DATViewer
User Manual

ACTIVE OPTICAL SYSTEMS
2021 Girard Blvd. Suite 150
Albuquerque, NM 87106
(505) 245-9970 x184
www.aos-llc.com

Table of Contents

1 Introduction

The DATViewer is a software utility to enable basic image processing and viewing of a set of camera images recorded in another software suite like the AOSViewer or the VimbaLogger.

2 Usage

2.1 GUI Architecture

Like most of the AOS software tools, the user input and settings are on the left of the screen and the outputs are on the right side of the screen. In this software, there are tabs on both sides of the screen. The tabs on the left organize the inputs. The tabs on the right allow the user to customize the displayed output.

2.2 Loading a DAT File

Upon opening the DATViewer, the user loads a file by selecting the Load DAT File button or by dragging and dropping the DAT file onto the DATViewer form or by associating the DAT file with the DATViewer and double clicking.

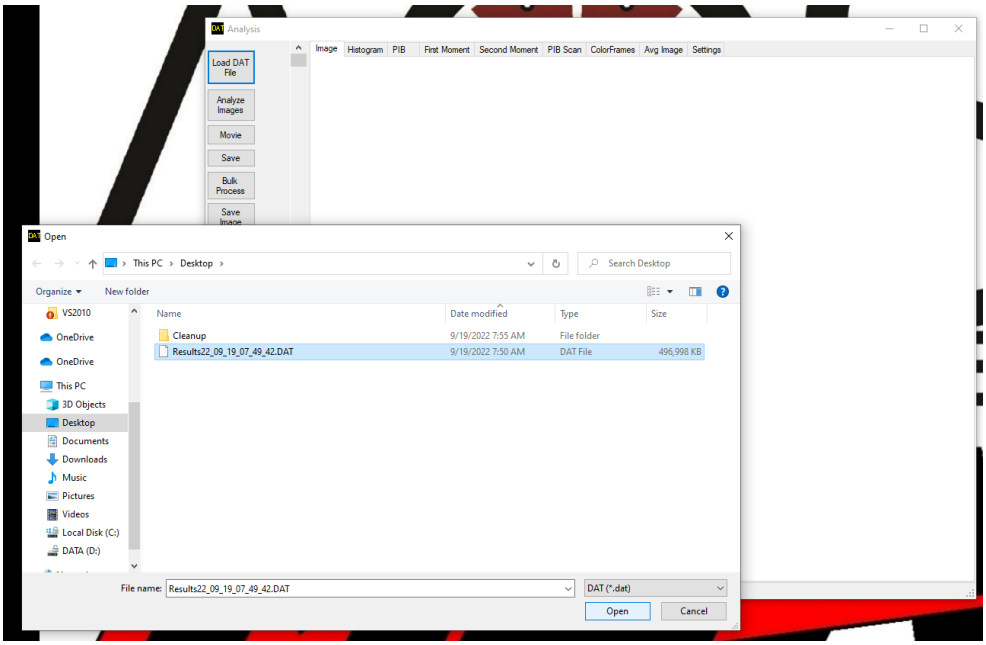

Figure 1: Screenshot of DAT File Loading

Figure 2 shows the screen after loading the DAT file into DATViewer. The first frame is immediately shown in the Image tab on the right (results) side of the screen. The vertical scroll bar shows the number of images and the currently selected image index and allows the user to scroll through the image set. The image can be put into a false color mode using the Use Colorbar checkbox and forcing a change to the display. The Min and Max Count boxes can be used to limit the range of the image being displayed.

