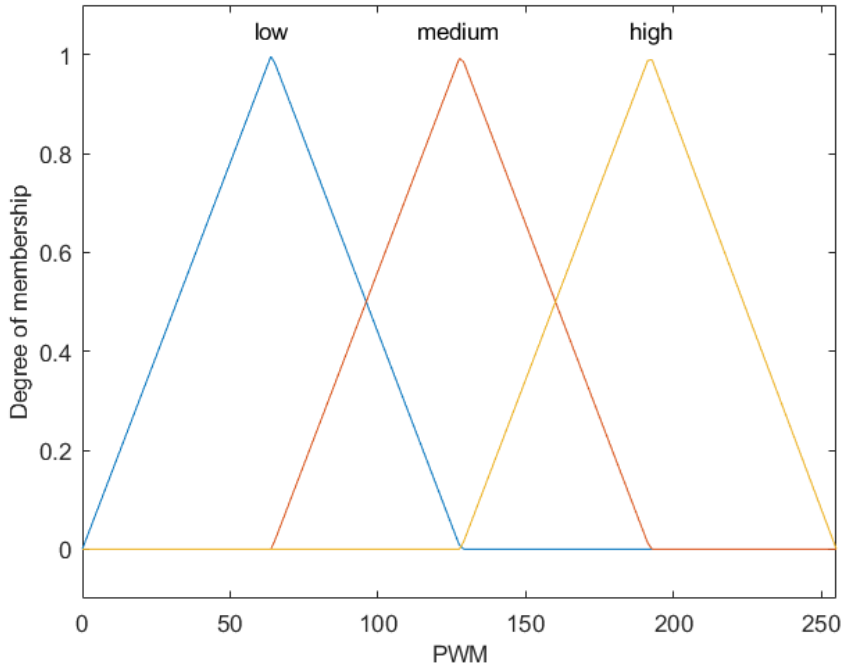
Entropy

```
InputVariable* entropy = new InputVariable;
entropy->setName("entropy");
entropy->setDescription("Image entropy");
entropy->setEnabled(true);
entropy->setRange(0.000, 6.000);
entropy->setLockValueInRange(true);
entropy->addTerm(new Trapezoid("poor", 0.000, 0.000, 1.000, 3.000));
entropy->addTerm(new Triangle("good", 1.000, 3.000, 5.000));
entropy->addTerm(new Trapezoid("excellent", 3.000, 5.000, 6.000, 6.000));
engine->addInputVariable(entropy);
```



SNR

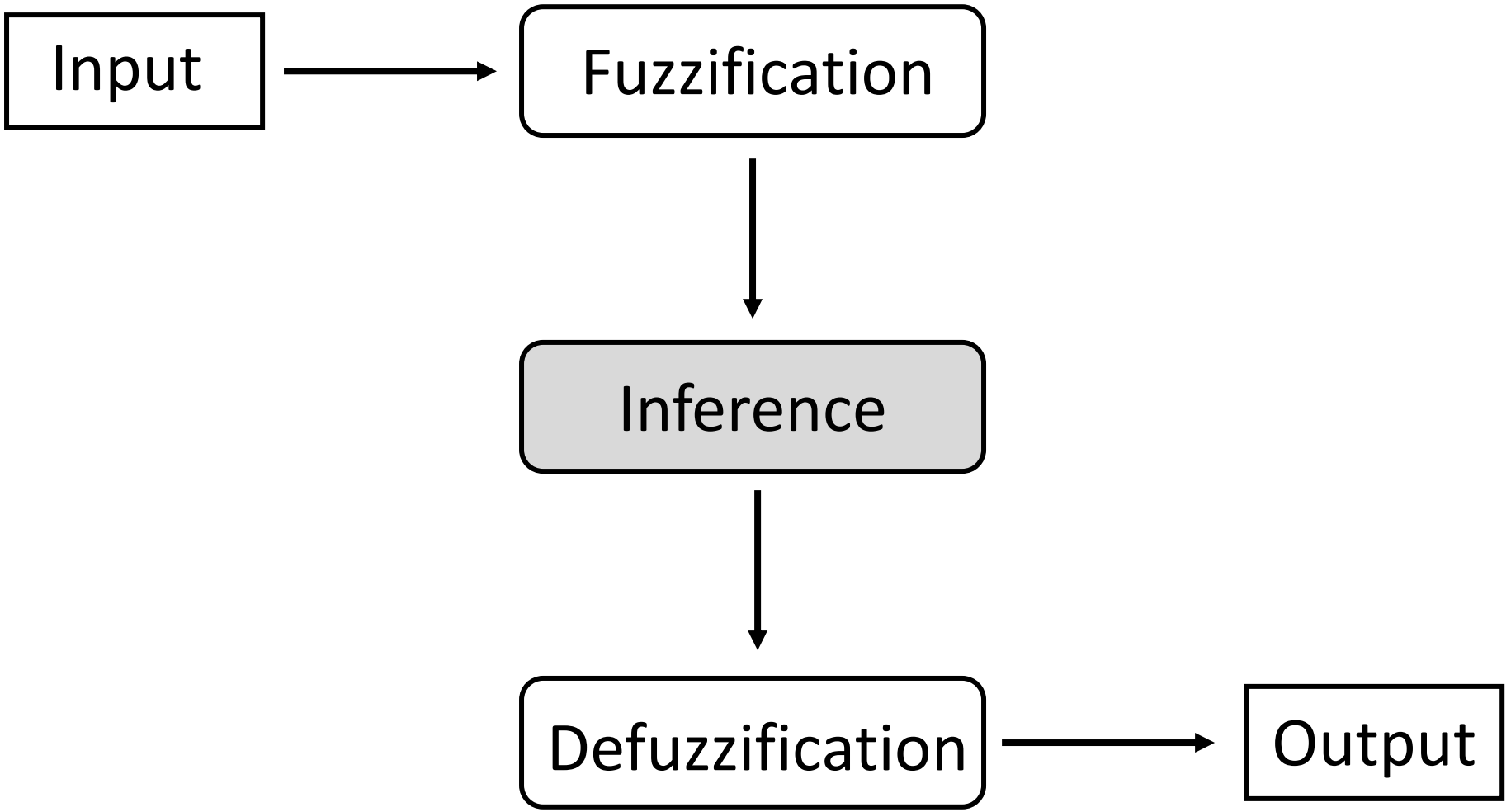
```
InputVariable* SNR = new InputVariable;
SNR->setName("SNR");
SNR->setDescription("Image SNR");
SNR->setEnabled(true);
SNR->setRange(-50.000, 7.000);
SNR->setLockValueInRange(true);
SNR->addTerm(new Trapezoid("poor", -50.000, -50.000, 0.000, 3.000));
SNR->addTerm(new Triangle("good", 1.000, 3.000, 5.000));
SNR->addTerm(new Trapezoid("excellent", 3.000, 5.500, 7.000, 7.000));
engine->addInputVariable(SNR);
```



