**Oat** is a set of programs for processing images, extracting object position information, and streaming data to disk and/or the network in real-time. Oat subcommands are independent programs that each perform a single operation but that can communicate through shared memory. This allows a user to chain operations together in arrangements suitable for particular context or tracking requirement. This architecture enables scripted construction of custom data processing chains. Oat is primarily used for real-time animal position tracking in the context of experimental neuroscience, but can be used in any circumstance that requires real-time object tracking.

build passing

### Contributors

- jonnew http://www.mit.edu/~jpnewman/

### Table of Contents

## Manual

### Introduction

Oat's design is influenced by the UNIX philosophy, suckless tools, and MEABench. Oat consists of a set of small, composable programs (called **components**). Components are equipped with standard interfaces that permit communication through shared memory to capture, process, and record video streams. Currently, Oat components act on two basic data types: `frames` and `positions`.

- `frame` - Video frame.
- `position` - 2D position.

Oat components can be chained together to realize custom dataflow networks that operate on instances of the aforementioned datatypes, called **tokens**. Token processing pipelines can be split and merged while maintaining thread-safety and sample synchronization. The messaging library underlying the communication between Oat components has been optimized to reduce token copying. For instance, `frame` passing is performed using a zero-copy protocol. This means that passing `frames` between components in a user-configured processing network incurs almost no memory or CPU cost compared to the monolithc equivalent. Further, great care was taken during implementations of Oat components to minimize time spent in critical sections. This means that individual components execute largely in parallel, even when components are highly interdependent, facilitating efficient use of multi-core CPUs and GPU-based processing acceleration.

To get a feel for how Oat is used, here is a script to detect the position of a single object in pre-recorded video file:

```bash
# Serve frames from a video file to the 'raw' stream
oat frameserve file raw -f ./video.mpg &

# Perform background subtraction on the 'raw' stream
# Serve the result to the 'filt' stream
# If an appropriately configured GPU is available, this process will
# use it
oat framefilt mog raw filt &

# Perform color-based object position detection on the 'filt' stream
# Serve the object positionto the 'pos' stream. Allow parameter tuning
# through a simple GUI.
oat posidet hsv filt pos --tune &

# Decorate the 'raw' stream with the detected position form the `pos` stream
# Serve the decorated images to the 'dec' stream
oat decorate -p pos raw dec &
```

```bash
# View the 'dec' stream
oat view frame dec &

# Record the 'dec' and 'pos' streams to file in the current directory
oat record -i dec -p pos -f ./
```

This script has the following graphical representation:

```
frameserve ---> framefilt ---> posidet ---> decorate ---> view
           \                          \    /        \
             -----------------------------           ---> record
                                        \           /
                                          ---------
```

Generally, an Oat component is called in the following pattern:

```
oat <component> [TYPE] [IO] [CONFIGURATION]
```

The `<component>` indicates the component that will be executed. Components are classified according to their type signature. For instance, `framefilt` (frame filter) accepts a frame and produces a frame. `posifilt` (position filter) accepts a position and produces a position. `frameserve` (frame server) produces a frame, and so on. The `TYPE` parameter specifies a concrete type of transform (e.g. for the `framefilt` component, this could be `bsub` for background subtraction). The `IO` specification indicates where the component will receive data from and to where the processed data should be published. A description of a component's purpose, its available TYPEs and correct IO specification can be examined using the `--help` command line switch

```bash
oat <component> --help
```

The `CONFIGURATION` specification is used to provide parameters to shape the component's operation and are TYPE-specific. Information on program options for a particular concrete transform TYPE can be printed using

```bash
oat <component> <type> --help
```

In addition to command line input, all options can be specified using a configuration file which is provided to the program using the `-c` command line argument.

```
  -c [ --config ] arg             Configuration file/key pair.
                                  e.g. 'config.toml mykey'
```

For instance:

```bash
oat frameserve gige raw -c config.toml gige-config
```

The configuration file may contain many configuration tables that specify options for multiple oat programs. These tables are addressed using a key (`gige-config`) in the example above. Configuration files are written in plain text using TOML.

A multi-component processing script can share a configuration file because each component accesses parameter information using a file/key pair, like so

```
[key]
parameter_0 = 1                 # Integer
parameter_1 = true              # Boolean
parameter_2 = 3.14              # Double
parameter_3 = [1.0, 2.0, 3.0]   # Array of doubles
```

or more concretely,

```
# Example configuration file for frameserve --> framefilt
[frameserve-config]
frame_rate = 30                 # FPS
roi = [10, 10, 100, 100]        # Region of interest

[framefilt-config]
mask = "~/Desktop/mask.png"     # Path to mask file
```

These could then be used in a processing script as follows:

```
oat frameserve gige raw -c config.toml frameserve-config &
oat framefilt mask raw filt -c config.toml framefilt-config
```

The type and sanity of parameter values are checked by Oat before they are used. Below, the type signature, usage information, available configuration parameters, examples, and configuration options are provided for each Oat component.

### Frame Server

`oat-frameserve` - Serves video streams to shared memory from physical devices (e.g. webcam or GIGE camera) or from file.

#### Signature

```
oat-frameserve --> frame
```

#### Usage

```
Usage: frameserve [INFO]
   or: frameserve TYPE SINK [CONFIGURATION]
Serve frames to SINK.

INFO:
  --help                 Produce help message.
  -v [ --version ]       Print version information.

TYPE
  wcam: Onboard or USB webcam.
  usb: Point Grey USB camera.
  gige: Point Grey GigE camera.
  file: Video from file (*.mpg, *.avi, etc.).
  test: Write-free static image server for performance testing.

SINK:
  User-supplied name of the memory segment to publish frames to (e.g. raw).
```

#### Configuration Options

**TYPE = `wcam`**

```
  -i [ --index ] arg      Camera index. Useful in multi-camera imaging
                          configurations. Defaults to 0.
  -r [ --fps ] arg        Frames to serve per second. Defaults to 20.
  --roi arg               Four element array of unsigned ints,
                          [x0,y0,width,height],defining a rectangular region of
                          interest. Originis upper left corner. ROI must fit
                          within acquiredmat size. Defaults to full sensor
                          size.
```

**TYPE = `gige` and `usb`**

**TYPE** = `file`

```
  -f [ --video-file ] arg   Path to video file to serve frames from.
  -r [ --fps ] arg          Frames to serve per second.
  --roi arg                 Four element array of unsigned ints,
                            [x0,y0,width,height],defining a rectangular region
                            of interest. Originis upper left corner. ROI must
                            fit within acquiredframe size. Defaults to full
                            video size.
```

**TYPE** = `test`

```
  -f [ --test-image ] arg   Path to test image used as frame source.
  -C [ --color ] arg        Pixel color format. Defaults to BGR.
                            Values:
                              GREY:  8-bit Greyscale image.
                              BGR: 8-bit, 3-chanel, BGR Color image.

  -r [ --fps ] arg          Frames to serve per second.
  -n [ --num-frames ] arg   Number of frames to serve before exiting.
```

#### Examples

```
# Serve to the 'wraw' stream from a webcam
oat frameserve wcam wraw

# Stream to the 'graw' stream from a point-grey GIGE camera
# using the gige_config tag from the config.toml file
oat frameserve gige graw -c config.toml gige_config

# Serve to the 'fraw' stream from a previously recorded file
# using the file_config tag from the config.toml file
oat frameserve file fraw -f ./video.mpg -c config.toml file_config
```

### Frame Filter

`oat-framefilt` - Receive frames from a frame source, filter, and publish to a second memory segment. Generally used to pre-process frames prior to object position detection. For instance, `framefilt` could be used to perform background subtraction or application of a mask to isolate a region of interest.