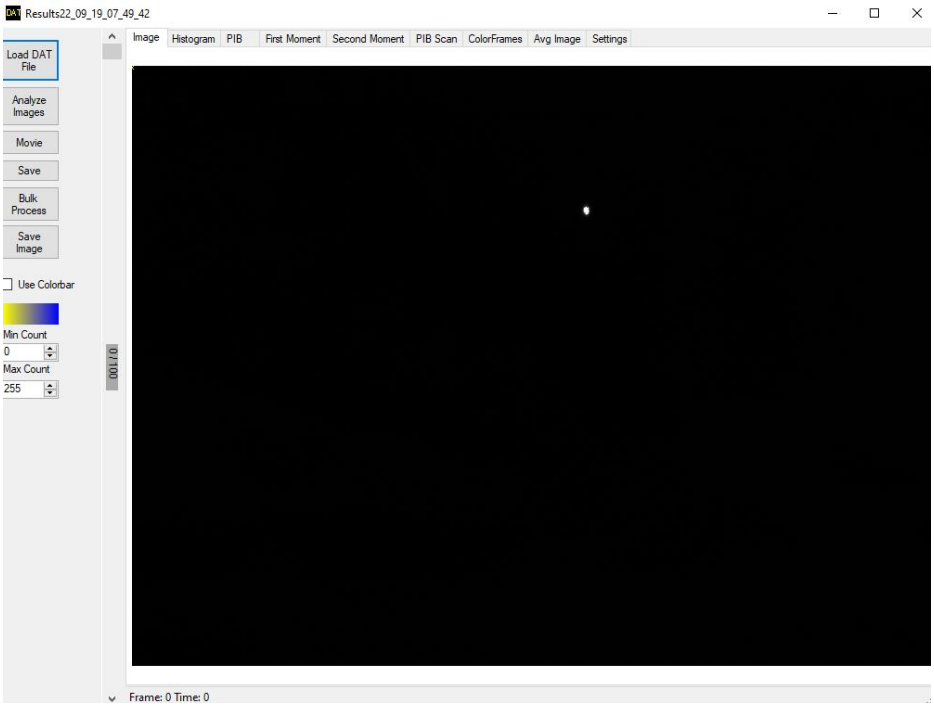

Figure 2: DATViewer after File Loaded

2.3 Analysis Settings

Figure 3 shows the Settings tab on the far right of the display which is used to adjust parameters of the analysis. This is a departure from the general philosophy of making the inputs on the left and results on the right because we wanted to maximize the image display screen real estate. On this tab, the user can see several different GUI elements for adjusting parameters of the analysis. In the Image Settings group box, the user can see the current image data in a text box (image pixel size and index) and adjust the image flipping in X and Y through checkboxes. The user can also force the program to show color frames separately in the Color Frames tab.

