**Q1. INITIALS AND COLOR TRANSFORMATION**

**Code**

|  |
| --- |
| % Q1. INITIALS  video = VideoReader('2\\face.mp4');  frames\_rgb = {};  while hasFrame(video)  frames\_rgb{end+1} = readFrame(video);  end  framecount = size(frames\_rgb, 2);  % convert frames to double-precision and range [0, 1]  frames\_rgb = cellfun(@(x) double(x) ./ 256.0, frames\_rgb, 'UniformOutput', false);  % convert frames to yiq color space  frames\_yiq = cellfun(@(x) rgb2ntsc(x), frames\_rgb, 'UniformOutput', false); |

**Result (face.mp4)**

|  |
| --- |
| framecount = 301  frames\_yiq = {1 x 301 cell} (592x528x3 double) |

**Description**

VieeoReader로 video 파일을 열고 한 프레임씩 읽어 frames\_rgb에 append한다.

cellfun 함수는 모든 cell에 동일한 함수를 적용한다.

**Q2. LAPLACIAN PYRAMID**

**Code**

|  |
| --- |
| pyramid = cellfun(@LaplacianPyramid, frames\_yiq, 'UniformOutput', false);  % pyramid : {1 x frame cell} {1 x level cell} (...)  pyramid = cat(1,pyramid{:});  % pyramid : {frame x level cell} (...)  pyramid = pyramid';  % pyramid : {level x frame cell} (...)  levelcount = size(pyramid, 1);  function ret = LaplacianPyramid(x)  ret = {};  [width, height, c] = size(x);  while((width > 1) || (height > 1))  hx = imgaussfilt(x, 2); % Gaussian Filtering  hx = hx(1:2:end, 1:2:end, :); % SubSampling  % Upsampling (To make differance image)  gx = imresize(hx, 2, 'nearest');  % Take Differance and add to return cell  ret{end+1} = x - gx(1:width, 1:height);  x = hx;  [width, height, c] = size(x);  end  % Finally add 1x1 image to return cell  ret{end+1} = x;  end |

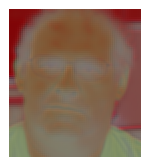
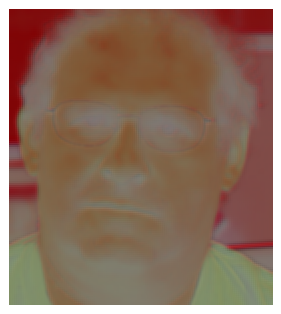
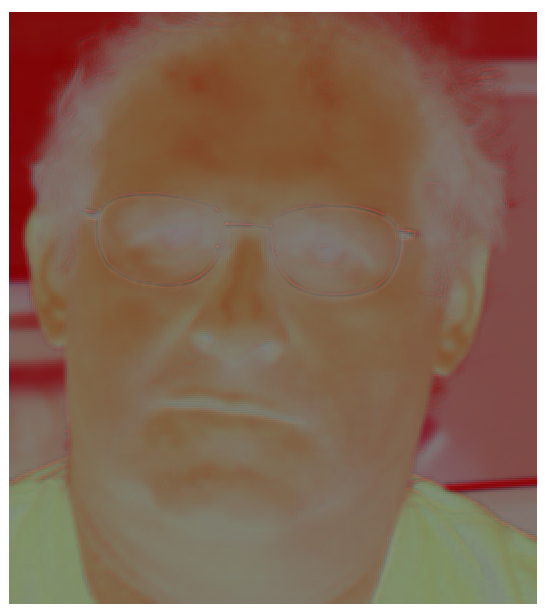
**Description**

우선 한 프레임의 라플라시안 피라미드를 반환하는 함수 LaplacianPyramid를 정의하였다.

cellfun을 사용하여 각 프레임에 함수를 적용한 후 cat과 transpose(‘)를 사용하여 다음 과정에서 사용하기 좋은 형태로 변경하였다.

**Result (face.mp4)**

Level 6까지의 라플라시안 피라미드



**Q3. TEMPORAL FILTERING**

**Code**

|  |
| --- |
| timeEachLevel = {};  for i = 1:levelcount  timeEachLevel{end+1} = cat(4, pyramid{i,:});  end  % timeEachLevel : {1 x level cell} ( width x height x 3 x frames )    for i = 1:levelcount  freqEachLevel = cellfun(@(x) fft(x, framecount, 4), timeEachLevel, ...  'UniformOutput', false);  end  freqEachLevel\_result = {};  band\_a1 = 9; % left boundary of passing band  band\_b1 = 11; % right boundary of passing band  band\_a2 = framecount - band\_b1 + 2;  band\_b2 = framecount - band\_a1 + 2;  amplifier = 100.0;      for i = 1:levelcount  % Copy frequency domain for save the original images  freqEachLevel\_result{end + 1} = freqEachLevel{i}(:,:,:,:);  % bandpass filtering and amplifying  freqEachLevel\_result{end}(:,:,:,band\_a1:band\_b1) = ...  freqEachLevel\_result{end}(:,:,:,band\_a1:band\_b1) .\* amplifier;  freqEachLevel\_result{end}(:,:,:,band\_a2:band\_b2) = ...  freqEachLevel\_result{end}(:,:,:,band\_a2:band\_b2) .\* amplifier;  end |

**Description**

앞에서 만든 라플라시안 피라미드를 레벨별로 (width x height x 3 x frame)의 4-d array로 형변환한 후 fft를 수행하였다.

