Introduction

TOF

- d-TOF
  - Pulsed modulation
  - time of flight를 direct로 계산하는 방법
  - high energy의 pulse 신호를 쏴주기 때문에 background illumination의 영향을 줄일 수 있음
  - 상대적으로 Optical power의 평균값은 작기 때문에 eye safety에 중요한 요소가 될 수 있음
- i-TOF
  - Continuous wave (CW) modulation

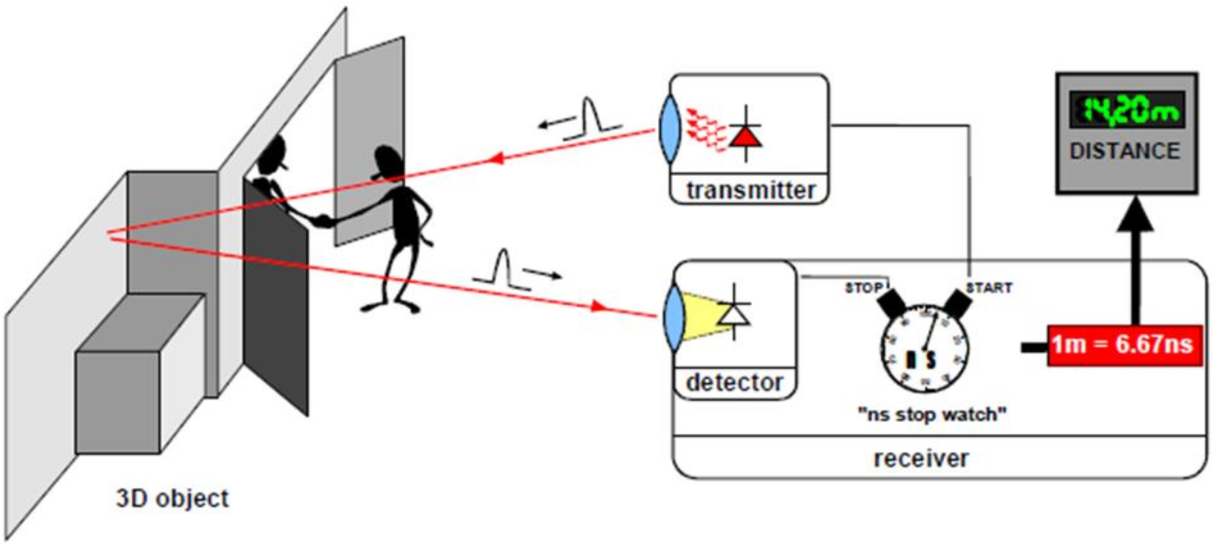


Figure 1.3 *Basic principle of an (optical) TOF ranging system.*

I-TOF

- Depth estimation

$$d = \frac{1}{2}ct$$

$d:distance$
$c:speed\ of\ light$
$t:time\ of\ flight$

- Relationship between phase shift and time of flight

$$\emptyset = 2\pi ft$$

$\emptyset:phase\ shift$
$f:moduation\ frequency$
$t:time\ of\ flight$

time of flight를 직접적으로 구하기 어려움
→ phase shift를 이용하여 t를 계산하는 것이 목표.

$$d = \frac{1}{2f}c\ \frac{\emptyset}{2\pi}$$

```
UR=calib.speed_of_light./(2*Fmod);

% rescale from rad to m and store in Distance
Distance(iFmod,ifiber) = Phase * UR(iFmod)/(2*pi);
```

$$d_{amb} = \frac{1}{2f}c$$

ambiguity distance:
해당 modulation으로 측정할 수 있는 최대 거리

80MHz: 1.8657m
60MHz: 2.4876m

LG이노텍

4-Bucket Method

◆ Cross correlation Function

- Emitted signal

$$s(t) = \cos(2\pi f t)$$

- Received signal

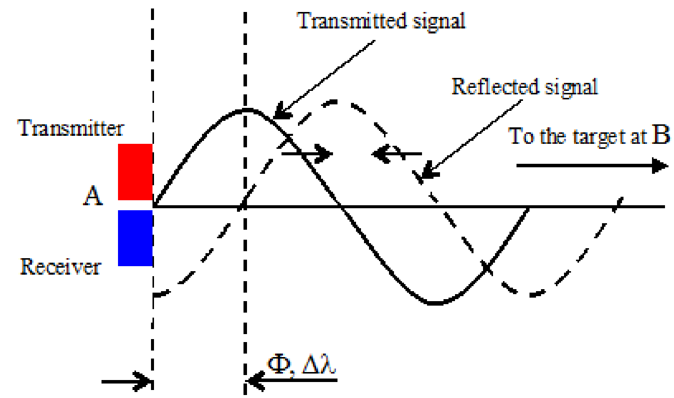
$$r(t) = A\cos\big(2\pi f(t-\tau)\big) + B$$