PWM

```
OutputVariable* mPWM = new OutputVariable;
mPWM->setName("mPWM");
mPWM->setDescription("PWM based on Mamdani inference");
mPWM->setEnabled(true);
mPWM->setRange(0.000, 255.000);
mPWM->setLockValueInRange(false);
mPWM->setAggregation(new Maximum);
mPWM->setDefuzzifier(new Centroid(1000));
mPWM->setDefaultValue(fl::nan);
mPWM->setLockPreviousValue(false);
mPWM->addTerm(new Triangle("low", 0.000, 64.000, 128.000));
mPWM->addTerm(new Triangle("medium", 64.000, 128.000, 192.000));
mPWM->addTerm(new Triangle("high", 128.000, 192.000, 255.000));
engine->addOutputVariable(mPWM);
```

From intuition to PD like fuzzy controller



Inference based on intuition:

- If gmean is dark or SNR is poor then PWM is high.
- If gmean is dark or entropy is poor then PWM is high.
- If SNR is poor or entropy is poor then PWM is high.
- If gmean is bright and entropy is excellent then PWM is medium.
- If gmean is bright and SNR is excellent then PWM is medium.
- If SNR is excellent and entropy is excellent then PWM is medium.

```
RuleBlock* mamdani = new RuleBlock;
mamdani->setName("mamdani");
mamdani->setDescription("Mamdani inference");
mamdani->setEnabled(true);
mamdani->setConjunction(new Minimum);
mamdani->setDisjunction(new Maximum);
mamdani->setImplication(new Minimum);
mamdani->setActivation(new General);
mamdani->addRule(Rule::parse("if gmean is dark or SNR is poor then mPWM is high", engine));
mamdani->addRule(Rule::parse("if gmean is dark or entropy is poor then mPWM is high", engine));
mamdani->addRule(Rule::parse("if SNR is poor or entropy is poor then mPWM is high", engine));
mamdani->addRule(Rule::parse("if gmean is bright and entropy is excellent then mPWM is medium", engine));
mamdani->addRule(Rule::parse("if gmean is bright and SNR is excellent then mPWM is medium", engine));
mamdani->addRule(Rule::parse("if SNR is excellent and entropy is excellent then mPWM is medium", engine));
engine->addRuleBlock(mamdani);
```