#### Signature

```
frame --> oat-framefilt --> frame
```

#### Usage

```
Usage: framefilt [INFO]
   or: framefilt TYPE SOURCE SINK [CONFIGURATION]
Filter frames from SOURCE and publish filtered frames to SINK.

INFO:
  --help                 Produce help message.
  -v [ --version ]       Print version information.

TYPE
  bsub: Background subtraction
  col: Color conversion
  mask: Binary mask
  mog: Mixture of Gaussians background segmentation.
  undistort: Correct for lens distortion using lens distortion model.
  thresh: Simple intensity threshold.

SOURCE:
  User-supplied name of the memory segment to receive frames from (e.g. raw).

SINK:
  User-supplied name of the memory segment to publish frames to (e.g. filt).
```

#### Configuration Options

**TYPE = `bsub`**

```
  -a [ --adaptation-coeff ] arg  Scalar value, 0 to 1.0, specifying how
                                 quickly the new frames are used to update the
                                 backgound image. Default is 0, specifying no
                                 adaptation and a static background image that
                                 is never updated.
  -f [ --background ] arg        Path to background image used for
```

```
                                      subtraction. If not provided, the first frame
                                      is used as the background image.
```

**TYPE = mask**

```
  -f [ --mask ] arg       Path to a binary image used to mask frames from
                          SOURCE. SOURCE frame pixels with indices
                          corresponding to non-zero value pixels in the mask
                          image will be unaffected. Others will be set to zero.
                          This image must have the same dimensions as frames
                          from SOURCE.
```

**TYPE = mog**

```
  -a [ --adaptation-coeff ] arg  Value, 0 to 1.0, specifying how quickly the
                                 statistical model of the background image
                                 should be updated. Default is 0, specifying
                                 no adaptation.
```

**TYPE = undistort**

```
  -k [ --camera-matrix ] arg       Nine element float array, [K11,K12,...,K33],
                                   specifying the 3x3 camera matrix for your
                                   imaging setup. Generated by oat-calibrate.
  -d [ --distortion-coeffs ] arg   Five to eight element float array,
                                   [x1,x2,x3,...], specifying lens distortion
                                   coefficients. Generated by oat-calibrate.
```

**TYPE = thresh**

```
  -I [ --intensity ] arg   Array of ints between 0 and 256, [min,max],
                           specifying the intensity passband.
```

### Examples

```
# Receive frames from 'raw' stream
# Perform background subtraction using the first frame as the background
# Publish result to 'sub' stream
oat framefilt bsub raw sub

# Receive frames from 'raw' stream
# Change the underlying pixel color to single-channel GREY
oat framefilt col raw gry -C GREY

# Receive frames from 'raw' stream
```

```bash
# Apply a mask specified in a configuration file
# Publish result to 'roi' stream
oat framefilt mask raw roi -c config.toml mask-config
```

### Frame Viewer

`oat-view` - Receive frames from named shared memory and display them on a monitor. Additionally, allow the user to take snapshots of the currently displayed frame by pressing s while the display window is in focus.

#### Signature

```
token --> oat-view
```

#### Usage

```
Usage: view [INFO]
   or: view TYPE SOURCE [CONFIGURATION]
Graphical visualization of SOURCE stream.

INFO:
  --help                 Produce help message.
  -v [ --version ]       Print version information.

TYPE
  frame: Display frames in a GUI  pose: Display a pose stream in a GUI

SOURCE:
  User-supplied name of the memory segment to receive frames from (e.g. raw).
```

#### Configuration Options

**TYPE = `frame`**

```
  --control-endpoint arg       ZMQ style endpoint specifier designating runtime
                               control port:'<transport>://<host>:<port>'. For
                               instance, 'tcp://*:5555' to specify TCP
                               communication on port 5555. Or, for interprocess
                               communication: '<transport>://<user-named-pipe>.
                               For instance 'ipc:///tmp/test.pipe'. Internally,
                               this is used to construct a ZMQ REQ socket that
                               that receives commands from oat-control.
                               Defaults to ipc:///tmp/oatcomms.pipe.

  -r [ --display-rate ] arg    Maximum rate, in Hz, at which the viewer is
                               updated irrespective of its source's rate. If
                               frames are supplied faster than this rate, they
                               are ignored. Setting this to a reasonably low
                               value prevents the viewer from consuming
                               processing resources in order to update the
```

```
                              display faster than is visually perceptible.
                              Defaults to 30.
  -m [ --min-max ] arg        2-element array of floats, [min,max], specifying
                              the requested dyanmic range of the display.
                              Pixel values below min will be mapped to min.
                              Pixel values above max will be mapped to max.
                              Others will be interprolated between min and
                              max. Defaults to off.
  -f [ --snapshot-path ] arg  The path to which in which snapshots will be
                              saved. If a folder is designated, the base file
                              name will be SOURCE. The time stamp of the
                              snapshot will be prepended to the file name.
                              Defaults to the current directory.
```

### Example

```
# View frame stream named raw
oat view frame raw

# View frame stream named raw and specify that snapshots should be saved
# to the Desktop with base name 'snapshot'
oat view frame raw -f ~/Desktop -n snapshot
```

### Position Detector

`oat-posidet` - Receive frames from named shared memory and perform object position detection within a frame stream using one of several methods. Publish detected positions to a second segment of shared memory.

#### Signature

```
frame --> oat-posidet --> position
```

#### Usage

```
Usage: posidet [INFO]
   or: posidet TYPE SOURCE SINK [CONFIGURATION]
Perform object detection on frames from SOURCE and publish object positions to SINK.

INFO:
  --help                 Produce help message.
  -v [ --version ]       Print version information.

TYPE
  aruco: Aruco board pose estimation (color: any)
  diff: Motion detector (color: mono)
  hsv: HSV color thresholds (color: hsv)
  rpg: RPG pose estimator (color: mono)  thresh: Simple amplitude threshold (color: mono)


SOURCE:
  User-supplied name of the memory segment to receive frames from (e.g. raw).

SINK:
  User-supplied name of the memory segment to publish positions to (e.g. pos).
```