- Cross-correlation between emitted and received signals

$$C(x) = \lim_{T\to\infty} \frac{1}{T}\int_{-T/2}^{T/2} r(t)s(t+x)dt$$

$$= \frac{A}{2}\cos(2\pi f\tau + 2\pi f x) + B$$

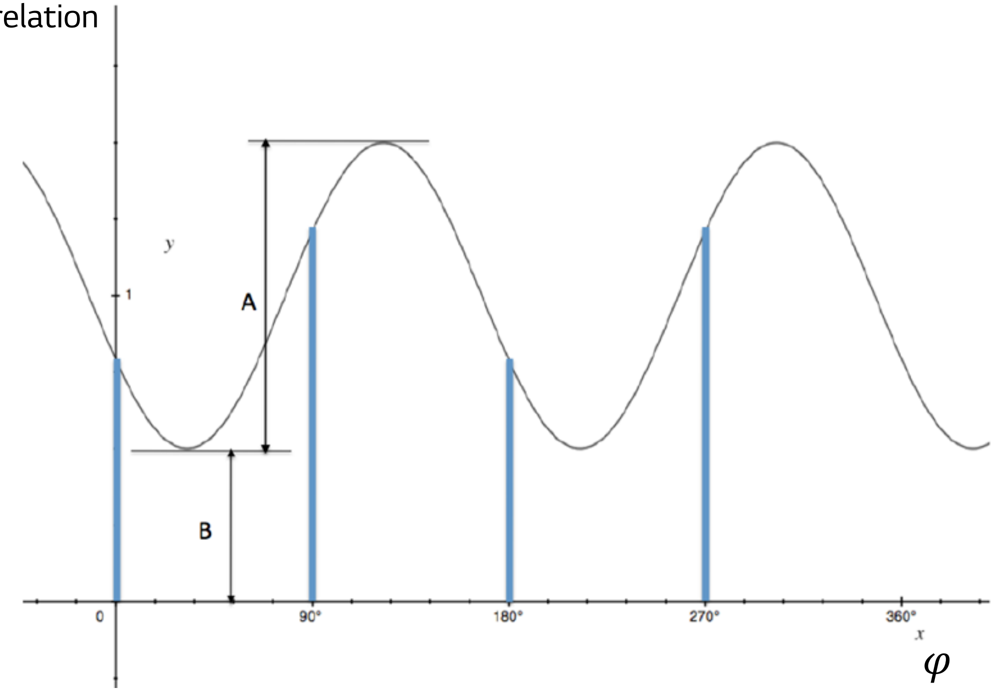
$$= \frac{A}{2}\cos(\emptyset + \varphi) + B$$



4-Bucket Method

- 4-Bucket Method

C(x): correlation



A: reflected amplitude
→ function of optical power
B: offset
→ ambient light, residual system offset

$$C(x_0) = \frac{A}{2}\cos(\emptyset + 0) + B = \frac{A}{2}\cos(\emptyset) + B$$

$$C(x_1) = \frac{A}{2}\cos\left(\emptyset + \frac{\pi}{2}\right) + B = -\frac{A}{2}\sin(\emptyset) + B$$

$$C(x_2) = \frac{A}{2}\cos(\emptyset + \pi) + B = -\frac{A}{2}\cos(\emptyset) + B$$

$$C(x_3) = \frac{A}{2}\cos\left(\emptyset + \frac{3\pi}{2}\right) + B = \frac{A}{2}\sin(\emptyset) + B$$

$$\emptyset = 2\pi f\tau = \arctan\left(\frac{C(x_3) - C(x_1)}{C(x_0) - C(x_2)}\right)$$

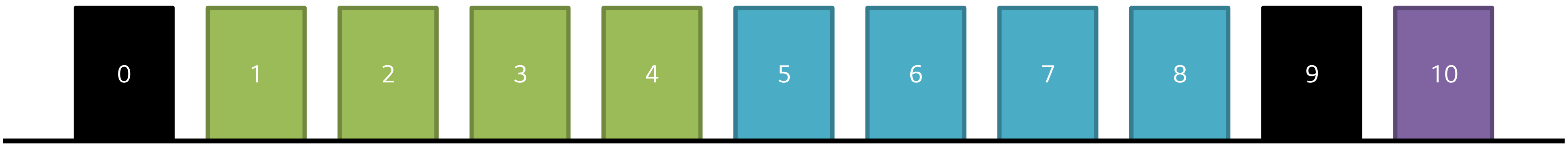
$$\mathrm{A} = \frac{1}{2}\sqrt{(C(x_3) - C(x_1))^2 + (C(x_0) - C(x_2))^2}$$

$$\mathrm{B} = \frac{1}{4}(C(x_0) + C(x_1) + C(x_2) + C(x_3))$$

PMD depth calculation method

◆ PMD Raw data for calibration

[super frame]