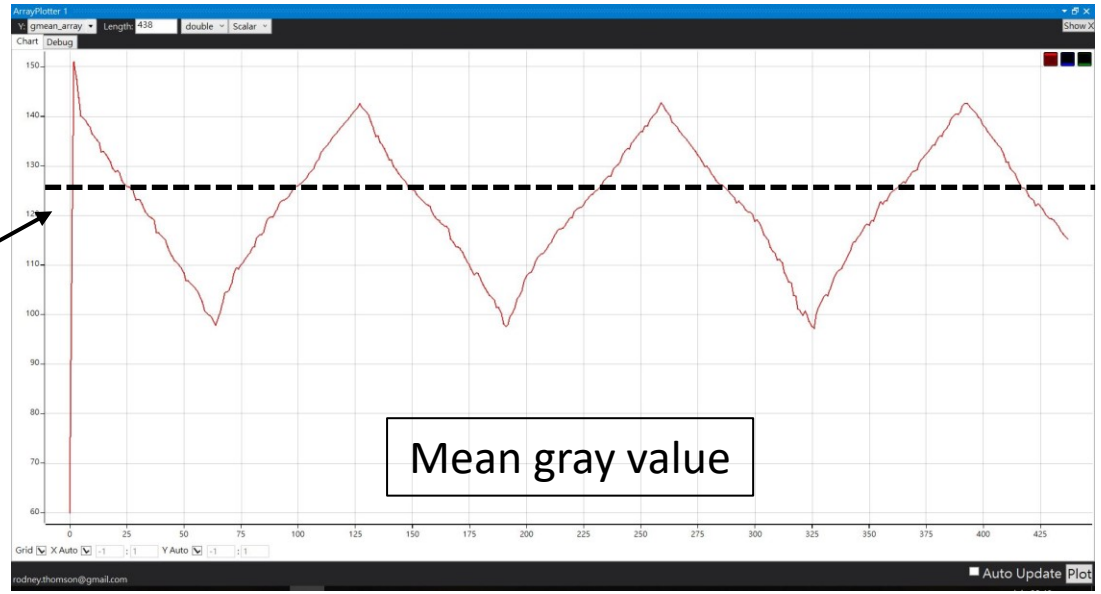
Inference based on intuition

↓

Result is not good. Especially for mean gray value

↓

Set point =128

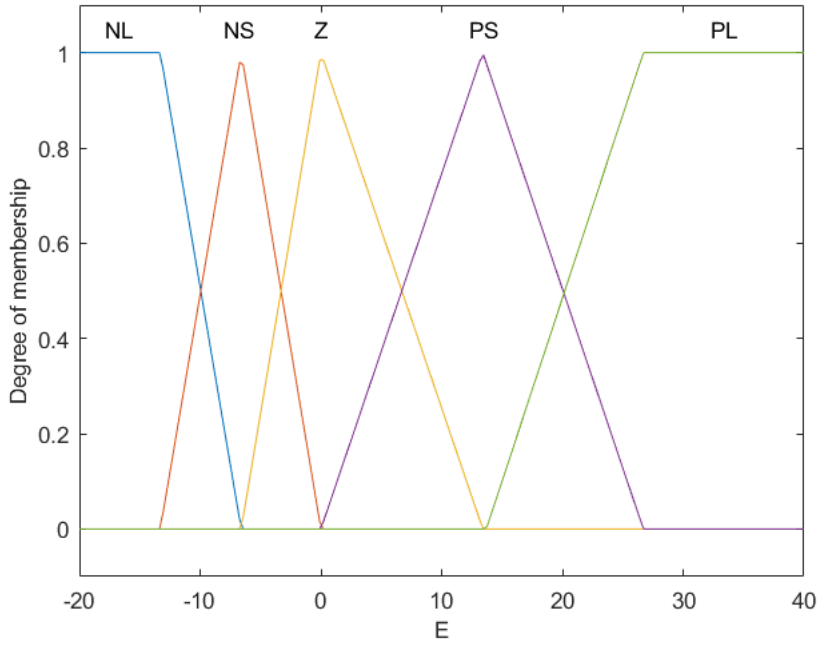


↓

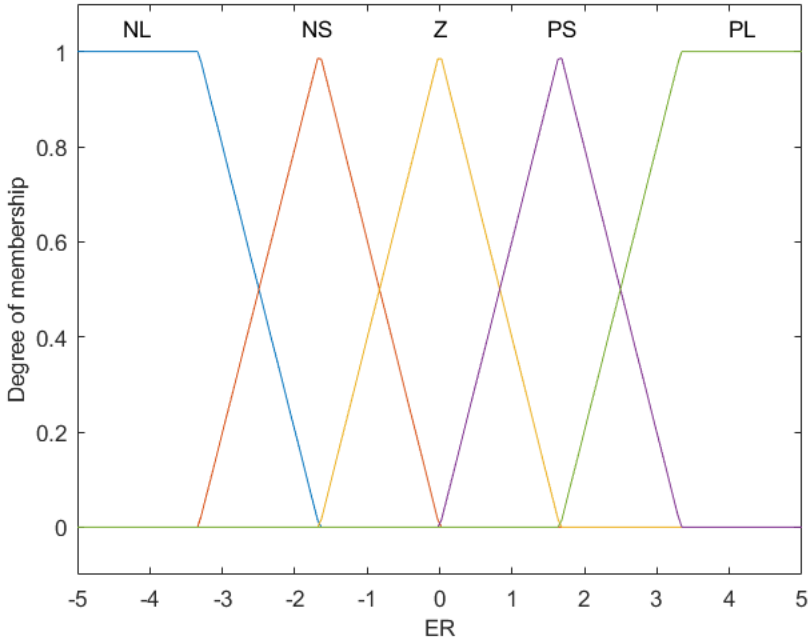
PD like fuzzy Inference

To implement the PD like fuzzy controller, the Error and the Error rate is computed.

	Mean gray value	Entropy	SNR	Error	Error rate	PWM
Upper boundary	164	5.22	6.53	40	5	255
Lower boundary	0	0	-48	-20	-5	0