#### Configuration Options

**TYPE = `aruco`**

*Note:* Requires that OpenCV is compiled with contrib module support.

```
  -D [ --dictionary ] arg          Aruco board dictionary to use for detection
                                   or printing when -p is defined. Dictionaries
                                   are defined by the size of each marker and
                                   the number of markers in the dictionary.
                                   These parameters are encoded by a string of
                                   the form:
```

```
                                  <Size>X<Size>_<Number of Markers>

                                Values:
                                  4X4_50 (default)
                                  4X4_100
                                  4X4_250
                                  4X4_1000
                                  5X5_50
                                  5X5_100
                                  5X5_250
                                  5X5_1000
                                  6X6_50
                                  6X6_100
                                  6X6_250
                                  6X6_1000
                                  7X7_50
                                  7X7_100
                                  7X7_250
                                  7X7_1000

-k [ --camera-matrix ] arg      Nine element float array, [K11,K12,...,K33],
                                specifying the 3x3 camera matrix for your
                                imaging setup. Generated by oat-calibrate.
-d [ --distortion-coeffs ] arg  Five to eight element float array,
                                [x1,x2,x3,...], specifying lens distortion
                                coefficients. Generated by oat-calibrate.
-S [ --board-size ] arg         Two element int array, [X,Y], specifying the
                                dimensions of the Aruco board (the number of
                                markers in the X and Y directions).
-l [ --length ] arg             Length, in meters, of each side of the
                                square markers within the Aruco board.
-s [ --separation ] arg         Separation, in meters, between each of the
                                markers within the Aruco board.
-R [ --refine-detection ]       Perform a secondary marker location
                                refinement step using knowledge of the board
                                layout after initial marker detection is
                                performed. Can lead to improved pose
                                estimation robustness.
-p [ --print ]                  Prior to performing position detection,
                                print the specified Aruco marker to a PNG
                                file, named 'board.png', in the current
                                directory.
-P [ --print-scale ] arg        The number of pixels to map to marker length
                                to determine printing resolution. For
                                instance, print-scale 50 indicates that each
```

```
                                   side of the marker will be 50 pixels.
                                   Defaults to 100.
  -t [ --thresh-params ] arg       Three element vector, [min,max,step],
                                   specify
```

**TYPE = hsv**

```
  -H [ --h-thresh ] arg   Array of ints between 0 and 256, [min,max],
                          specifying the hue passband.
  -S [ --s-thresh ] arg   Array of ints between 0 and 256, [min,max],
                          specifying the saturation passband.
  -V [ --v-thresh ] arg   Array of ints between 0 and 256, [min,max],
                          specifying the value passband.
  -e [ --erode ] arg      Contour erode kernel size in pixels (normalized box
                          filter).
  -d [ --dilate ] arg     Contour dilation kernel size in pixels (normalized
                          box filter).
  -a [ --area ] arg       Array of floats, [min,max], specifying the minimum
                          and maximum object contour area in pixels^2.
  -t [ --tune ]           If true, provide a GUI with sliders for tuning
                          detection parameters.
```

**TYPE = diff**

```
  -d [ --diff-threshold ] arg   Intensity difference threshold to consider an
                                object contour.
  -b [ --blur ] arg             Blurring kernel size in pixels (normalized box
                                filter).
  -a [ --area ] arg             Array of floats, [min,max], specifying the
                                minimum and maximum object contour area in
                                pixels^2.
  -t [ --tune ]                 If true, provide a GUI with sliders for tuning
                                detection parameters.
```

**TYPE = rpg**

*Note:* Requires Oat compilation against Eigenlinear algebra template library.

```
  -M [ --marker-positions ] arg    Nx3 element array of 3D marker positions in
                                   meters.
  -T [ --threshold ] arg           Detection threshold value, 0-256. Defaults
                                   to 100.
  -s [ --gauss-sigma ] arg         Sigma of Gaussian blur filter. Defaults to
                                   0.6.
  -a [ --area ] arg                Array of floats, [min,max], specifying the
                                   minimum and maximum object contour area in
```

```
                                    pixels^2. Defaults to [10,200].
  -w [ --width-height-dist ] arg    Minimum ratio of width-to-height of a
                                    bounding box around a detected blob. Ideally
                                    the ratio of the width to the height of the
                                    bounding rectangle should be 1. Defaults to
                                    0.5.
  -c [ --circ-dist ] arg            Maximum allowable distortion, 0 to 1.0, of a
                                    bounding box around the detected blob,
                                    calculated as the area of the blob divided
                                    by pi times half the height or half the
                                    width of the bounding rectangle. Defaults to
                                    0.5.
  -b [ --back-proj-tol ] arg        Maximum allowable back projection tolerance,
                                    in pixels, to consider a projected point to
                                    correspond to the detected point in the
                                    captured image. Defaults to 5.
  -n [ --nn-pixel-tol ] arg         Tolerance, in pixels, that determines the
                                    correspondences between the LEDs and the
                                    detections in the image when predicting the
                                    position of the LEDs in the image. Defaults
                                    to 7.
  -C [ --pose-cert-thresh ] arg     Minimum ratio of how many of the
                                    back-projected points must be within the
                                    'back-proj-tol' for a correspondence between
                                    the LEDs and the detections to be correct.
                                    Defaults to 0.75.
  -p [ --pose-corr-thresh ] arg     Minimum correspondence ratio, 0 to 1.0, to
                                    continue with smart pose estimation rather
                                    than brute-force correspondence search.
                                    Defaults to 0.7.
  -r [ --roi-border ] arg           Thickness of the border (in pixels) around
                                    the predicted area of the LEDs in the image
                                    that defines the region of interest for
                                    image detection of the LEDs in the next
                                    frame. Defaults to 20.
  -k [ --camera-matrix ] arg        Nine element float array, [K11,K12,...,K33],
                                    specifying the 3x3 camera matrix for your
                                    imaging setup. Generated by oat-calibrate.
  -d [ --distortion-coeffs ] arg    Five to eight element float array,
                                    [x1,x2,x3,...], specifying lens distortion
                                    coefficients. Generated by oat-calibrate.
  -t [ --tune ]                     if true, provide a gui with sliders for
                                    tuning detection parameters.
```

**TYPE = `thresh`**

```
  -T [ --thresh ] arg     Array of ints between 0 and 256, [min,max],
                          specifying the intensity passband.
  -e [ --erode ] arg      Contour erode kernel size in pixels (normalized box
                          filter).
  -d [ --dilate ] arg     Contour dilation kernel size in pixels (normalized
                          box filter).
  -a [ --area ] arg       Array of floats, [min,max], specifying the minimum
                          and maximum object contour area in pixels^2.
  -t [ --tune ]           If true, provide a GUI with sliders for tuning
                          detection parameters.
```

**Example**

```
# Use color-based object detection on the 'raw' frame stream
# publish the result to the 'cpos' position stream
# Use detector settings supplied by the hsv_config key in config.toml
oat posidet hsv raw cpos -c config.toml hsv_config

# Use motion-based object detection on the 'raw' frame stream
# publish the result to the 'mpos' position stream
oat posidet diff raw mpos
```

### Position Generator

`oat-posigen` - Generate positions for testing downstream components. Publish generated positions to shared memory.