Sequence	0	1				2				3	4
Raw image	Gray Scale	0˚	90˚	180˚	270˚	0˚	90˚	180˚	270˚	Gray Scale	Gray Scale
Modulation Frequency [MHz]	60.24	80.32				60.24					
Duty Cycle (Illumination)	0% (OFF)	25%				25%				0% (OFF)	25%

PMD depth calculation method

◆ RAW data → Phase

modulation 별 각각의 phase 계산

```
function [Phase, Amplitude] = ProcessNPhase (Raw, Phaseshift)
N_Raw=numel(Raw);
for i=1:1:N_Raw
    if ~isfloat(Raw{i})
        Raw{i}=double(Raw{i});
    end
end

% initialize frames for real and imaginary parts
Re=zeros(size(Raw{1}));
Im=Re;

% compile real and imaginary parts
for i=1:1:N_Raw
    Re = Re - 2/N_Raw*Raw{i}*cosd(Phaseshift(i));
    Im = Im + 2/N_Raw*Raw{i}*sind(Phaseshift(i));
end

% calculate phase and amplitude values
Phase = pi + atan2(Im, Re);
Amplitude = sqrt(Im.^2 + Re.^2);
end
```

Re = 1/2 * (Raw(180°) – Raw(0°))

Im = 1/2 * (Raw(90°) – Raw(270°))

tangent가 주기가 pi라 값은 같은 듯 한데 왜 pi를 더할까?

arctan2의 범위 (-pi, +pi) ➔ (0, 2pi)

$$\emptyset = \pi + \arctan\left(\frac{Raw(x_1) - Raw(x_3)}{Raw(x_2) - Raw(x_0)}\right)$$

$$A = \frac{1}{2}\sqrt{(Raw(x_1) - Raw(x_3))^2 + (Raw(x_2) - Raw(x_0))^2}$$

LG이노텍

PMD depth calculation method

◆ Phase → Distance

```
for ifiber = 1:N_fiber
    if ~obj.FB.valid_fibers(ifiber)
        % skip invalid fibers
        continue;
    end

    % create a logical mask for this fiber (for FPPN values)
    Mask=false(obj.FB.FRData.SensorROI([4,3]));
    Mask(obj.FB.FRData.FPixel{1,ifiber},obj.FB.FRData.FPixel{2,ifiber})=true;

    % iterate over all modulation frequencies
    for iFmod=1:N_Fmod
        BPhase=obj.FB.FRData.Phase{iFmod,ifiber};

        % apply wiggling compensation
        BPhase = BPhase + harmonics.apply(obj.PhaseWiggling{iFmod}, BPhase);

        % apply temperature drift compensation
        Delta_Temp = obj.FB.FRData.illuminationTemperature - obj.TempCompensation{iFmod}(1);
        Delta_Distance = obj.TempCompensation{iFmod}(2) * Delta_Temp ...
            + obj.TempCompensation{iFmod}(3) * Delta_Temp.^2 ...
            + obj.TempCompensation{iFmod}(4) * Delta_Temp.^3; % in m
        Delta_Phase = Delta_Distance*(2*pi)/UR(iFmod); % in rad
        BPhase = BPhase - Delta_Phase; % in rad

        % apply FPPN
        BPhase = BPhase + obj.FPPN{iFmod}(Mask)*(2*pi)/UR(iFmod);

        % average over pixels in blob;
        Phase = mean(fix_unambiguous_range(BPhase));

        % rescale from rad to m and store in Distance
        Distance(iFmod,ifiber) = Phase * UR(iFmod)/(2*pi);
    end
end

FiberLengths = CALC_2Freq(Distance(1,:),Distance(2,:),Fmod);
```

modulation 별 각각의 distance 계산

Distance 계산
1) phase wiggling compensation
2) temperature drift compensation
3) FPPN compensation
4) rad to m

$$d = \frac{1}{2f} c \frac{\emptyset}{2\pi}$$

두 modulation에서 나온 거리를 이용하여 최종 거리 계산

PMD depth calculation method

◆ CALC_2Freq

```
function D=CALC_2Freq(d1,d2,Fmod)
flong=lcm(Fmod(1),Fmod(2));
UR=flong./[Fmod(1),Fmod(2)];
[~,u,~]=gcd(UR(1),UR(2));
fak=flong*2/calib.speed_of_light;

d_diff=fak*(d2-d1);
f=round(d_diff);
D=f*u*UR(1)+fak*d1+(d_diff-f)/2; %주기 d1기준 위상차
D=mod(D,prod(UR))/fak;
end
```