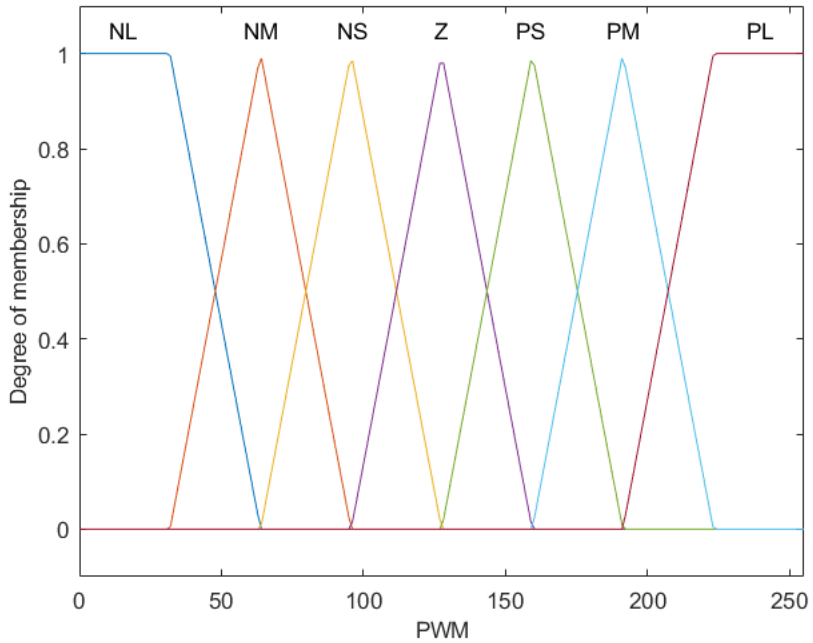


Error



Error rate

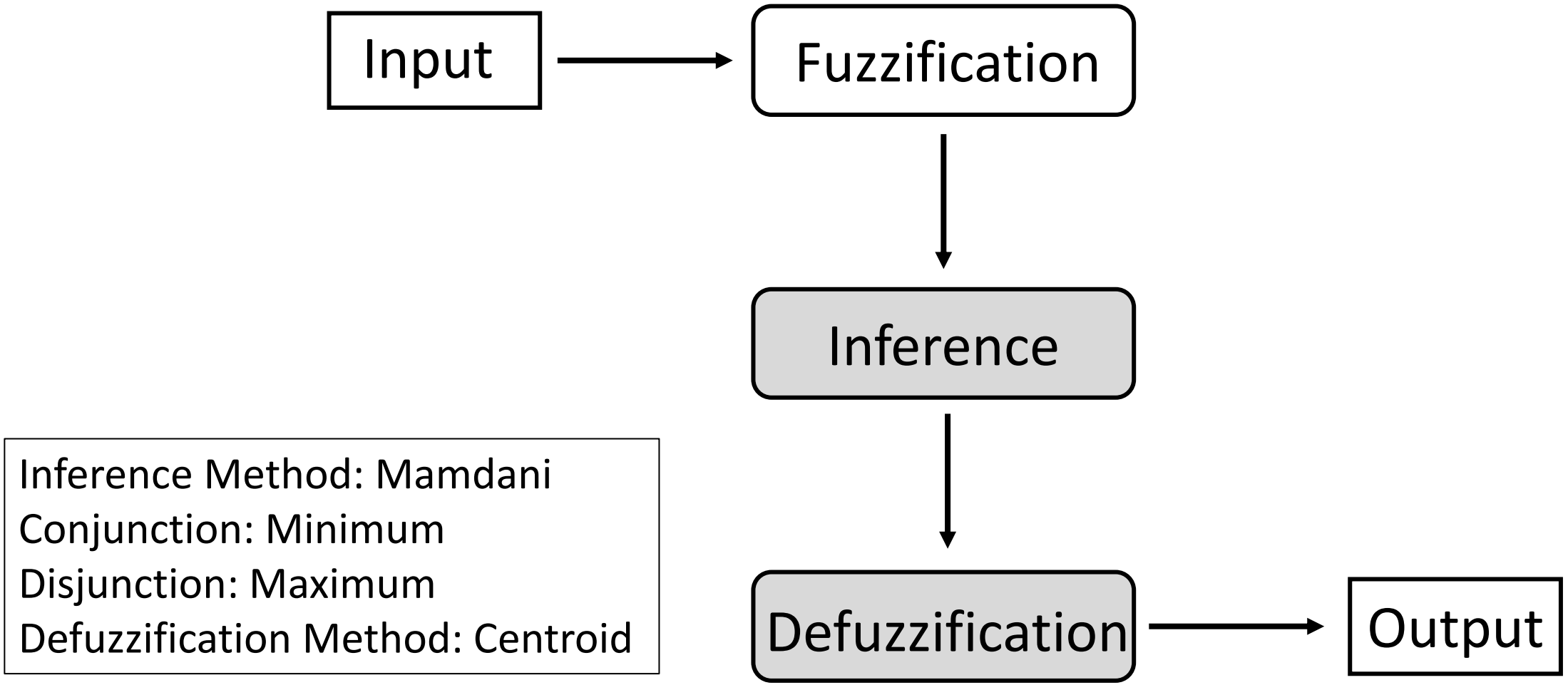
PWM



PD like fuzzy Inference:

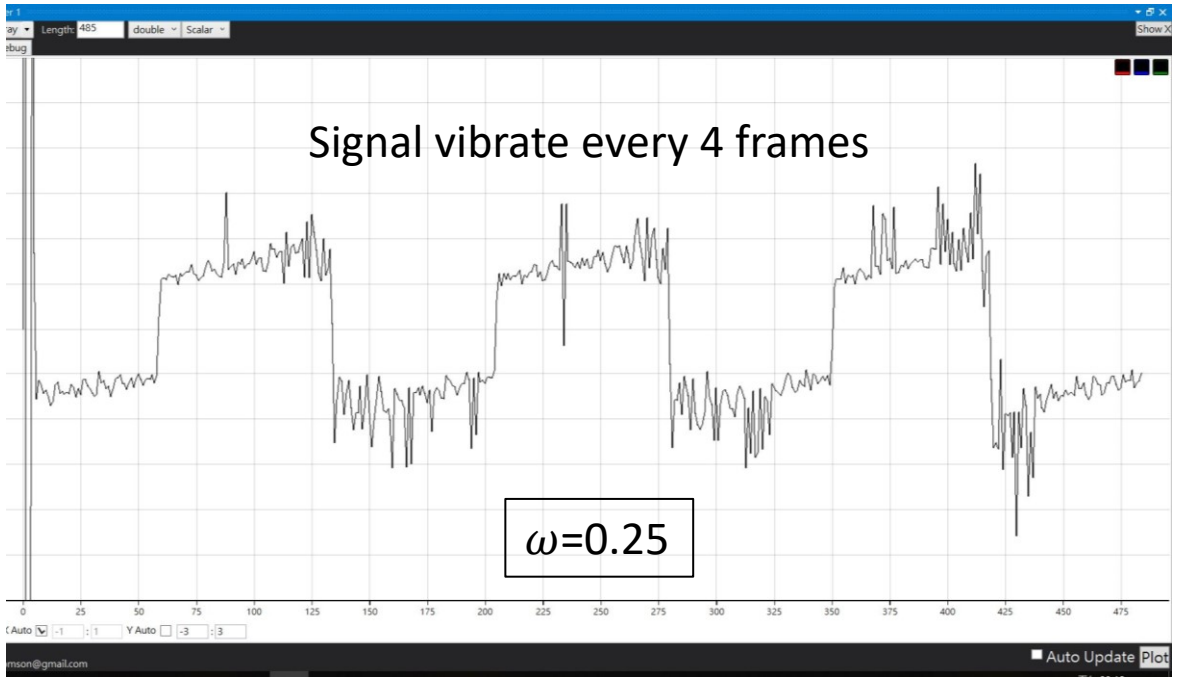
ER \ E	NL	NS	Z	PS	PL
PL	Z	PS	PM	PL	PL
PS	NS	Z	PS	PM	PL
Z	NM	NS	Z	PS	PM
NS	NL	NM	NS	Z	PS
NL	NL	NL	NM	NS	Z

```
//PD LIKE FUZZY
mamdani->addRule(Rule::parse("if E is PL and ER is PL then mPWM is PL", engine));
mamdani->addRule(Rule::parse("if E is PL and ER is PS then mPWM is PL", engine));
mamdani->addRule(Rule::parse("if E is PS and ER is PL then mPWM is PL", engine));

mamdani->addRule(Rule::parse("if E is Z and ER is PL then mPWM is PM", engine));
mamdani->addRule(Rule::parse("if E is PS and ER is PS then mPWM is PM", engine));
mamdani->addRule(Rule::parse("if E is PL and ER is Z then mPWM is PM", engine));

mamdani->addRule(Rule::parse("if E is NS and ER is PL then mPWM is PS", engine));
mamdani->addRule(Rule::parse("if E is Z and ER is PS then mPWM is PS", engine));
mamdani->addRule(Rule::parse("if E is PS and ER is Z then mPWM is PS", engine));
mamdani->addRule(Rule::parse("if E is PL and ER is NL then mPWM is PS", engine));

mamdani->addRule(Rule::parse("if E is PL and ER is NL then mPWM is Z", engine));
mamdani->addRule(Rule::parse("if E is PS and ER is NS then mPWM is Z", engine));
mamdani->addRule(Rule::parse("if E is Z and ER is Z then mPWM is Z", engine));
mamdani->addRule(Rule::parse("if E is NS and ER is PS then mPWM is Z", engine));
mamdani->addRule(Rule::parse("if E is NL and ER is PL then mPWM is Z", engine));


mamdani->addRule(Rule::parse("if E is NL and ER is PS then mPWM is NS", engine));
mamdani->addRule(Rule::parse("if E is NS and ER is Z then mPWM is NS", engine));
mamdani->addRule(Rule::parse("if E is Z and ER is NS then mPWM is NS", engine));
mamdani->addRule(Rule::parse("if E is PS and ER is NL then mPWM is NS", engine));

mamdani->addRule(Rule::parse("if E is NL and ER is Z then mPWM is NM", engine));
mamdani->addRule(Rule::parse("if E is NS and ER is NS then mPWM is NM", engine));
mamdani->addRule(Rule::parse("if E is Z and ER is NL then mPWM is NM", engine));

mamdani->addRule(Rule::parse("if E is NS and ER is NL then mPWM is NL", engine));
mamdani->addRule(Rule::parse("if E is NL and ER is NS then mPWM is NL", engine));
mamdani->addRule(Rule::parse("if E is NL and ER is NL then mPWM is NL", engine));
```