#### Signature

```
oat-posigen --> position
```

#### Usage

```
Usage: posigen [INFO]
   or: posigen TYPE SINK [CONFIGURATION]
Publish generated positions to SINK.

INFO:
  --help                  Produce help message.
  -v [ --version ]        Print version information.

TYPE
  rand2D: Randomly accelerating 2D Position

SINK:
  User-supplied name of the memory segment to publish positions to (e.g. pos).
```

#### Configuration Options

**TYPE = `rand2D`**

```
  -r [ --rate ] arg           Samples per second. Defaults to as fast as
                              possible.
  -n [ --num-samples ] arg    Number of position samples to generate and serve.
                              Deafaults to approximately infinite.
  -R [ --room ] arg           Array of floats, [x0,y0,width,height], specifying
                              the boundaries in which generated positions
                              reside. The room has periodic boundaries so when a
                              position leaves one side it will enter the
                              opposing one.
  -a [ --sigma-accel ] arg    Standard deviation of normally-distributed random
                              accelerations
```

#### Example

```bash
# Publish randomly moving positions to the 'pos' position stream
oat posigen rand2D pos
```

### Position Filter

`oat-posifilt` - Receive positions from named shared memory, filter, and publish to a second memory segment. Can be used to, for example, remove discontinuities due to noise or discontinuities in position detection with a Kalman filter or annotate categorical position information based on user supplied region contours.

#### Signature

```
position --> oat-posifilt --> position
```

#### Usage

```
Usage: posifilt [INFO]
   or: posifilt TYPE SOURCE SINK [CONFIGURATION]
Filter positions from SOURCE and publish filtered positions to SINK.

INFO:
  --help                 Produce help message.
  -v [ --version ]       Print version information.

TYPE
  kalman: Kalman filter
  homography: homography transform
  region: position region annotation

SOURCE:
  User-supplied name of the memory segment to receive positions from (e.g. pos).

SINK:
  User-supplied name of the memory segment to publish positions to (e.g. filt).
```

#### Configuration Options

**TYPE = `kalman`**

```
  --dt arg                   Kalman filter time step in seconds.
  -T [ --timeout ] arg       Seconds to perform position estimation detection
                             with lack of position measure. Defaults to 0.
  -a [ --sigma-accel ] arg   Standard deviation of normally distributed, random
                             accelerations used by the internal model of object
                             motion (position units/s2; e.g. pixels/s2).
  -n [ --sigma-noise ] arg   Standard deviation of randomly distributed
                             position measurement noise (position units; e.g.
                             pixels).
```

```
  -t [ --tune ]              If true, provide a GUI with sliders for tuning
                             filter parameters.
```

**TYPE = homography**

```
  -H [ --homography ] arg   A nine-element array of floats, [h11,h12,...,h33],
                            specifying a 3x3 homography matrix for 2D position.
                            Can be produced by oat-calibrate homography.
```

**TYPE = region**

```
  --regions arg           NOTE: Regions can only be specified in a config file.
                          Regions contours are specified as n-point matrices,
                          [[x0, y0],[x1, y1],...,[xn, yn]], which define the
                          vertices of a polygon:

                            <region> = [[+float, +float],
                                        [+float, +float],
                                        ...
                                        [+float, +float]]

                          The name of the contour is used as the region label
                          (10 characters max). For example, here is an
                          octagonal region called CN and a tetragonal region
                          called R0:

                            CN = [[336.00, 272.50],
                                  [290.00, 310.00],
                                  [289.00, 369.50],
                                  [332.67, 417.33],
                                  [389.33, 413.33],
                                  [430.00, 375.33],
                                  [433.33, 319.33],
                                  [395.00, 272.00]]

                            R0 = [[654.00, 380.00],
                                  [717.33, 386.67],
                                  [714.00, 316.67],
                                  [655.33, 319.33]]
```

**Example**

```
# Perform Kalman filtering on object position from the 'pos' position stream
# publish the result to the 'kpos' position stream
# Use detector settings supplied by the kalman_config key in config.toml
```

```
oat posifilt kalman pos kfilt -c config.toml kalman_config
```

### Position Combiner

`oat-posicom` - Combine positions according to a specified operation.

#### Signature

```
position 0 --> |
position 1 --> |
  :            | oat-posicom --> position
position N --> |
```

#### Usage

```
Usage: posicom [INFO]
   or: posicom TYPE SOURCES SINK [CONFIGURATION]
Combine positional information from two or more SOURCES and Publish combined position to SIN

INFO:
  --help                 Produce help message.
  -v [ --version ]       Print version information.

TYPE
  mean: Geometric mean of positions

SOURCES:
  User-supplied position source names (e.g. pos1 pos2).

SINK:
  User-supplied position sink name (e.g. pos).
```

#### Configuration Options

**TYPE = `mean`**

```
  -h [ --heading-anchor ] arg   Index of the SOURCE position to use as an
                                anchor when calculating object heading. In this
                                case the heading equals the mean directional
                                vector between this anchor position and all
                                other SOURCE positions. If unspecified, the
                                heading is not calculated.
```

#### Example

```
# Generate the geometric mean of 'pos1' and 'pos2' streams
# Publish the result to the 'com' stream
oat posicom mean pos1 pos2 com
```

#### Frame Decorator

`oat-decorate` - Annotate frames with sample times, dates, and/or positional information.

##### Signature

```
     frame --> |
position 0 --> |
position 1 --> | oat-decorate --> frame
  :            |
position N --> |
```

##### Usage

```
Usage: decorate [INFO]
   or: decorate SOURCE SINK [CONFIGURATION]
Decorate the frames from SOURCE, e.g. with object position markers and sample number. Publis

SOURCE:
  User-supplied name of the memory segment from which frames are received (e.g. raw).

SINK:
  User-supplied name of the memory segment to publish frames to (e.g. out).

INFO:
  --help                          Produce help message.
  -v [ --version ]                Print version information.

CONFIGURATION:
  -c [ --config ] arg             Configuration file/key pair.
                                  e.g. 'config.toml mykey'
  --control-endpoint arg          ZMQ style endpoint specifier designating
                                  runtime control port:'<transport>://<host>:<p
                                  ort>'. For instance, 'tcp://*:5555' to
                                  specify TCP communication on port 5555. Or,
                                  for interprocess communication:
                                  '<transport>://<user-named-pipe>. For
                                  instance 'ipc:///tmp/test.pipe'. Internally,
                                  this is used to construct a ZMQ REQ socket
                                  that that receives commands from oat-control.
```

```
                               Defaults to ipc:///tmp/oatcomms.pipe.

  -p [ --position-sources ] arg   The name of position SOURCE(s) used to draw
                                  object position markers.

  -t [ --timestamp ]              Write the current date and time on each
                                  frame.

  -s [ --sample ]                 Write the frame sample number on each frame.

  -S [ --sample-code ]            Write the binary encoded sample on the corner
                                  of each frame.

  -R [ --region ]                 Write region information on each frame if
                                  there is a position stream that contains it.

  -h [ --history ]                Display position history.

  -i [ --invert-font ]            Invert font color.
```

#### Example

```
# Add textual sample number to each frame from the 'raw' stream
oat decorate raw -s

# Add position markers to each frame from the 'raw' stream to indicate
# objection positions for the 'pos1' and 'pos2' streams
oat decorate raw -p pos1 pos2
```

### Recorder

`oat-record` - Save frame and position streams to file.