80MHz와 60MHz의 최소공배수 = 240960000 = 240MHz

UR = [3, 4]

u = -1 / gcd = 최대 공약수 - [g, c, d] = GCD[A, B] → g = A*c+B*d

$$d = \frac{1}{2f} c \frac{\emptyset}{2\pi}$$

$$d_{diff} = \frac{2f_{240}}{c} \frac{c}{2f_{60}} \frac{\emptyset_{60}}{2\pi} - \frac{2f_{240}}{c} \frac{c}{2f_{80}} \frac{\emptyset_{80}}{2\pi}$$

$$= 4\frac{\emptyset_{60}}{2\pi} - 3\frac{\emptyset_{80}}{2\pi}$$

$$d = \frac{c}{2f_{60}} \frac{\emptyset_{60}}{2\pi} = \frac{c}{2f_{240}} \frac{4\emptyset_{60}}{2\pi}$$

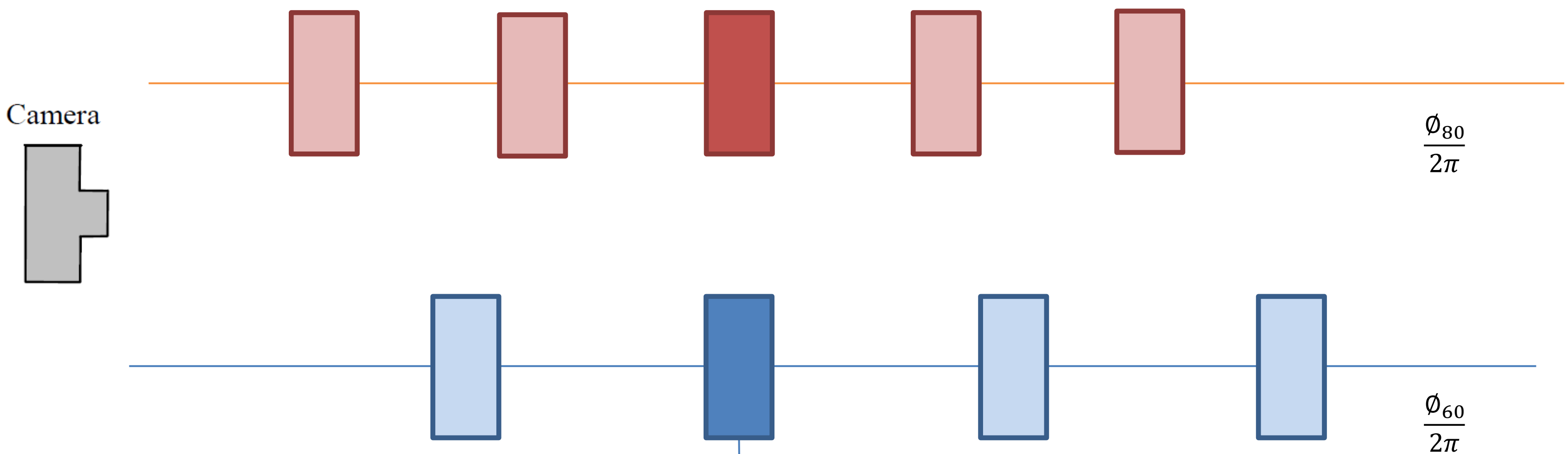
$$d = \frac{c}{2f_{80}} \frac{\emptyset_{80}}{2\pi} = \frac{c}{2f_{240}} \frac{3\emptyset_{80}}{2\pi}$$

PMD depth calculation method

- CALC_2Freq

기본 concept: 두 주파수(f_A, f_b)에서 나올 수 있는 가능 거리가 동일한 거리 찾기

$\frac{c}{2f_A}\left(\frac{\phi_A}{2\pi}+0\right)$ $\frac{c}{2f_A}\left(\frac{\phi_A}{2\pi}+1\right)$ $\frac{c}{2f_A}\left(\frac{\phi_A}{2\pi}+2\right)$ $\frac{c}{2f_A}\left(\frac{\phi_A}{2\pi}+3\right)$ $\frac{c}{2f_A}\left(\frac{\phi_A}{2\pi}+4\right)$