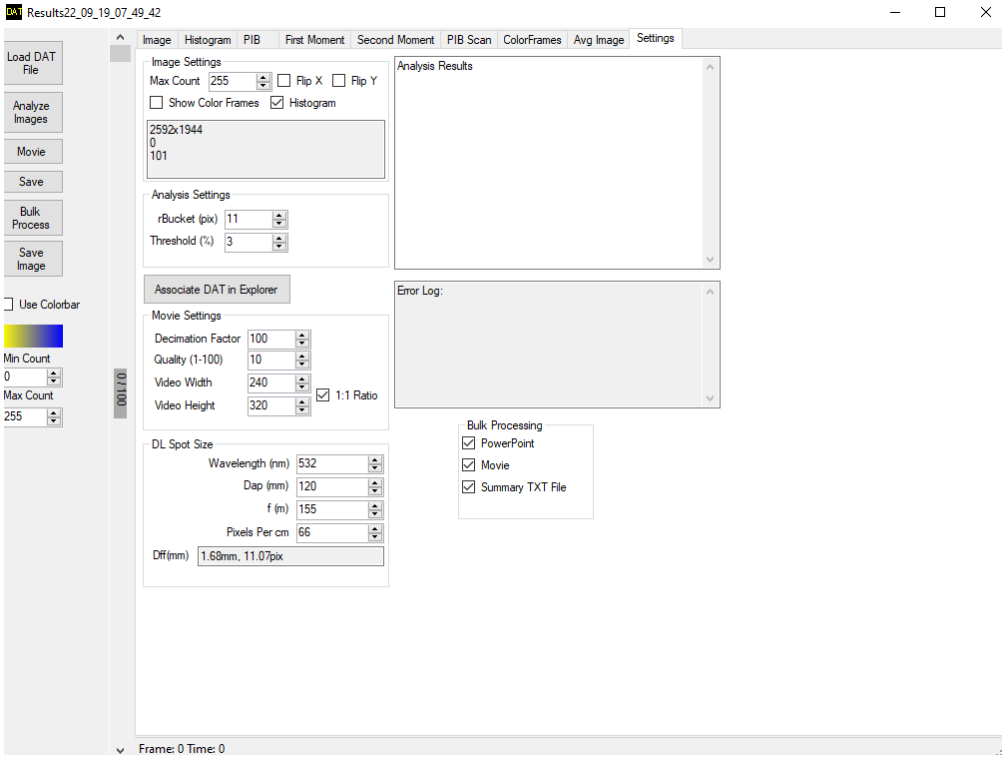

Figure 3: DATViewer Settings

The user can also select the Histogram checkbox to force the program to calculate the Histogram. Figure 4 shows an example Histogram output for an individual frame. This should be left off when possible to maximize calculation throughput.

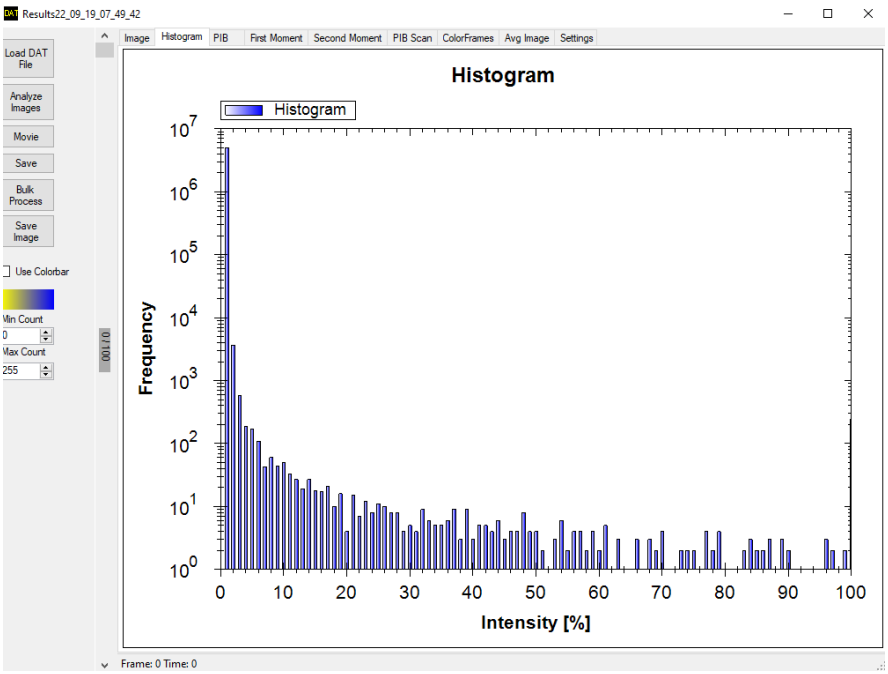

Figure 4: Example Histogram Output

The Analysis Settings group box allows the user to select the bucket radius for the PIB calculations and the threshold for the moments calculations in percent. The Associate DAT in Explorer button allows the user to associate DAT files with the DATViewer such that double clicking them in Windows will open them in the DATViewer.

The Movie Settings group box allows the user to select parameters of the movie making process. The Decimation Factor is the number of frames that are incremented in the movie output. This is useful when making a huge data set into a movie with a reduced frame count. The Quality input is to adjust the percent quality of the movie output in the AVI format. The output Video Width and Height allow the user to specify the movie output frame size in pixels.

