Framework

ModuleProps

Base Properties class for all modules

```
int fps; // can be updated during runtime with setProps
bool logHealth; // can be updated during runtime with setProps
size_t qlen;
QuePushStrategyType quePushStrategyType;
```

QuePushStrategyType

```
enum QuePushStrategyType {
     BLOCKING,
     NON_BLOCKING_ANY, // try push independently to child modules
     NON_BLOCKING_ALL_OR_NONE // try push only if all the child modules have the que free or
};
```

Module

Base class for all the modules

```
void addOutputPin(framemetadata_sp& metadata, string& pinId);
bool setNext(boost::shared_ptr<Module> next);
```

Pipeline

```
PipeLine(string name)
bool appendModule(boost::shared_ptr<Module> pModule);
bool init();
void run_all_threaded();

void stop();
void term();
void wait_for_all();
```

ExternalSourceModule

```
std::pair<bool, uint64_t> produceExternalFrame(ApraData* data)
```

- `locked` is atomically incremented and decremented
- std::pair<bool, uint64_t> is returned
- if data->fIndex == 0 fIndex is incremented by the framework and returns the value
- if data->fIndex != 0 fIndex is same as data->fIndex
- if `false` is returned, then the data is not accepted.Framework Que is full and there is no bandwidth to process more data
- it is upto the caller application to act on the response of `produceExternalFrame`

```
bool stop()
when ExternalSourceModule is not attached to the pipeline, manually call stop to trigger a t
```

FileReaderModuleProps

```
FileReaderModuleProps(const std::string& strFullFileNameWithPattern, int startIndex = 0, int
```

- loops till the `maxIndex` is reached `-1` loop the entire pattern/directory
- starts the loop from `startIndex`

FileReaderModule

```
FileReaderModule(FileReaderModuleProps props);
```

JPEGDecoderIMProps

```
JPEGDecoderIMProps(bool sw);
```

- `true` if deocde is on software
- `false` if decode is on hardware

JPEGDecoderIM

```
JPEGDecoderIM(JPEGDecoderIMProps props);
```

- Currently only supports single channel output

CalcHistogramCVProps

```
CalcHistogramCVProps(int bins);
// All the properties can be updated during run time using setProps
int bins;
vector<int> roi;
string maskImgPath;
```

- `bins` number of histogram bins

- `maskImgPath` absolute path of mask image path
  Expected to be Same resolution as the dataset
  if both roi and mask_img_path is specified roi is used and mask_img_path ignored

- `roi` topleft_x topleft_y width height
  if both roi and mask_img_path is specified roi is used and mask_img_path ignored

CalcHistogramCV

```
CalcHistogramCV(CalcHistogramCVProps props);
CalcHistogramCVProps getProps();
void setProps(CalcHistogramCVProps props);

// Example on how to change Properties dynamically
```

```
auto histogram = new CalcHistogramCV(CalcHistogramCVProps(32));
auto newHistProps = histogram->getProps();
newHistProps.bins = newHistProps.bins = 16;
histogram->setProps(newHistProps);
```

ChangeDetectionProps

```
ChangeDetectionProps(int refWindoweLength, int refDelayLength, int insWindowLength, double t
// All the properties can be updated during run time using setProps
```

- `refWindowLength` Number of frames to average
  Assume `fps = 30` and you want to average `2` seconds then specify `refWindowLength` as `60`
- `refDelayLength` Number of frames to delay from the current frame
  default `-1` - Static reference
  if moving average is required speicfy the delay
  Assume `fps = 30` and you want to delay 2 minutes then specify `refDelayLength` as `3600`
- `insWindowLength` Number of frames to average
  Assume `fps = 30` and you want to average 0.5 seconds then specify `insWindowLength` as `15`
- `compareMethod` https://docs.opencv.org/4.1.1/d8/dc8/tutorial_histogram_comparison.html
  HISTCMP_CHISQR (1) works for all the datasets provided

ChangeDetection

Sends ChangeDetectionResult to the next module

```
ChangeDetection(ChangeDetectionProps props);
ChangeDetectionProps getProps();
void setProps(ChangeDetectionProps props);

// Example on how to change Properties dynamically
auto module = new ChangeDetection(ChangeDetectionProps(60, -1, 30, 1, 1));
auto newProps = module->getProps();
newProps.threshold = 2;
module->setProps(newProps);
```

JPEGEncoderIMProps

```
bool sw; // software or hardware mode
unsigned short quality; // range 1-100 JPEG Compression quality - 100 means best quality but
```

JPEGEncoderIM

Intel Media SDK based encoder

```
JPEGEncoderIM(JPEGEncoderProps props);
```

- Currently only supports single channel as input

FileWriterModuleProps

```
FileWriterModuleProps(const std::string& strFullFileNameWithPattern);

// Example
auto props = new FileWriterModuleProps("C:/Users/developer/Downloads/Sai/temp/enc/frame_????
```

FileWriterModule

Basic FileWriter

```
FileWriterModule(FileWriterModuleProps props);
```

LoggerProps

```
bool enableConsoleLog;
bool enableFileLog;
std::string fileLogPath;
boost::log::trivial::severity_level logLevel;
```

Logger

```
static void initLogger(LoggerProps props); // valid only once - if called second time - noth
void setLogLevel(boost::log::trivial::severity_level severity);
void setConsoleLog(bool enableLog);
void setFileLog(bool enableLog);

The below macros are available for logging
LOG_TRACE << "A trace severity message";
LOG_DEBUG << "A debug severity message";
LOG_INFO << "An informational severity message";
LOG_WARNING << "A warning severity message";
LOG_ERROR << "An error severity message";
LOG_FATAL << "A fatal severity message";
```