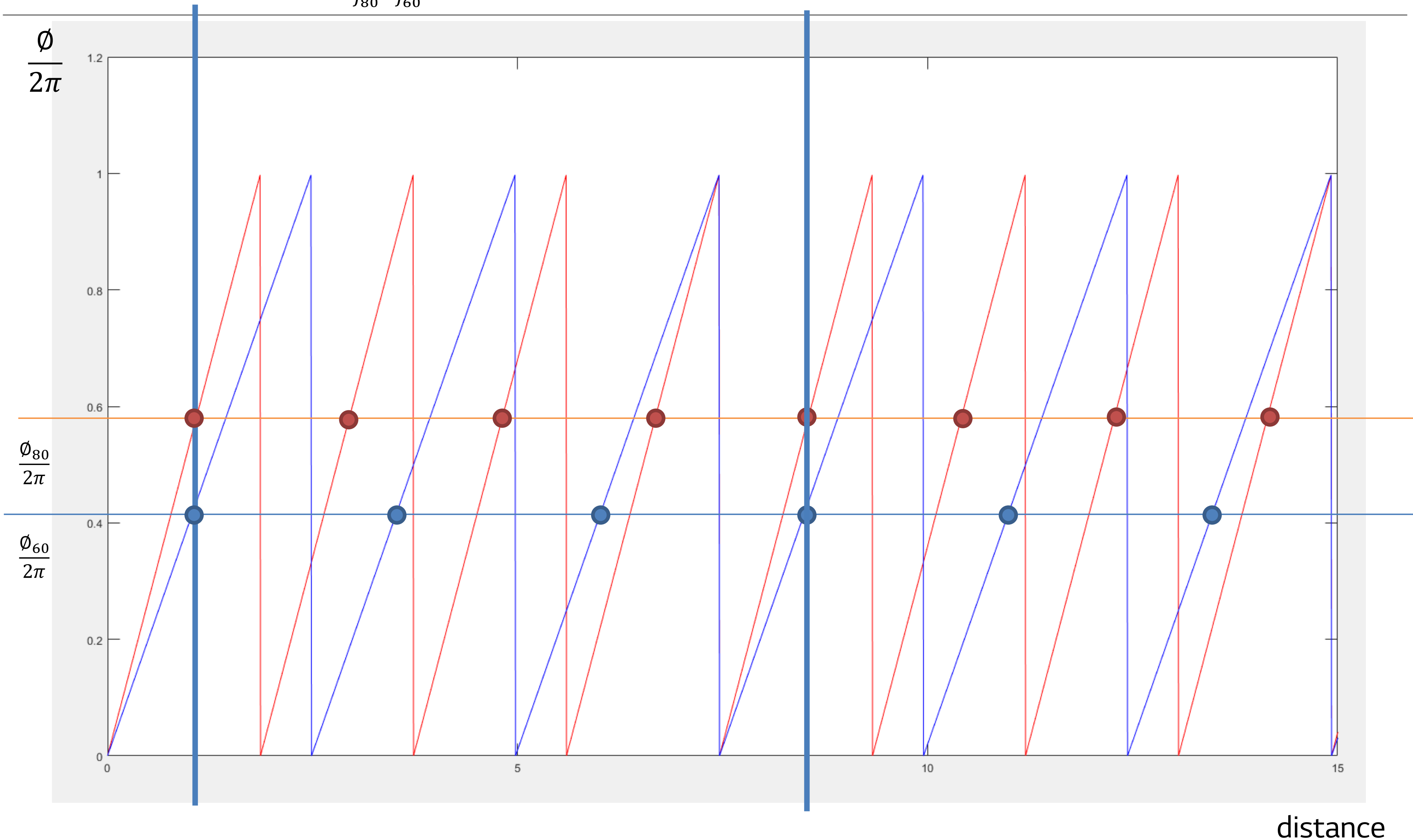


$\frac{c}{2f_B}\left(\frac{\phi_B}{2\pi}+0\right)$ $\frac{c}{2f_B}\left(\frac{\phi_B}{2\pi}+1\right)$ $\frac{c}{2f_B}\left(\frac{\phi_B}{2\pi}+2\right)$ $\frac{c}{2f_B}\left(\frac{\phi_B}{2\pi}+3\right)$