The DL Spot Size group box allows the user to calculate the diffraction limited spot size in pixels based on the input wavelength, aperture diameter, focal length, and pixel size in pixels per centimeter.

The top right text box shows a summary of the processing results. The bottom right is an Error Log.

The Bulk Processing group box allows the user to select analysis and processing steps when analyzing multiple DAT files.

2.4 Starting Analysis

The Analyze Images button on the left side of the screen starts the image analysis process.

2.4.1 Moments

The thresholded first and second moments are calculated for each image with the results displayed on the First Moment and Second Moment tabs. Figure 5 shows example outputs of the first and second moments vs. frame index for an example taken from a laser pointer on a wall moving up and down, then left and right, then in a circle.

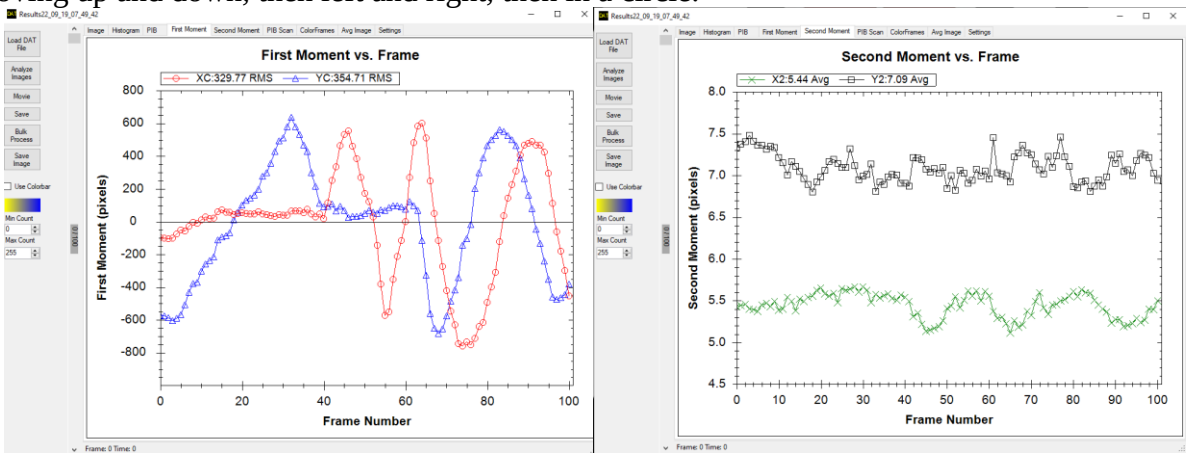

Figure 5: Example Outputs of the First and Second Moments

2.4.2 Power in the Bucket (PIB)

The PIB is also calculated for every frame in the DAT file during the analysis. Figure 6 shows the calculated PIB vs. radius for an individual frame and the PIB for the user-specified radius from the Settings tab for every frame in the DAT file.