이후 frequency domain의 이미지에서 원하는 band만을 원하는 배수로 증폭시킨다

반대쪽 밴드의 bound가 framecount – a + 2가 되는 이유는 fft 결과의 1번 index는 DC component이기에 2번 index와 마지막 index가 대응하기 때문이다.

**Q4. EXTRACTING THE FREQUENCY BAND OF INTEREST**

**Code**

|  |
| --- |
| plotdata = [];  for i = 1:framecount  m = 0;  for j = 1:levelcount  cell = freqEachLevel{j}(:, :, :, i);  m = m + mean(abs(cell(:)));  end  plotdata = [plotdata; m];  end  plot(plotdata);  ylim([0, 1]) |

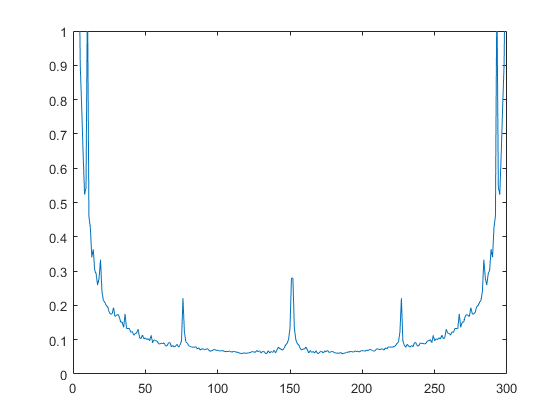
**Description**

video 전체에서의 각 frequency의 평균 magnitude를 구하여 그래프로 그려보았다.

Discrete FFT의 결과에서 n번째 index의 주기는 N(전체 길이) / n 이다. 사람의 심장 박동은 대략 분당 70회이고, 영상들은 30frame/s이므로, 혈류를 나타내는 index는 대략 (전체길이) / (24-30) 일 것을 예상할 수 있다.

**Result**

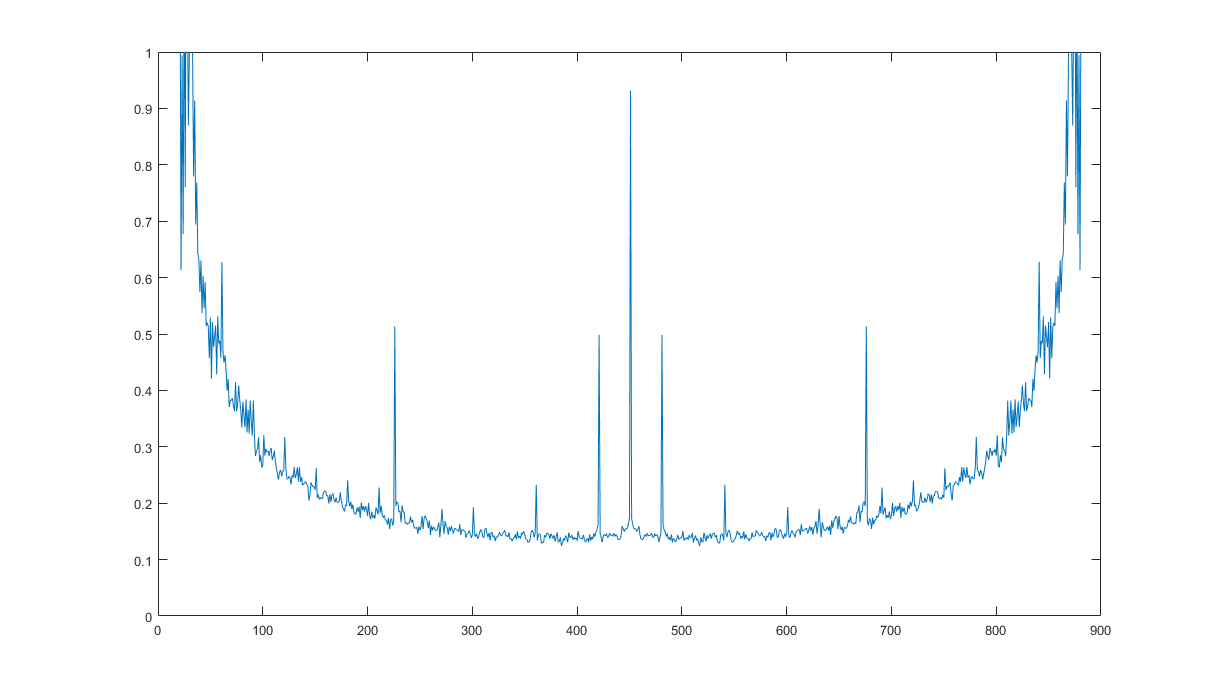
face.mp4의 frequency별 평균 magnitude



Matlab Index 기준으로 10 (293), 28 (275), 76 (227)이 특이하게 높음을 확인할 수 있었다.