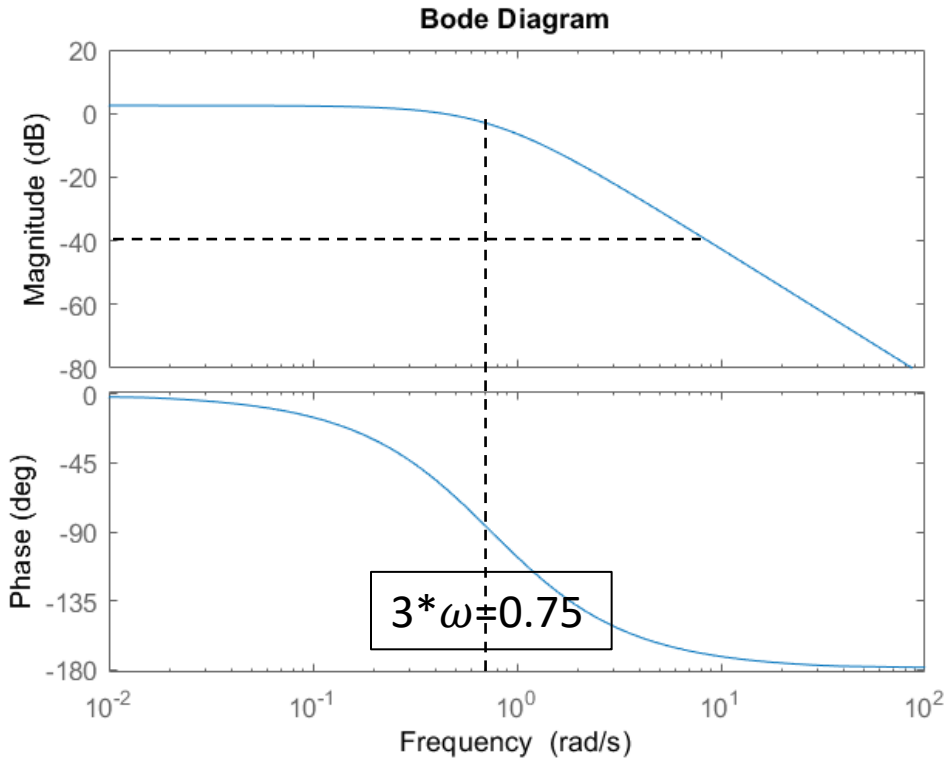

Input
Fuzzification
Inference
Defuzzification
Output
Inference Method: Mamdani
Conjunction: Minimum
Disjunction: Maximum
Defuzzification Method: Centroid