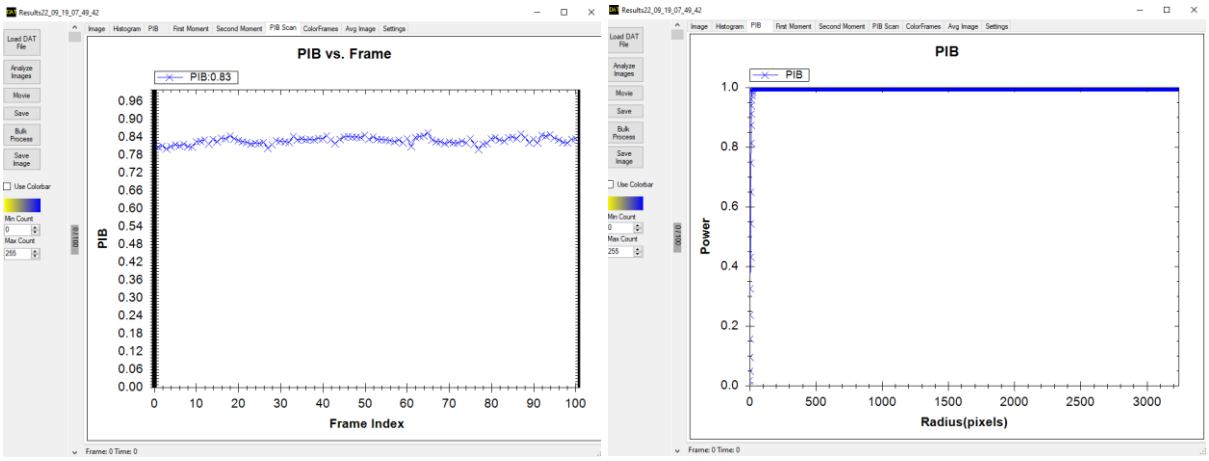

Figure 6: (left) PIB for User-Specified Radius for Every Frame; (right) Individual Frame PIB vs. Radius

2.4.3 Average Images

Figure 7 shows the results on the Avg Image tab. While analyzing each image, the software calculates the average image without recentering on the left which shows the path traced by the laser pointer in this demonstration. On the top right it shows the average image with centroid recentering, which shows a single high intensity spot because the motion was slow relative to the exposure time. It also shows the PIB for both average image cases.