- `frame` streams are compressed and saved as individual video files ( H.264 compression format AVI file).
- `position` streams saved to separate JSON file. Optionally, they can be saved to numpy binary files. JSON position files have the following structure:

```
{
    oat-version: X.X,
    header: {
        timestamp: YYYY-MM-DD-hh-mm-ss,
        sample_rate_hz: X.X
    },
    positions: [
        position,
        position,
        ...,
        position
    ]
}
```

where each `position` object is defined as:

```
{
  tick: Int,                   | Sample number
  usec: Int,                   | Microseconds associated with current sample number
  unit: Int,                   | Enum specifying length units (0=pixels, 1=meters)
  pos_ok: Bool,                | Boolean indicating if position is valid
  pos_xy: [Double, Double],    | Position x,y values
  vel_ok: Bool,                | Boolean indicating if velocity is valid
  vel_xy: [Double, Double],    | Velocity x,y values
  head_ok: Bool,               | Boolean indicating if heading  is valid
  head_xy: [Double, Double],   | Heading x,y values
  reg_ok: Bool,                | Boolean indicating if region tag  is valid
  reg: String                  | Region tag
}
```

When using JSON and the `consise-file` option is specified, data fields are only populated if the values are valid. For instance, in the case that only object position is valid, and the object velocity, heading, and region information are not calculated, an example position data point would look like this:

```
{ tick: 501,
  usec: 50100000,
  unit: 0,
```

```
  pos_ok: True,
  pos_xy: [300.0, 100.0],
  vel_ok: False,
  head_ok: False,
  reg_ok: False }
```

When using binary file format, position entries occupy single elements of a numpy structured array with the following `dtype`:

```
[('tick', '<u8'),
 ('usec', '<u8'),
 ('unit', '<i4'),
 ('pos_ok', 'i1'),
 ('pos_xy', '<f8', (2,)),
 ('vel_ok', 'i1'),
 ('vel_xy', '<f8', (2,)),
 ('head_ok', 'i1'),
 ('head_xy', '<f8', (2,)),
 ('reg_ok', 'i1'),
 ('reg', 'S10')]
```

Multiple recorders can be used in parallel to (1) parallelize the computational load of video compression, which tends to be quite intense and (2) save to multiple locations simultaneously (3) to save the same data stream multiple times in different formats.

#### Signature

```
position 0 --> |
position 1 --> |
  :            |
position N --> |
               | oat-record
   frame 0 --> |
   frame 1 --> |
     :         |
   frame N --> |
```

#### Usage

```
Usage: record [INFO]
   or: record [CONFIGURATION]
Record any Oat token source(s).

INFO:
  --help                          Produce help message.
  -v [ --version ]                Print version information.
```

```
CONFIGURATION:
  -c [ --config ] arg            Configuration file/key pair.
                                 e.g. 'config.toml mykey'
  --control-endpoint arg         ZMQ style endpoint specifier designating
                                 runtime control port:'<transport>://<host>:<p
                                 ort>'. For instance, 'tcp://*:5555' to
                                 specify TCP communication on port 5555. Or,
                                 for interprocess communication:
                                 '<transport>://<user-named-pipe>. For
                                 instance 'ipc:///tmp/test.pipe'. Internally,
                                 this is used to construct a ZMQ REQ socket
                                 that that receives commands from oat-control.
                                 Defaults to ipc:///tmp/oatcomms.pipe.

  -s [ --frame-sources ] arg     The names of the FRAME SOURCES that supply
                                 images to save to video.
  -p [ --position-sources ] arg  The names of the POSITION SOURCES that supply
                                 object positions to be recorded.
  -n [ --filename ] arg          The base file name. If not specified,
                                 defaults to the SOURCE name.
  -f [ --folder ] arg            The path to the folder to which the video
                                 stream and position data will be saved. If
                                 not specified, defaults to the current
                                 directory.
  -d [ --date ]                  If specified, YYYY-MM-DD-hh-mm-ss_ will be
                                 prepended to the filename.
  -o [ --allow-overwrite ]       If set and save path matches and existing
                                 file, the file will be overwritten instead of
                                 a incremental numerical index being appended
                                 to the file name.
  -F [ --fourcc ] arg            Four character code (https://en.wikipedia.org
                                 /wiki/FourCC) used to specify the codec used
                                 for AVI video compression. Must be specified
                                 as a 4-character string (see
                                 http://www.fourcc.org/codecs.php for possible
                                 options). Not all valid FOURCC codes will
                                 work: it must be implemented by the low
                                 level writer. Common values are 'DIVX' or
                                 'H264'. Defaults to 'None' indicating
                                 uncompressed video.
  -b [ --binary-file ]           Position data will be written as numpy data
                                 file (version 1.0) instead of JSON. Each
                                 position data point occupies a single entry
                                 in a structured numpy array. Individual
                                 position characteristics are described in the
```

```
                                  arrays dtype.
  -c [ --concise-file ]           If set and using JSON file format,
                                  indeterminate position data fields will not
                                  be written e.g. pos_xy will not be written
                                  even when pos_ok = false. This means that
                                  position objects will be of variable size
                                  depending on the validity of whether a
                                  position was detected or not, potentially
                                  complicating file parsing.
```

**Example**

```bash
# Save positional stream 'pos' to current directory
oat record -p pos

# Save positional stream 'pos1' and 'pos2' to current directory
oat record -p pos1 pos2

# Save positional stream 'pos1' and 'pos2' to Desktop directory and
# prepend the timestamp to the file name
oat record -p pos1 pos2 -d -f ~/Desktop

# Save frame stream 'raw' to current directory
oat record -s raw

# Save frame stream 'raw' and positional stream 'pos' to Desktop
# directory and prepend the timestamp and the word 'test' to each filename
oat record -s raw -p pos -d -f ~/Desktop -n test

# Save the pos stream twice, one binary and one JSON file, in the current
# directory
oat record -p pos &
oat record -p pos -b
```

### Position Socket

`oat-posisock` - Stream detected object positions to the network in either client or server configurations.

#### Signature

```
position --> oat-posisock
```

#### Usage

```
Usage: posisock [INFO]
   or: posisock TYPE SOURCE [CONFIGURATION]
Send positions from SOURCE to a remote endpoint.

INFO:
  --help                 Produce help message.
  -v [ --version ]       Print version information.

TYPE:
  std: Asynchronous position dump to stdout.
  pub: Asynchronous position publisher over ZMQ socket.
       Publishes positions without request to potentially many
       subscribers.
  rep: Synchronous position replier over ZMQ socket.
       Sends positions in response to requests from a single
       endpoint.Several transport/protocol options. The most
       useful are tcp and interprocess (ipc).
  udp: Asynchronous, client-side, unicast user datagram protocol
       over a traditional BSD-style socket.

SOURCE:
  User-supplied name of the memory segment to receive positions from (e.g. pos).
```

#### Configuration Options

**TYPE = `std`**

```
  -p [ --pretty-print ]    If true, print formated positions to the command
                           line.
```

**TYPE = `pub`**

```
  -e [ --endpoint ] arg   ZMQ-style endpoint. For TCP:
                          '<transport>://<host>:<port>'. For instance,
```

```
                          'tcp://*:5555'. Or, for interprocess communication:
                          '<transport>:///<user-named-pipe>. For instance
                          'ipc:///tmp/test.pipe'.
```