One more thing

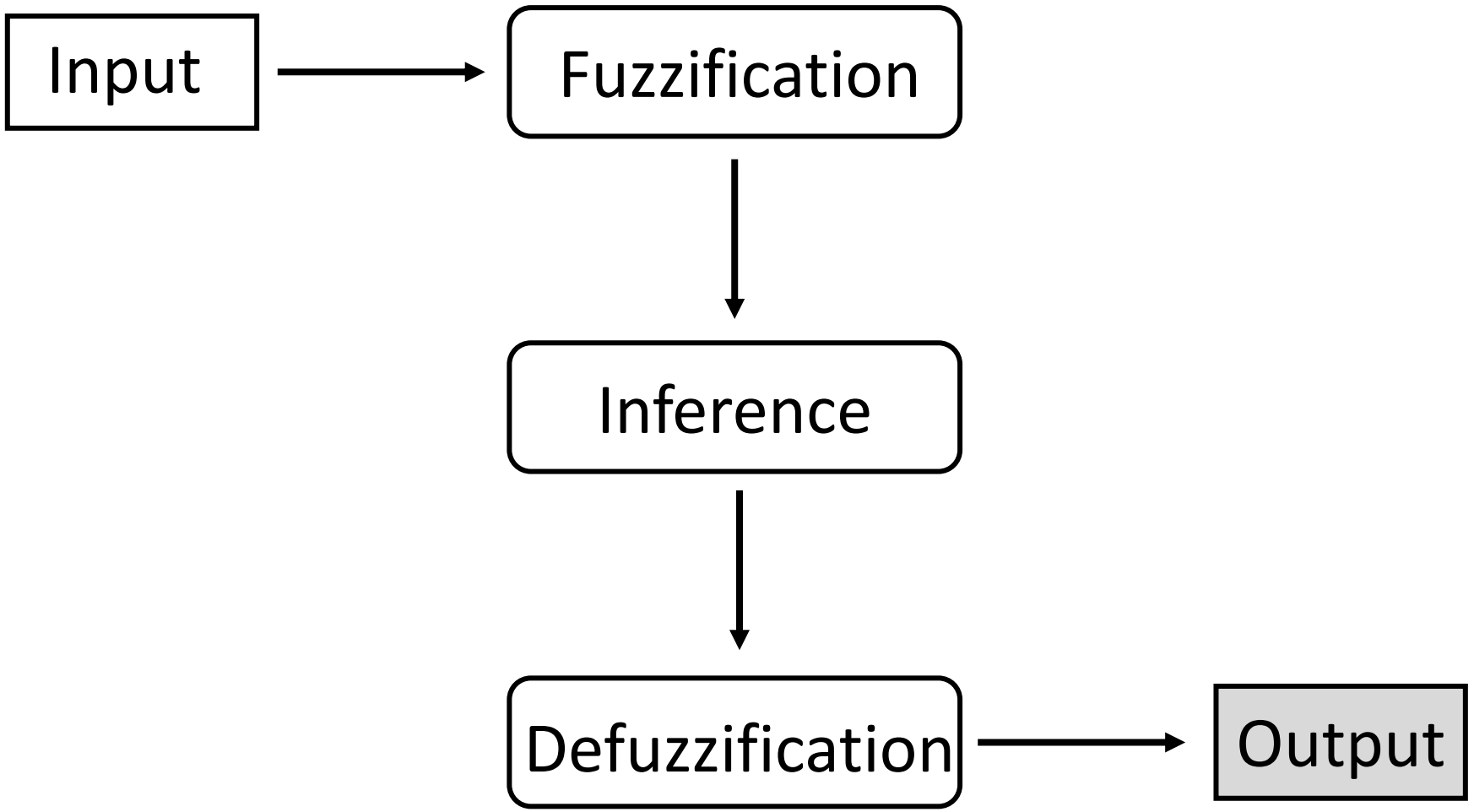
Error rate



2nd order Low pass filter

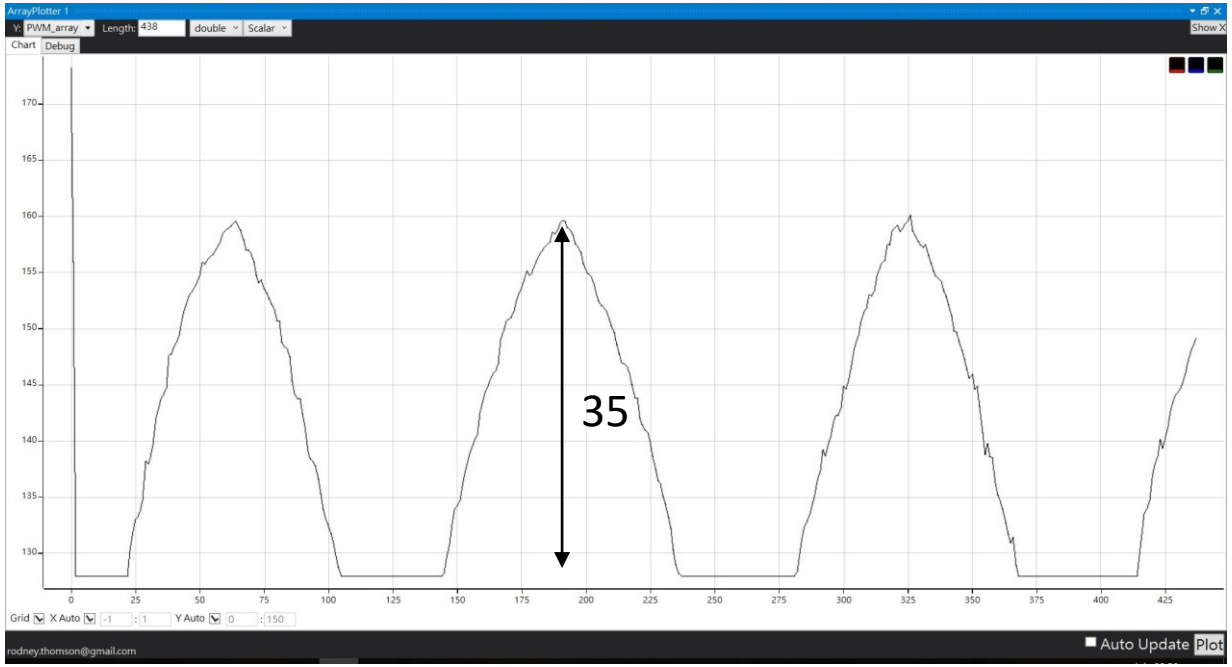


ER ⟶ Filter ⟶ $u(k) = \frac{\omega^2}{\omega^2+2\omega+1} \times ER + \frac{2\omega+2}{\omega^2+2\omega+1} \times u(k-1) - \frac{1}{\omega^2+2\omega+1} \times u(k-2)$