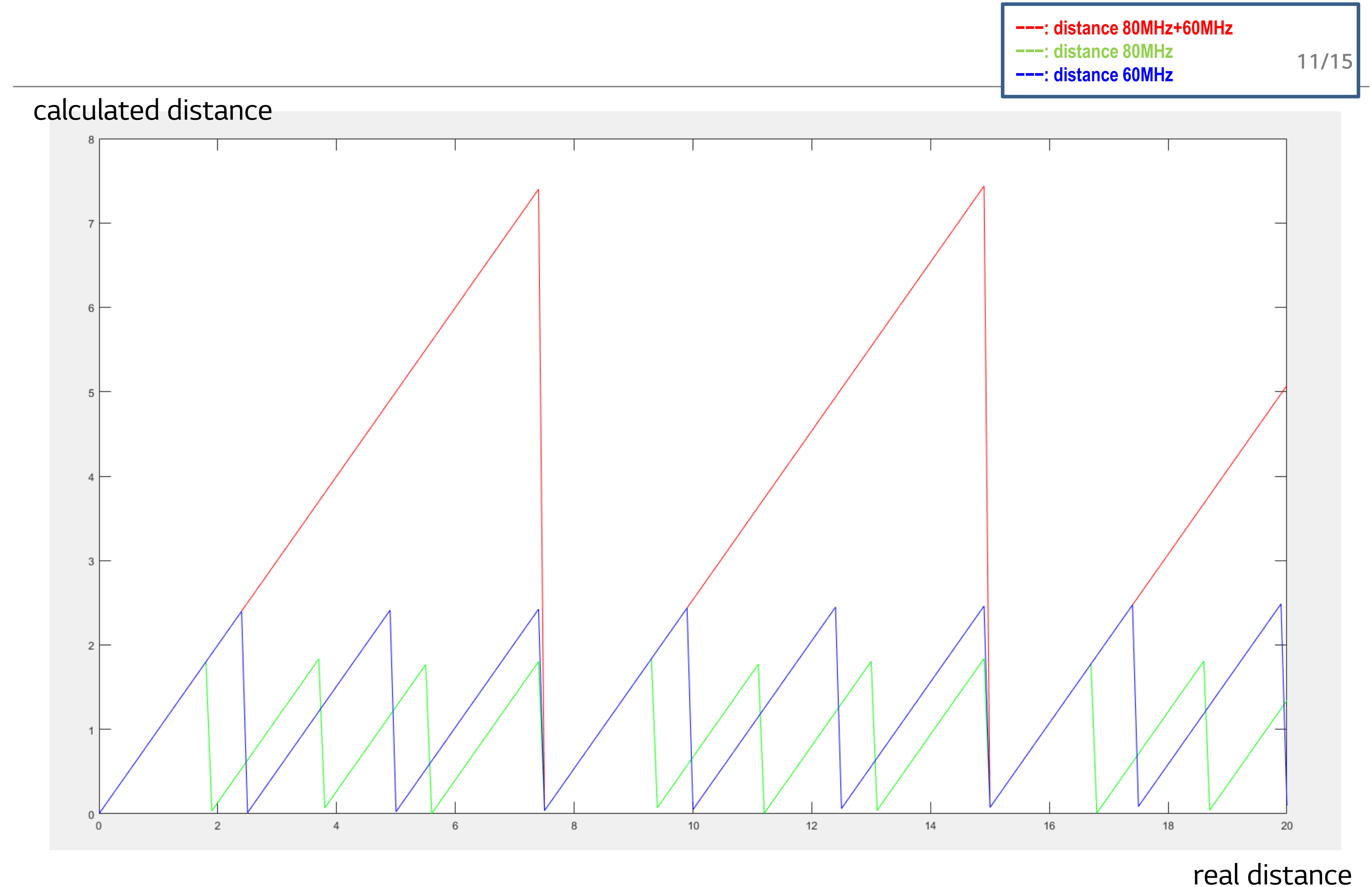
목표: 거리가 같게 나오는 n_A, n_B 구하기

$$d = \frac{c}{2f_A}\left(\frac{\phi_A}{2\pi}+n_A\right) = \frac{c}{2f_B}\left(\frac{\phi_B}{2\pi}+n_B\right)$$

f_{80}: f_{60} → $\frac{1}{f_{80}}:\frac{1}{f_{60}}$→4:3→둘이 만나는 주기 12



LG이노텍


---: distance 80MHz+60MHz
---: distance 80MHz
---: distance 60MHz
calculated distance
8
7
6
5
4
3
2
1
0
0
2
4
6
8
10
12
14
16
18
20
real distance

PMD depth calculation method

- CALC_2Freq

$$d = \frac{c}{2f_A}\left(\frac{\emptyset_A}{2\pi} + n_A\right) = \frac{c}{2f_B}\left(\frac{\emptyset_B}{2\pi} + n_B\right)$$

