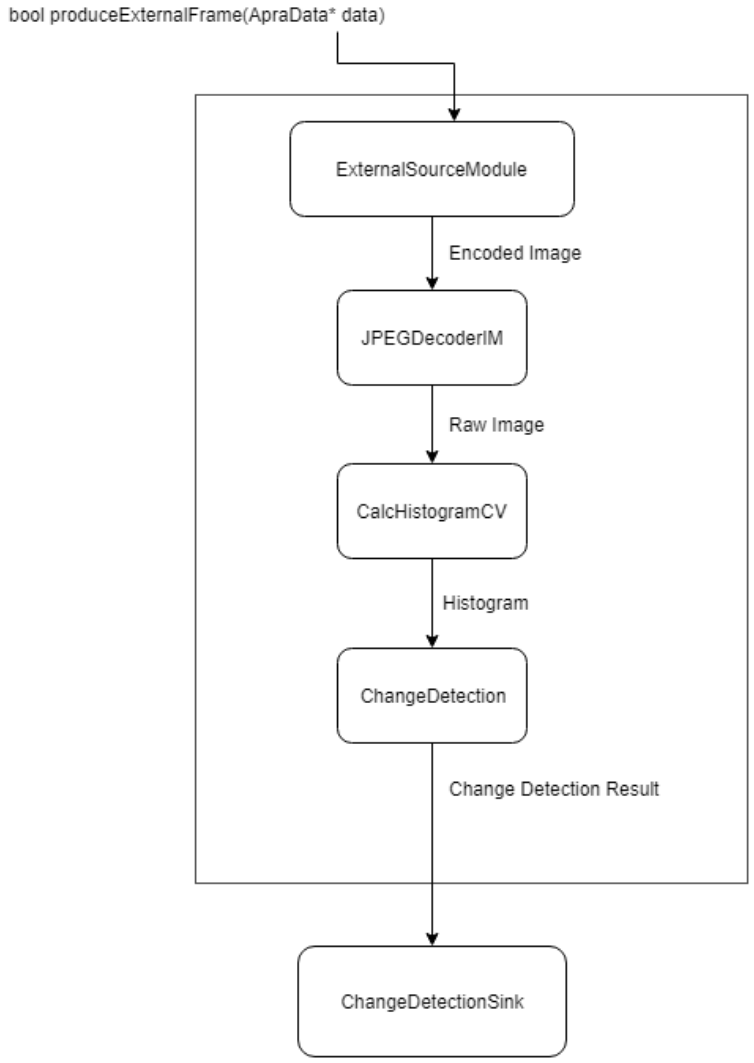
Integration

Flow



ApraData

```
ApraData(void* _buffer, size_t _size, uint64_t _fIndex)
size_t getLocked()
```

- `getLocked() !=0` means framework is using the buffer
- buffer should **not be modified/deleted** till `getLocked()` returns `0`

Initialize ExternalSourceModule

```
auto source = boost::shared_ptr<ExternalSourceModule>(new ExternalSourceModule());
source->addOutputPin(framemetadata_sp(new FrameMetadata(FrameMetadata::ENCODED_IMAGE)));
source->init();
```

Sending buffer to the pipeline

```
auto data = boost::shared_ptr<ApraData>(new ApraData(buffer, size, 0));
std::pair<bool, uint64_t> ret = source->produceExternalFrame(data->get());
if (!ret.first)
{
      // failed to push to the que
      // Que is full
}
```

Checking if buffer can be reused/deleted

```
if (data->getLocked() == 0)
{
     // buffer can be reused/deleted
}
```

ChangeDetectionResult

```
bool mChangeDetected;

double mDistance;
chi square distance

uint64_t fIndex;
frame index - propagated from source to sink
```

ChangeDetectionSink

```
bool process(frame_container& frames);

boost::shared_ptr<ChangeDetectionResult> res;
result is available inside process function
```

- `res->fIndex` can be used with produceExternalFrame to manage memory
- store the output `fIndex` of `produceExternalFrame` and input `buffer`
- when `res->fIndex` comes, correspoding buffer can be reused/deleted

How to use ChangeDetectionResult

```
class ECSSink : public ChangeDetectionSink
{
public:
     ECSSink(bool _debug=false) : ChangeDetectionSink(ChangeDetectionSinkProps(_debug))
     {
             debug = _debug;
     }

     virtual ~ECSSink() {}
protected:
     void handleResult(ChangeDetectionResult& res)
     {
             if (false)
             {
                     LOG_INFO << "frameIndex<" << res.fIndex << "> changeDetected<" << res.m
                     return;
```

```
            }

            static bool curStatus = true;
            if (curStatus != res.mChangeDetected)
            {
                    std::cout << std::endl;
                    std::cout << "frameIndex<" << res.fIndex << "> changeDetected<" << std:
                    curStatus = res.mChangeDetected;
            }
            if (!res.mChangeDetected)
            {
                    std::cout << ".";
            }
            else
            {
                    std::cout << "*";
            }
    }

    bool debug;
};

// Connect this to the ChangeDetectionModule instead of ChangeDetectionSink
auto ecsSink = boost::shared_ptr<ECSSink>(new ECSSink());
changeDetection->setNext(ecsSink);
```