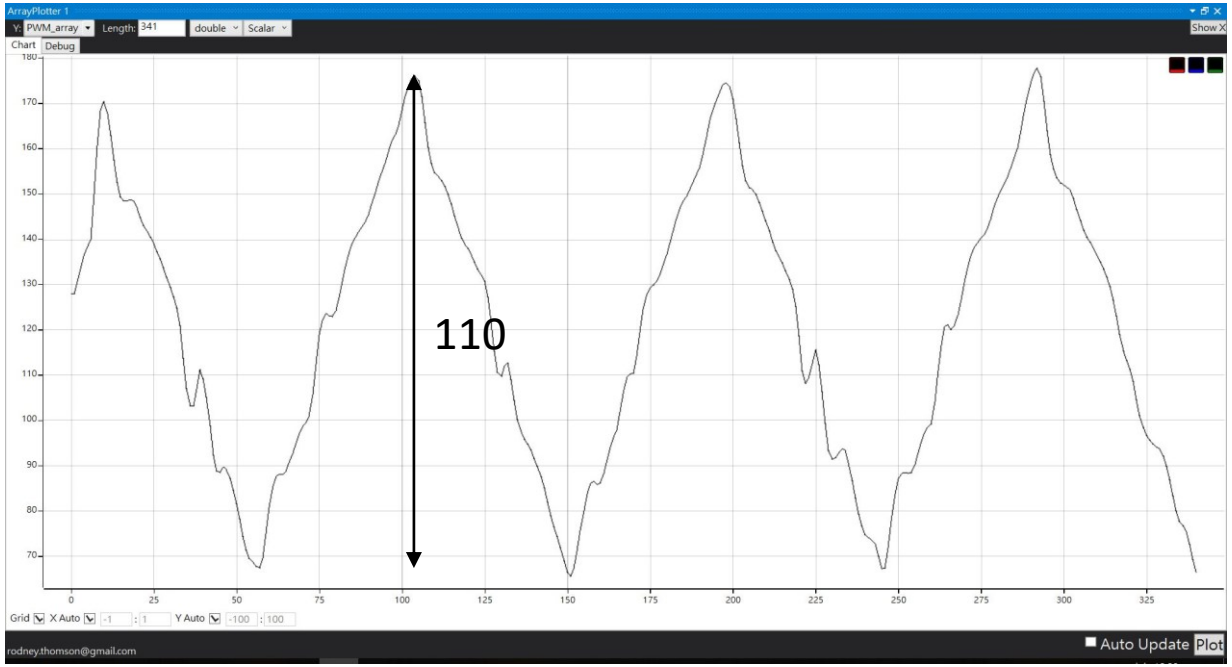

Input
Fuzzification
Inference
Defuzzification
Output

Controller Output PWM

Inference based on intuition



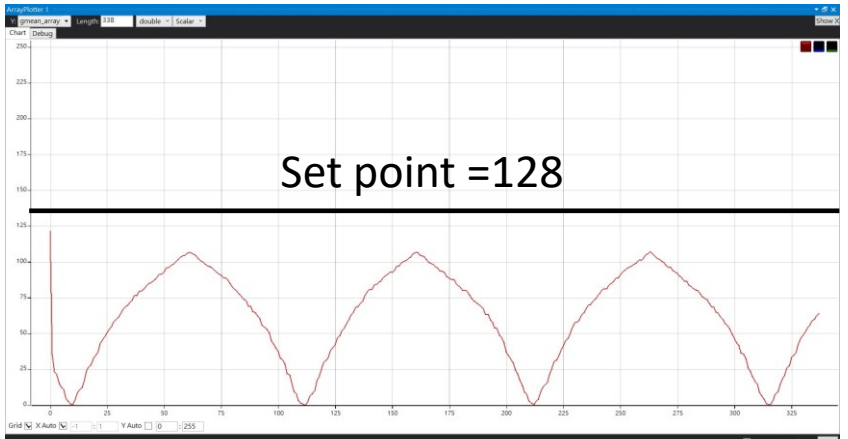
PD like fuzzy Inference



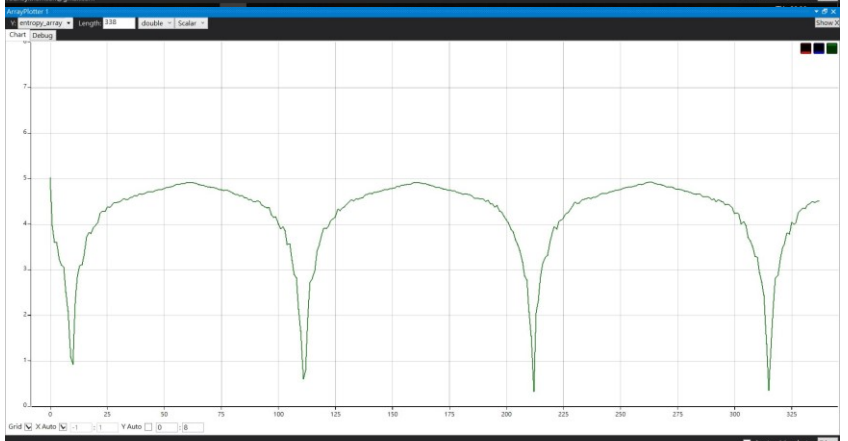
Result

Without fuzzy controller

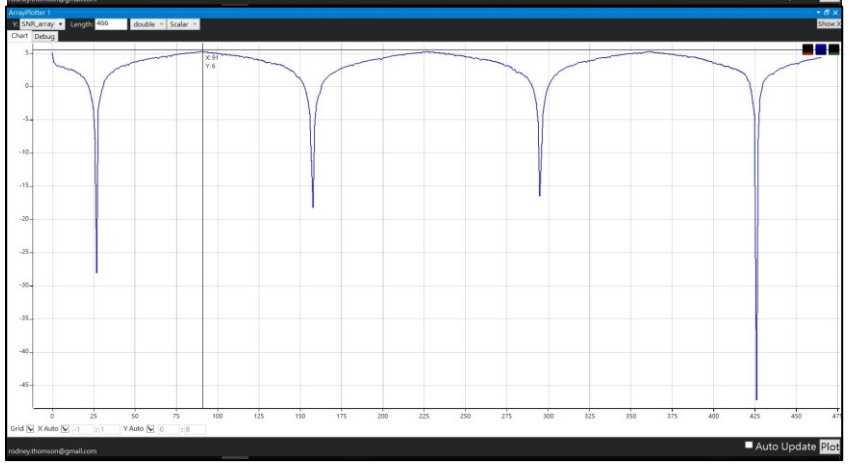
Mean gray value



Entropy

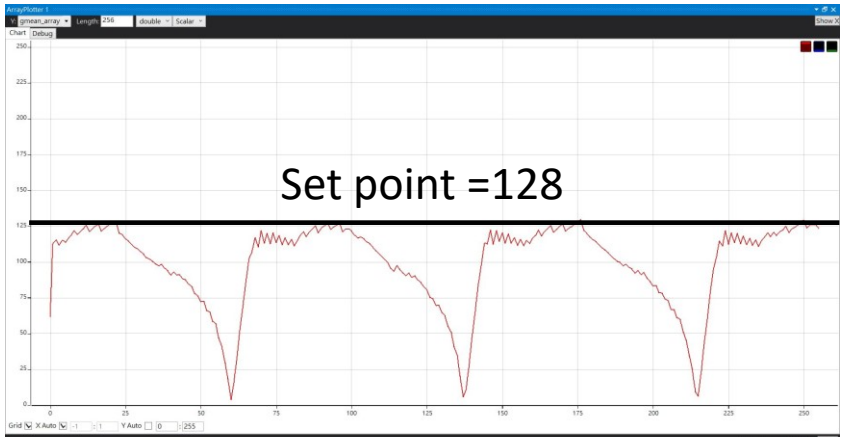


SNR