**TYPE = `rep`**

```
  -e [ --endpoint ] arg   ZMQ-style endpoint. For TCP:
                          '<transport>://<host>:<port>'. For instance,
                          'tcp://*:5555'. Or, for interprocess communication:
                          '<transport>:///<user-named-pipe>. For instance
                          'ipc:///tmp/test.pipe'.
```

**type = `udp`**

```
  -h [ --host ] arg       Host IP address of remote device to send positions
                          to. For instance, '10.0.0.1'.
  -p [ --port ] arg       Port number of endpoint on remote device to send
                          positions to. For instance, 5555.
```

#### Example

```bash
# Reply to requests for positions from the 'pos' stream to port 5555 using TCP
oat posisock rep pos -e tcp://*:5555

# Asychronously publish positions from the 'pos' stream to port 5556 using TCP
oat posisock pub pos -e tcp://*:5556

# Dump positions from the 'pos' stream to stdout
oat posisock std pos
```

### Buffer

`oat-buffer` - A first in, first out (FIFO) token buffer that can be use to decouple asynchronous portions of a data processing network. An example of this is the case when a precise external clock is used to govern image acquisition via a physical trigger line. In this case, 'hickups' in the data processing network following the camera should not cause the camera to skip frames. Of course, there is no free lunch: if the processing pipline cannot keep up with the external clock on average, then the buffer will eventually fill and overflow.

#### Signatures

```
position --> oat-buffer --> position

frame --> oat-buffer --> frame
```

#### Usage

```
Usage: buffer [INFO]
   or: buffer TYPE SOURCE SINK [CONFIGURATION]
Place tokens from SOURCE into a FIFO. Publish tokens in FIFO to SINK.

INFO:
  --help                 Produce help message.
  -v [ --version ]       Print version information.

TYPE
  frame: Frame buffer
  pose: Pose buffer

SOURCE:
  User-supplied name of the memory segment to receive tokens from (e.g. input).

SINK:
  User-supplied name of the memory segment to publish tokens to (e.g. output).
```

#### Example

```
# Acquire frames on a gige camera driven by an exnternal trigger
oat frameserve gige raw -c config.toml gige-trig

# Buffer the frames separate asychronous sections of the processing network
oat buffer frame raw buff

# Filter the buffered frames and save
```

```
oat framefilt mog buff filt
oat record -f ~/Desktop/ -p buff filt
```

In the above example, one must be careful to fully separate the network across the buffer boundary in order for it to provide any functionality. For instance, if we changed the record command to the following

```
oat record -f ~/Desktop/ -p raw filt
```

Then the buffer would do nothing since the raw token stream must be synchronous with the recorder, which bypasses the buffer. In this case, the buffer is just wasting CPU cycles. Here is a graphical representation of the first configuration where the `oat-buffer` is used properly. The synchronization boundary is shown using vertical lines.

```
               |
frameserve --> buffer --> framefilt --> record
               |    \                  /
               |     ------------------
```

In the second configuration, the connection from frameserve to record breaks the synchronization boundary.

```
               |
frameserve --> buffer --> framefilt --> record
          \    |                      /
           ----|----------------------
```

### Calibrate

`oat-calibrate` - Interactive program used to generate calibration parameters for an imaging system that can be used to parameterize `oat-framefilt` and `oat-posifilt`. Detailed usage instructions are displayed upon program startup.

#### Signature

```
frame --> oat-calibrate
```

#### Usage

```
Usage: calibrate [INFO]
   or: calibrate TYPE SOURCE [CONFIGURATION]
Camera calibration and homography generation routines.

INFO:
  --help                 Produce help message.
  -v [ --version ]       Print version information.

TYPE
  camera: Generate calibration parameters (camera matrix and distortion coefficients).
  homography: Generate homography transform between pixels and world units.

SOURCE:
  User-supplied name of the memory segment to receive frames from (e.g. raw).
```

#### Configuration Options

**TYPE = `camera`**

```
  -k [ --calibration-key ] arg    The key name for the calibration entry that
                                  will be inserted into the calibration file.
                                  e.g. 'camera-1-homography'

  -f [ --calibration-path ] arg   The calibration file location. If not is
                                  specified,defaults to './calibration.toml'.
                                  If a folder is specified, defaults to
                                  '<folder>/calibration.toml
                                  . If a full path including file in specified,
                                  then it will be that path without
                                  modification.
  -s [ --chessboard-size ] arg    Int array, [x,y], specifying the number of
                                  inside corners in the horizontal and vertical
                                  demensions of the chessboard used for
```

```
                                    calibration.

  -w [ --square-width ] arg         The length/width of a single chessboard
                                    square in meters.
```

**TYPE** = **`homography`**

```
  -k [ --calibration-key ] arg      The key name for the calibration entry that
                                    will be inserted into the calibration file.
                                    e.g. 'camera-1-homography'

  -f [ --calibration-path ] arg     The calibration file location. If not is
                                    specified,defaults to './calibration.toml'.
                                    If a folder is specified, defaults to
                                    '<folder>/calibration.toml
                                    . If a full path including file in specified,
                                    then it will be that path without
                                    modification.
  -m [ --method ] arg               Homography estimation method. Defaults to 0.

                                    Values:
                                      0: RANSAC-based robust estimation method
                                         (automatic outlier rejection).
                                      1: Best-fit using all data points.
                                      2: Compute the homography that fits four
                                         points. Useful when frames contain known
                                         fiducial marks.
```

### Kill

`oat-kill` - Issue SIGINT to all running Oat processes started by the calling user. A side effect of Oat's architecture is that components can become orphaned in certain circumstances: abnormal termination of attached sources or sinks, running pure sources in the background and forgetting about them, etc. This utility will gracefully interrupt all currently running oat components.

#### Usage

```
Usage: kill
```

#### Example

```
# Interupt all currently running oat components
oat kill
```

### Clean

`oat-clean` - Programmer's utility for cleaning shared memory segments after following abnormal component termination. Not required unless a program terminates without cleaning up shared memory. If you are using this for things other than development, then please submit a bug report.

#### Usage

```
Usage: clean [INFO]
   or: clean NAMES [CONFIGURATION]
Deallocate the named shared memory segments specified by NAMES.

INFO:
  --help                Produce help message.
  -v [ --version ]      Print version information.
  -q [ --quiet ]        Quiet mode. Prevent output text.
  -l [ --legacy ]       Legacy mode. Append  "_sh_mem" to input NAMES before
                        removing.
```

#### Example

```
# Remove raw and filt blocks from shared memory after abnormal terminatiot of
# some components that created them
oat clean raw filt
```

## Installation

First, ensure that you have installed all dependencies required for the components and build configuration you are interested in in using. For more information on dependencies, see the dependencies section below. To compile and install Oat, starting in the top project directory, create a build directory, navigate to it, and run cmake on the top-level CMakeLists.txt like so:

```
mkdir release
cd release
cmake -DCMAKE_BUILD_TYPE=Release [CMAKE OPTIONS] ..
make
make install
```

If you just want to build a single component component, individual components can be built using `make [component-name]`, e.g. `make oat-view`. Available cmake options and their default values are:

```
-DUSE_FLYCAP=Off // Compile with support for Point Grey Cameras
-DBUILD_DOCS=Off     // Generate Doxygen documentation
```