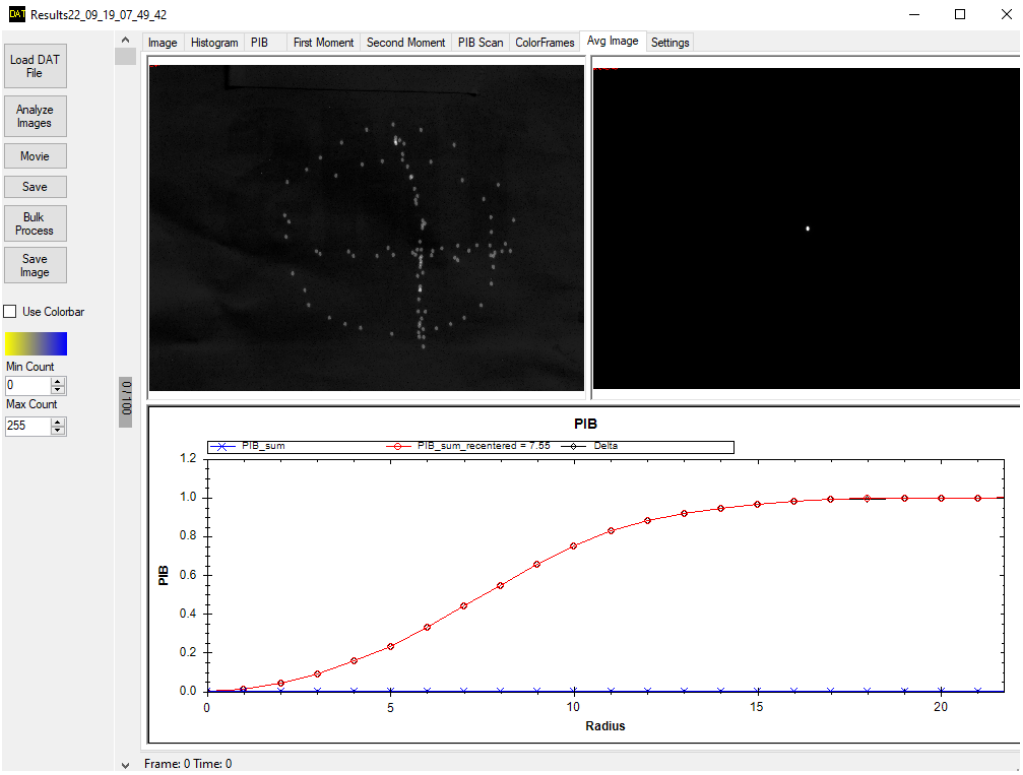

Figure 7: Example Average Image Output and Average Image PIB vs. Radius

2.4.4 File Outputs

Upon completion, the results are saved to a folder where the data file was located. Figure 8 shows the generated files from the analysis. The x2 and y2.csv files are a list of the x and y second moments. The xc and yc.csv are a list of the first moment results. PIBScan.csv is the normalized PIB obtained at the user-specified radius on a 0 to 1 scale. Average.tif is the average image without recentering. Recentered_average.tif is the average image after recentering each frame on the centroid location.

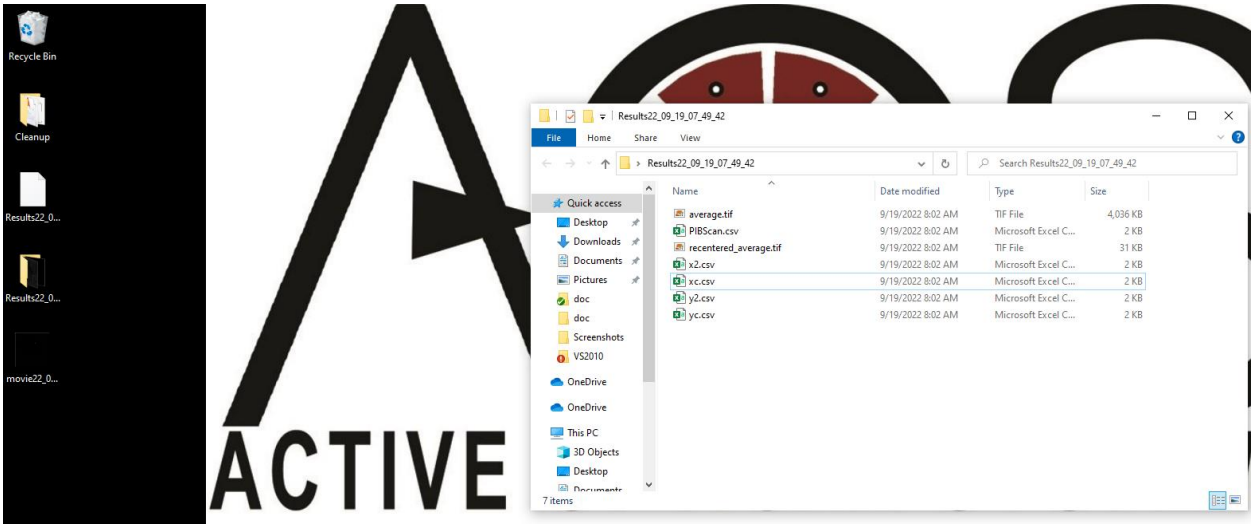

Figure 8: Files Generated During Analysis

2.5 Other Functionality

The Movie button allows the user to make an AVI movie file from the data using the settings on the Settings tab. The Save Image button allows the user to save an individual image from the DAT file. The Bulk Process button allows the user to select multiple DAT files and process all the data using the same settings.

3 Appendix

The DAT file format is a simple binary file format for recording image data. This file format is owned by AOS and is subject to change at any time. What follows is accurate at the time of this writing.

3.1 DAT File Header Contents

Description	Size	Notes
Version	Double (8 bytes)	
Bit Depth	Int32 (4 bytes)	8 or 16 typically (bits per pixel)

3.2 DAT Image Record Contents

Description	Size	Notes
Frame ID Number	Uint64	Index
Time	Double	
Width	Uint32	Pixels
Height	Uint32	Pixels
Buffer Size	Uint32	Total byte count
Image Data	Uint8 or Uint16	Array of Pixel Data