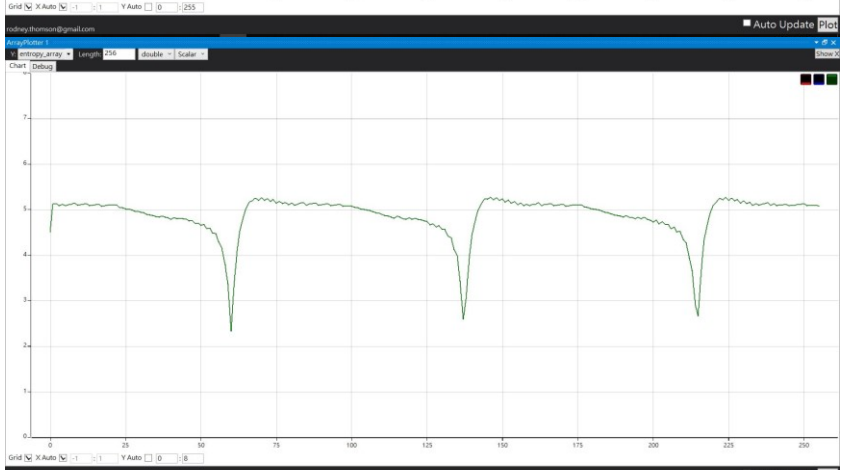


Auto Exposure

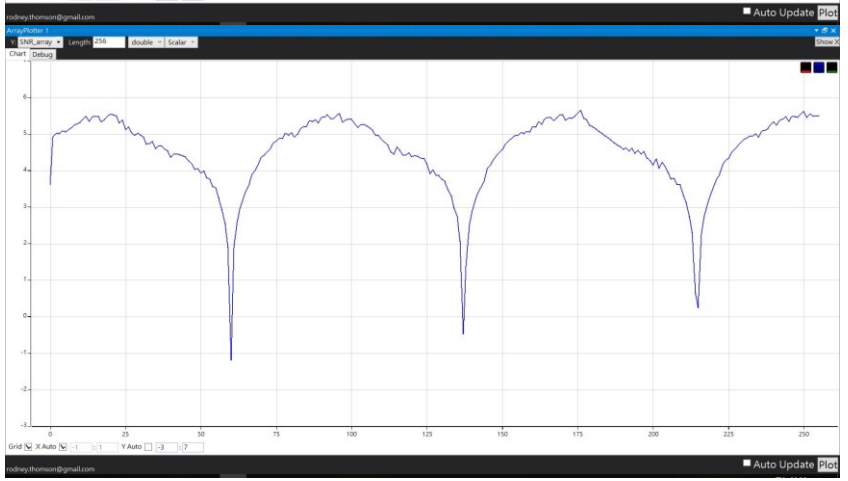
Mean gray value



Entropy

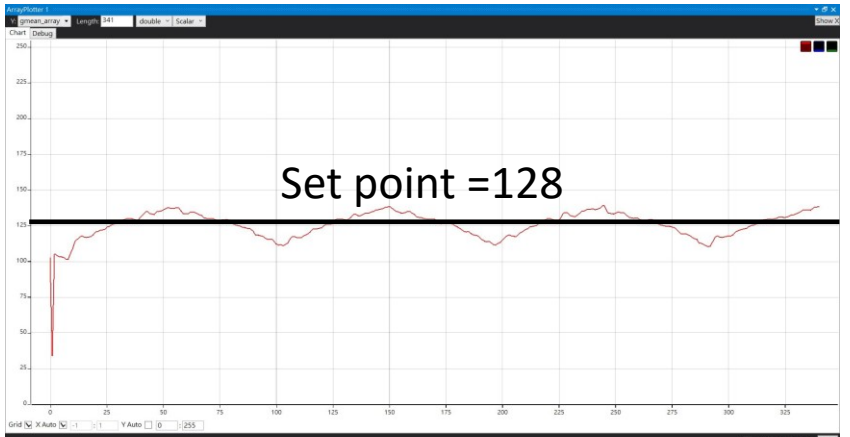


SNR

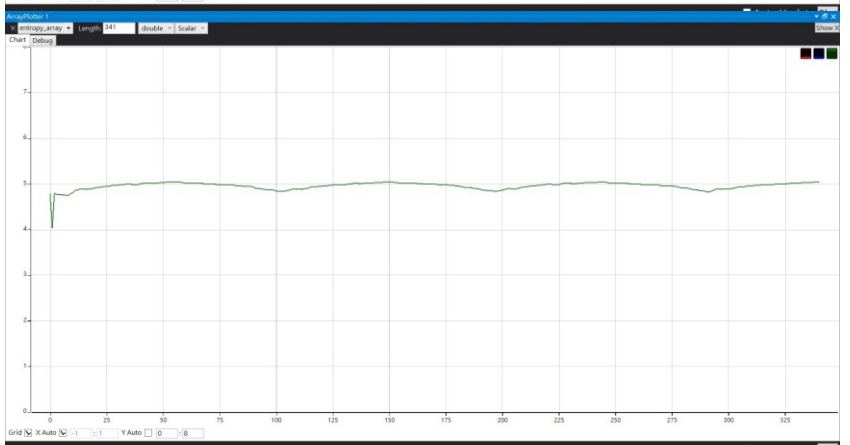


Fuzzy controller

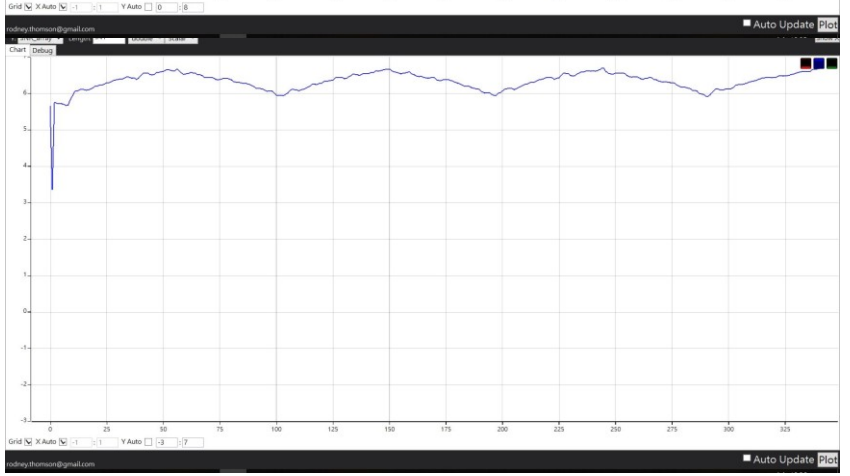
Mean gray value



Entropy



SNR



Conclusion

	Proposed Method	AE
Fps	≅18	≅12
Error	-10 to 16	0 to 128

→ Proposed Method is better than AE in low light situation.