If you had to install Boost from source, you must let cmake know where it is installed via the following switch. Obviously, provide the correct path to the installation on your system.

```
-DBOOST_ROOT=/opt/boost_1_59_0
```

To complete installation, add the following to your `.bashrc` or equivalent. This makes Oat commands available within your user profile (once you start a new terminal):

```
# Make Oat commands available to user
eval "$(<path/to/Oat>/oat/bin/oat init -)"
```

If you get runtime link errors when you try to run an Oat program such as >error while loading shared libraries: libboost_program_options.so.1.60.0 then you need to ad the following entry to your `.bashrc`

```
export LD_LIBRARY_PATH=</path/to/boost>/stage/lib:$LD_LIBRARY_PATH
```

### Dependencies

### License compatibility

Oat is licensed under the GPLv3.0. Its dependences' are licenses are shown below:

- Eigen3: MPL
- Flycapture SDK: NON-FREE specialized license (This is an optional package. If you compile without Flycapture support, you can get around

this. Also, see the `GigE interface cleanup` entry in the TODO section for a potentially free alternative.)
- OpenCV: BSD
- ZeroMQ: LGPLv3.0
- Boost: Boost software license
- cpptoml: Some kind of Public Domain Dedication
- RapidJSON: BSD
- Catch: Boost software license

These licenses do not violate the terms of Oat's license. If you feel otherwise please submit an bug report.

#### Eigen

Eigen is a template library for linear algebra that is required by some Oat component types:

- `oat-posidet rpg`

To build these component sub-types, you must install Eigen. Eigen can be obtained either from its source repoor via snapshot download here.

#### Flycapture SDK

The FlyCapture SDK is used to communicate with Point Grey digital cameras. It is not required to compile any Oat components. However, the Flycapture SDK is required if a Point Grey camera is to be to be used with the `oat-frameserve` component to acquire images. If you simply want to process pre-recorded files or use a web cam, e.g. via

```
oat-frameserve file raw -f video.mpg
oat-frameserve wcam raw
```

then this library is *not* required.

To install the Point Grey SDK:

- Go to point-grey website
- Download the FlyCapture2 SDK (version >= 2.7.3). Annoyingly, this requires you to create an account with Point Grey.
- Extract the archive and use the `install_flycapture.sh` script to install the SDK on your computer and run

```
tar xf flycapture.tar.gz
cd flycapture
sudo ./install_flycapture
```

### Boost

The Boost libraries are required to compile all Oat components. You will need to install versions >= 1.56. To install Boost, use APT or equivalent,

```
sudo apt-get install libboost-all-dev
```

If you are using an Ubuntu distribution older than Wily Werewolf, Boost will be too old and you will need to install from source via

```
# Install latest boost
wget http://sourceforge.net/projects/boost/files/latest/download?source=files -O tarboost
tar -xf tarboost
cd ./boost*
./bootstrap.sh
./b2 --with-program_options --with-system --with-thread --with-filesystem
cd ..
sudo mv boost* /opt
```

Finally, if you are getting runtime linking errors, you will need to place the following in `.bashrc`

```
export LD_LIBRARY_PATH=<path to boost root directory>/stage/lib:$LD_LIBRARY_PATH
```

### OpenCV

opencv is required to compile the following oat components:

- `oat-frameserve`
- `oat-framefilt`
- `oat-view`
- `oat-record`
- `oat-posidet`
- `oat-posifilt`
- `oat-decorate`
- `oat-positest`

### ffmpeg Support

**Note**: OpenCV must be installed with ffmpeg support in order for offline analysis of pre-recorded videos to occur at arbitrary frame rates. If it is not, gstreamer will be used to serve from video files at the rate the files were recorded. No cmake flags are required to configure the build to use ffmpeg. OpenCV will be built with ffmpeg support if something like

```
-- FFMPEG:          YES
-- codec:           YES (ver 54.35.0)
-- format:          YES (ver 54.20.4)
-- util:            YES (ver 52.3.0)
```

```
-- swscale:         YES (ver 2.1.1)
```

appears in the cmake output text. The dependencies required to compile OpenCV with ffmpeg support, can be obtained as follows:

```
TODO
```

#### opencv_contrib Support

To use `oat-posidet arcuo`, you must compile Oat against OpenCV that has `contrib` support. To compile OpenCV with `contrib` support, clone down the opecv_contrib repo. Then, when configuring the OpenCV build, set the path the contrib modules path as follows: using `-DOPENCV_EXTRA_MODULES_PATH=<opencv_contrib>/modules`

#### OpenGL Support

To increase Oat's video visualization performance using `oat view`, you can build OpenCV with OpenGL and/or OpenCL support. Both will open up significant processing bandwidth to other Oat components and make for faster processing pipelines. To compile OpenCV with OpenGL and OpenCL support, first install dependencies:

```
sudo apt-get install libgtkglext1 libgtkglext1-dev
```

Then, add the `-DWITH_OPENGL=ON` and the `-DWITH_OPENCL=ON` flags to the cmake command below. OpenCV will be build with OpenGL and OpenCL support if `OpenGL support: YES` and `Use OpenCL: YES` appear in the cmake output text. If OpenCV is compiled with OpenCL and OpenGL support, the performance benefits will be automatic, no compiler options need to be set for Oat.

#### CUDA Support

If you have NVIDIA GPU that supports CUDA, you can build OpenCV with CUDA support to enable GPU accelerated video processing. To do this, will first need to install the CUDA toolkit.

- Be sure to **carefully** read the installation instructions since it is a multistep process. Here are some additional hints that worked for me:
- I have found that installing the toolkit via 'runfile' to be the most painless. To do this you will need to switch your system to text mode using Ctrl + Alt + F1, and killing the X-server via `sudo service lightdm stop` (or equivalent), and running the runfile with root privileges.
- I have had the most success on systems that do not use GNOME or other fancy desktop environments. The install on lubunut, which uses LXDE as its desktop environment, was especially smooth.

- Do **not** install the nvidia drivers along with the CUDA toolkit installation. I found that (using ubuntu 14.04) this causes all sorts of issues with X, cinnamon, etc, to the point where I could not even boot my computer into anything but text mode. Instead, install the NVIDIA drivers using either the package manager (`nvidia-current`) or even more preferably, using the [`device-drivers`]'(http://askubuntu.com/a/476659) program or equivalent.
- If you hare getting a `cv::exception` complaining that about `code=30(cudaErrorUnknown) "cudaGetDeviceCount(&device_count)"` or similar, run the affected command as root one time.

If OpenCV is compiled with CUDA suport, the CUDA-enabled portions of the Oat codebase will be enabled automatically. No compile flags are required.

#### QT Support

GUI functionality is enhanced in OpenCV is compiled with Qt support. You can build OpenCV with Qt by first installing the Qt SDK and these dependencies:

```
# Additional dependencies for integraged QT with OpenGL
sudo apt-get install libqt5opengl5 libqt5opengl5-dev
```

The you can compile OpenCV using QT support by adding `-DWITH_QT=ON` flag to the cmake command below. QT functionality will then be used by Oat automatically.