목표: 거리가 같게 나오는 n_A, n_B 구하기

$$\rightarrow min\left|\frac{c}{2f_A}\left(\frac{\emptyset_A}{2\pi} + n_A\right) - \frac{c}{2f_B}\left(\frac{\emptyset_B}{2\pi} + n_B\right)\right|$$

거리가 같다 = 거리차 최소

80:60 → 4:3

$$\rightarrow min\left|\frac{cM_B}{2f_AM_B}\left(\frac{\emptyset_A}{2\pi} + n_A\right) - \frac{cM_A}{2f_BM_A}\left(\frac{\emptyset_B}{2\pi} + n_B\right)\right|$$

최소 공배수

$$M_A = \frac{f_A}{\gcd(f_A,f_B)},\ M_B = \frac{f_B}{\gcd(f_A,f_B)}$$

$$\rightarrow min\left|M_B\left(\frac{\emptyset_A}{2\pi} + n_A\right) - M_A\left(\frac{\emptyset_B}{2\pi} + n_B\right)\right|$$

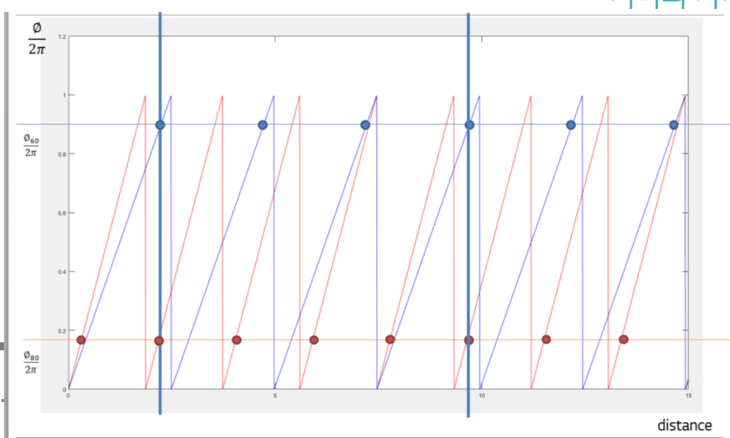
$$p_A = \frac{\emptyset_A}{2\pi},\ p_B = \frac{\emptyset_B}{2\pi}$$

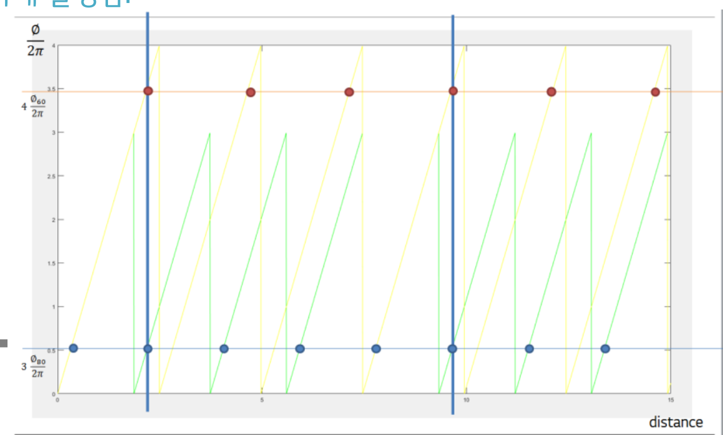
공통부분 없애기

$$\rightarrow min|M_B(p_A + n_A) - M_A(p_B + n_B)|$$

최소공배수로 phase를 맞추면
거리의 차가 일정하게 발생함.

$$d = \frac{c}{2f_{60}}\frac{\emptyset_{60}}{2\pi} = \frac{c}{2f_{240}}\frac{4\emptyset_{60}}{2\pi} \qquad d = \frac{c}{2f_{80}}\frac{\emptyset_{80}}{2\pi} = \frac{c}{2f_{240}}\frac{3\emptyset_{80}}{2\pi}$$





LG이노텍

PMD depth calculation method

- CALC_2Freq

$min|M_B(p_A + n_A) - M_A(p_B + n_B)|$

solution1)
최소가 되는 n_A, n_B 조합을 경우의수로 찾아보는 방법

One naive approach is to evaluate all possible combinations of n_A and n_B and select the options that give the difference closest to zero. With values established for n_A and n_B, distance is computed by

$$d = \frac{c}{2}\left[\frac{w(n_A + p_A)}{f_A} + \frac{(1-w)(n_B + p_B)}{f_B}\right] \quad (8)$$

where w is a weighting factor between 0 and 1 and is chosen to minimize the variance in the output distance estimate.