이중 10 (293)이 혈류량을 나타내는 frequency일 것으로 추정하였다.

baby2.mp4의 frequency별 평균 magnitude



Matlab Index 기준으로 27-32 (870-875), 61 (841), 74 (828), 226 (676)이 특이하게 높음을 확인할 수 있었다.

아기의 심장 박동은 성인보다 빠른 분당 120회 정도이므로 61 (841)이 혈류량을 나타내는 frequency일 것으로 추정하였다.

**Q5. IMAGE RECONSTRUCTION**

**Code**

|  |
| --- |
| for i = 1:levelcount  timeEachLevel\_result = cellfun(@(x) ifft(x, framecount, 4), freqEachLevel\_result, ...  'UniformOutput', false);  end    respyramid\_tmp = cellfun(@(x) num2cell(x, [1 2 3]), timeEachLevel\_result, ...  'UniformOutput', false);  % respyramid\_tmp : (1 x level cell} {1 x 1 x 1 x frame cell} (...)  respyramid = {};  for i = 1:framecount  ret = {};    for j = 1:levelcount  ret{end+1} = respyramid\_tmp{j}{1,1,1,i};  end  % ret : {1 x level cell} (...)  respyramid{end+1} = ret;  end  % resppyraid : {1 x frame cell} {1 x level cell} (...)    resframes\_yiq = cellfun(@InverseLaplacianPyramid, respyramid, 'UniformOutput', false);  resframes\_rgb = cellfun(@ntsc2rgb, resframes\_yiq, 'UniformOutput', false);  resvideo = VideoWriter('2\\face\_res\_8\_12\_x100.mp4', 'MPEG-4');  open(resvideo)  for i = 1:framecount  writeVideo(resvideo, resframes\_rgb{i})  end  close(resvideo)  function ret = InverseLaplacianPyramid(x)  % use real part only  gx = real(x{size(x, 2)});  for i = size(x, 2)-1:-1:1  % UpSample  gx = imresize(gx, 2, 'nearest');  [width, height, c] = size(x{i});  gx = gx(1:width, 1:height);  % Add Difference Image  gx = gx + real(x{i});  end  ret = gx;  end |

**Description**

우선 각 레벨별로 ifft를 적용하여 time domain으로 다시 돌린다.

이후 Laplacian Pyramid를 다시 이미지로 복원해야 하는데 여기에 frame별로 cellfun을 적용하기 위해, num2cell 함수로 우선 각각의 frame을 cell로 만든다

그 후 frame이 안에 각 level별 cell이 들어있는 형태로 만들고 InverseLaplacianPyramid를 적용시킨다.

yiq 이미지를 다시 rgb로 바꾼후 결과 영상을 저장하면 된다.

**Result**

**face.mp4**

Index 8-12 (0.7-1.1 Hz), x100

Index 10 (0.9 Hz), x100 (Best)

Index 28 (2.7 Hz), x100

**baby2.mp4**

Index 26 (0.83 Hz), x100

Index 50 (1.63 Hz), x100

Index 50 (1.63 Hz), x300

Index 52 (1.7 Hz), x100

Index 61 (2.0 Hz), x100

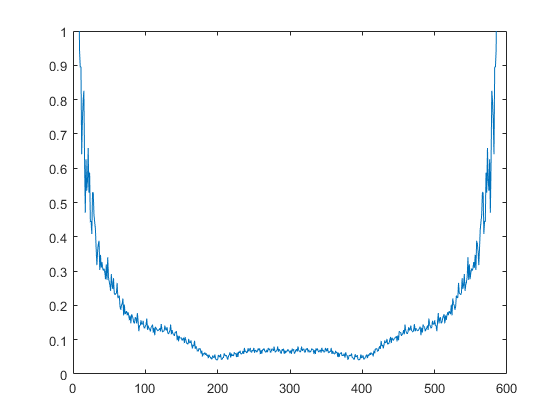
Index 226 (7.5 Hz), x100

**Q5. CAPTURE AND MOTION-MAGNIFY YOUR OWN VIDEO(S)**

**Origial Video**

**Result**

taewoo.mp4의 frequency별 평균 magnitude



Index 21 (1.01 Hz), x100(호흡으로 추정)

Index 48 (2.38 Hz), x100(고갯짓)

Index 81 (4.1 Hz), x100(눈 깜빡임)