3.3 DAT Reader Matlab Code

```
function [img,timeIn,bitdepth]=ReadDAT(fname,index)
%function [img,timeIn,bitdepth]=ReadDAT(fname,index)
% fname = string file name with path if not in local directory
% index = vector of indices [optional]
% img = cell array of images
% timeIn = vector of times
% bitdepth = number of bits in the data (negative for color Bayer packed images)
img=[];
timeIn=[];
if (~exist('fname','var'))
    error('Need a file name');
end;
if (~exist('index','var'))
    index=[];
end;

fp = fopen(fname, 'rb');
if (fp<=0)
    disp('Could not open file');
    return;
end;

frame_ids = [];
fb = fname(1:(length(fname)-4));
if (~isempty(strfind(fb,'_')))

    [nums,strs]=parse(fb,'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ_.');
    if (length(nums)>5)
        year=nums(1);
```

```
        month=nums(2);
        day=nums(3);
        hour=nums(4);
        min=nums(5);
        sec=nums(6);
    end;
else
    nums = zeros(1,8);
    year=nums(1);
    month=nums(2);
    day=nums(3);
    hour=nums(4);
    min=nums(5);
    sec=nums(6);
end;

version = fread(fp, 1, 'double', 'ieee-le');
v=2.0;
if (version~=2.0 && version~=3.0)
    v=1.0;
    fseek(fp,0,'bof');
else
    v=version;
end;

bitdepth=8;
if (v>=3)
    bitdepth = fread(fp, 1, 'int32', 'ieee-le');
end;
%disp(sprintf('Version: %.1f',v));

cnt = 1;
while ~feof(fp)
    position = ftell(fp);
    frameID = fread(fp, 1, 'uint64', 'ieee-le');
    if(isempty(frameID))% || frameID == 0)
        break;
    end
    if (v>1.5)
        timeIn(cnt) = fread(fp,1,'double','ieee-le');
    end;
    width = fread(fp, 1, 'uint32', 'ieee-le');
    height = fread(fp, 1, 'uint32', 'ieee-le');
    buf_size = fread(fp, 1, 'uint32', 'ieee-le');
    bytesPerPixel = buf_size / (width*height);
    if (bytesPerPixel==1)
        data = fread(fp, buf_size, 'uint8', 'ieee-le');
    else
        data = fread(fp, width*height, 'uint16', 'ieee-le');
    end;

    if (length(index)==1 && index~=1)
        headersize = 20;
        if (v>1.5) headersize = headersize + 8; end;
```

```
        startOffset = 0;
        if (v > 1.5) startOffset = 8; end; % static offset of 8 for version number
double
        if (v > 2.5) startOffset = 12; end;  % additional offset of 4 for bit depth

        fileOffset = ((buf_size+headersize)*index) + startOffset;
        fseek(fp,fileOffset,'bof');
        %         fseek(fp,(headersize+buf_size)*(index-1),'cof');

        frameID = fread(fp, 1, 'uint64', 'ieee-le');
        if(isempty(frameID))% || frameID == 0)
            break;
        end
        if (v>1.5)
            timeIn(cnt) = fread(fp,1,'double','ieee-le');
        end;
        width = fread(fp, 1, 'uint32', 'ieee-le');
        height = fread(fp, 1, 'uint32', 'ieee-le');
        buf_size = fread(fp, 1, 'uint32', 'ieee-le');
        data = fread(fp, buf_size, 'uint8', 'ieee-le');
    end;

    frame_ids(cnt+1) = frameID;

    if (width*height~=length(data))
        warning('Data lengths do not match');
        return;
    end
    imgTotal = reshape(data,width,height)';

    img{cnt}=imgTotal;

    cnt = cnt + 1;

    if (length(index)==1)
        img = img{1};
        break;
    end;

    if (cnt>max(index))
        break;
    end;
end
fclose(fp);
if (length(index)==1)
    return;
end;
if (length(index)>0 && v>1.5)
    img = img{index};
    timeIn = timeIn(index);
end;
return;
```