PMD depth calculation method

◆ CALC_2Freq

$$min|M_B(p_A + n_A) - M_A(p_B + n_B)|$$

solution2)

residue to binary converter을 이용한 방법

step1. Chinese remainder theorem을 이용한 n_B 계산

$$e = p_A M_B - p_B M_A$$ = d_diff

$$n_B = \text{mod}\,[k_0 round(e), M_B] \qquad (9)$$

마이너스 위상차 보상

step2. scale 전 거리 계산

$$X = wM_B(n_A + p_A) + (1-w)M_A(n_B + p_B)$$
$$= M_A n_B + M_A p_B + w\,(e - round(e))\,. \qquad (10)$$

UR(1)*f*u   fak*d1   1/2*(d_diff-f)

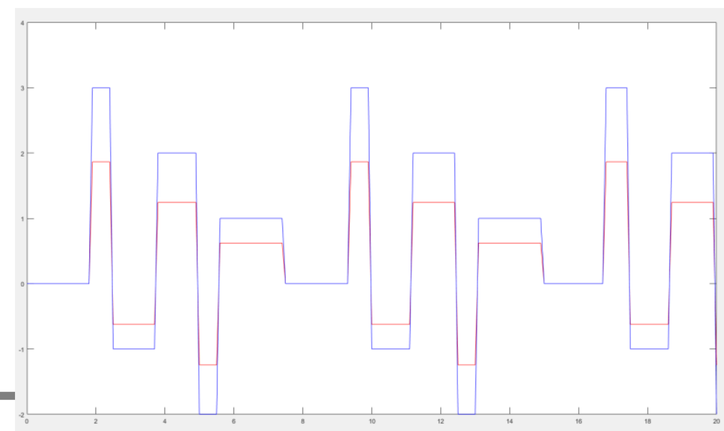
step3. scaling

$$d = d_E \cdot \frac{X}{M_A M_B}.$$ D/fak

```
function D=CALC_2Freq(d1,d2,Fmod)
flong=lcm(Fmod(1),Fmod(2));
UR=flong./[Fmod(1),Fmod(2)];
[~,u,~]=gcd(UR(1),UR(2));
fak=flong*2/calib.speed_of_light;

d_diff=fak*(d2-d1);
f=round(d_diff);
D=f*u*UR(1)+fak*d1+(d_diff-f)/2; %주기 d1기준 위상차
D=mod(D,prod(UR))/fak;
end
```

마이너스 위상차 보상



LG이노텍

PMD depth calculation method

- CALC_2Freq

$$min|M_B(p_A + n_A) - M_A(p_B + n_B)|$$

solution2)

residue to binary converter을 이용한 방법

step1. Chinese remainder theorem을 이용한 n_B 계산

$$e = p_A M_B - p_B M_A \qquad = \text{d\_diff} = 3\,\frac{\emptyset_{80}}{2\pi} - 4\,\frac{\emptyset_{60}}{2\pi}$$

$$n_B = \text{mod}\,[k_0 round(e), M_B] \qquad (9)$$

0, 1, 2 → 반복되는 주파수를 제거하고서 몇 번째 주기인지 찾는 부분

step2. scale 전 거리 계산

$$X = wM_B(n_A + p_A) + (1 - w)M_A(n_B + p_B)$$

$$= M_A n_B + M_A p_B + w\,(e - round(e)). \qquad (10)$$

step3. scaling

$$d = \frac{c}{2f_B}\left(\frac{\emptyset_B}{2\pi} + n_B\right) = \frac{c}{2f_B M_A}\left(\frac{\emptyset_B M_A}{2\pi} + n_B M_A\right)$$

$$d = d_E \cdot \frac{X}{M_A M_B}.$$

D/fak

```
function D=CALC_2Freq(d1,d2,Fmod)
flong=lcm(Fmod(1),Fmod(2));
UR=flong./[Fmod(1),Fmod(2)];
[~,u,~]=gcd(UR(1),UR(2));
fak=flong*2/calib.speed_of_light;

d_diff=fak*(d2-d1);
f=round(d_diff);
D=f*u*UR(1)+fak*d1+(d_diff-f)/2; %주기 d1기준 위상차
D=mod(D,prod(UR))/fak;
end
```

d_ diff: 0→3→-1→2→-2→1→0

mod(d_diff, 4): 0→3→3→2→2→1→0

k0*d_diff: 0→-3→1→-2→2→-1→0

mod(k0d_diff, 4): 0→1→1→2→2→3→0

마이너스 위상차 보상