#### OpenCV Build

Finally, to compile and install OpenCV:

```
# Install OpenCV's dependencies
sudo apt-get install build-essential # Compiler
sudo apt-get install cmake git # For building opencv and Oat
sudo apt-get install libgtk2.0-dev pkg-config libavcodec-dev libavformat-dev libswscale-dev
sudo apt-get install libv4l-dev # Allows changing frame rate with webcams
sudo apt-get install python-dev python-numpy libtbb2 libtbb-dev libjpeg-dev libpng-dev libti
sudo apt-get install # ffmpeg support [TODO]
sudo apt-get install # OpenGL support [TODO]
sudo ldconfig -v

# Get OpenCV
wget https://github.com/Itseez/opencv/archive/3.1.0.zip -O opencv.zip
unzip opencv.zip -d opencv

# Build OpenCV
cd opencv/opencv-3.0.0-rc1
mkdir release
```

```
cd release

# Run cmake to generate Makefile
# Add -DWITH_CUDA=ON for CUDA support and -DWITH_OPENGL for OpenGL support
cmake -DWITH_LIBV4L=ON -DCMAKE_BUILD_TYPE=RELEASE -DCMAKE_INSTALL_PREFIX=/usr/local ..

# Build the project and install
make
sudo make install
```

### ZeroMQ

ZeroMQ is required by the following Oat components:

- `oat-record`
- `oat-posisock`

Download, compile, and install ZeroMQ as follows:

```
wget http://download.zeromq.org/zeromq-4.1.4.tar.gz -O tarzmq
tar -xf tarzmq
cd ./zeromq*
./configure --without-libsodium
make
sudo make install
sudo ldconfig
```

Additionally, you will need to download the ZeroMQ C++ binding (this is just a single header file) and place it somewhere that your compiler will find it.

```
wget https://raw.githubusercontent.com/zeromq/cppzmq/master/zmq.hpp
sudo mv zmq.hpp /usr/local/include/
```

### RapidJSON, cpptoml, and Catch

These libraries are installed automatically by cmake during the build process.

RapidJSON is required by the following Oat components:

- `oat-record`
- `oat-posisock`

cpptoml is required by the following Oat components:

- `oat-frameserve`
- `oat-framefilt`
- `oat-posidet`
- `oat-posifilt`
- `oat-posicom`
- `oat-positest`

Catch is required to make and run tests using `make test`

## Performance

Oat is designed for use in real-time video processing scenarios. This boils down the following definition

> The average execution time for an Oat dataflow network must not exceed the camera(s) image transfer period

If this condition is not met, then frames will eventually be dropped. There is no way around this. The guts of Oat consist of a simple, but very efficient message passing library that links together processing routines taken from a variety of sources (some written by me, some by third party projects such as OpenCV). The speed of each processing step is determined both by its computational complexity and deftness of implementation, both of which can vary quite a lot for different components. To see some rudimentary performance numbers for Oat components in isolation, have a look at these numbers. There is definitely room for optimization for some components. And, several components that are ripe for GPU implementation do not have one yet. This comes down to free time. If anyone wants to try there hand at making some of the bottleneck components faster, please get in touch.

Outside of code optimization, there are a few things a user should be aware of to make efficient use of Oat, which are listed below.

#### Frames are slow

The first thing to know is that working with `frames` is orders of magnitude slower than working with `positions`. Therefore, minimizing the number of processing steps operating on `frames` is a good way to reduce computational requirements. Processing on `positions` is in the noise in comparison.

#### Parallelism

Increasing the number of components in your chain does not necessarily cause an appreciable an increase in processing time because Oat components run in parallel. Instead, up to the limit of the number of hyperthreads/GPU resources your computer supports, the slowest component in a dataflow network will largely determine the speed of the processing rather than the number of components within the processing network.

### Resolution

Do you really need that 10 MP camera? Recall that increases in sensor resolution cause a power 2 increase in then number of pixels you need to smash into RAM, process, write to disk, and, probably, post process. Its really best to use the lowest resolution camera that suites your needs, both for the sake of real-time processing in Oat and your future sanity when trying to deal with those 30 GB video files.

### Hard-disk

If you are saving video, then the write speed of your hard disk can become the limiting factor in a processing network. To elaborate, I'm just quoting my response to this issue:

> **Q:** I also ran into an issue with RAM and encoding. I have 8 GB, and they fill up within about 20 seconds, then goes into swap.
>
> **A:** I suspect the following is the issue:
>
> (22 FPS * 5 MP * 24 bits/pixel) / ( 8 bits/ byte) = 330 MB/sec
>
> This (minus compression, which I'm admittedly ignoring, but is probably made up for by the time it takes to do the compression...) is the requisite write speed (in actuality, not theoretically) of your hard disk in order not to get memory overflow.
>
> 8 GB / 0.330 GB =~ 24 seconds.
>
> The RAM is filling because your hard disk writes are not occurring fast enough. Oat is pushing frames to be written into a FIFO in main memory that the recorder thread is desperately trying to write to disk. Getting more RAM will just make the process persist for a bit longer before failing. I would get an SSD for streaming video to and then transfer those videos to a slower long term storage after recording.

## Setting up a Point-grey PGE camera in Linux

`oat-frameserve` supports using Point Grey GIGE cameras to collect frames. I found the setup process to be straightforward and robust, but only after cobbling together the following notes.

### Camera IP Address Configuration

First, assign your camera a static IP address. The easiest way to do this is to use a Windows machine to run the Point Grey 'GigE Configurator'. If someone knows a way to do this without Windows, please tell me. An example IP Configuration might be:

- Camera IP: 192.168.0.1
- Subnet mask: 255.255.255.0
- Default gateway: 192.168.0.64

### Point Grey GigE Host Adapter Card Configuration

Using network manager or something similar, you must configure the IPv4 configuration of the GigE host adapter card you are using to interface the camera with your computer.

- First, set the ipv4 method to **manual**.
- Next, you must configure the interface to (1) have the same network prefix and (2) be on the same subnet as the camera you setup in the previous section.
  - Assuming you used the camera IP configuration specified above, your host adapter card should be assigned the following private IPv4 configuration:
    * POE gigabit card IP: 192.168.0.100
    * Subnet mask: 255.255.255.0
    * DNS server IP: 192.168.0.1
- Next, you must enable jumbo frames on the network interface. Assuming that the camera is using `eth2`, then entering

  ```
  sudo ifconfig eth2 mtu 9000
  ```

into the terminal will enable 9000 MB frames for the `eth2` adapter. - Finally, to prevent image tearing, you should increase the amount of memory Linux uses for network receive buffers using the `sysctl` interface by typing

```
sudo sysctl -w net.core.rmem_max=1048576 net.core.rmem_default=1048576
```

into the terminal. *In order for these changes to persist after system reboots, the following lines must be added to the bottom of the `/etc/sysctl.conf` file*:

```
net.core.rmem_max=1048576
net.core.rmem_default=1048576
```

These settings can then be reloaded after reboot using

```
sudo sysctl -p
```

### Multiple Cameras

- If you have two or more cameras/host adapter cards, they can be configured as above but *must exist on a separate subnets*. For instance, we could repeat the above configuration steps for a second camera/host adapter card using the following settings:
  - Camera Configuration:
    * Camera IP: 192.168.**1**.1
    * Subnet mask: 255.255.255.0
    * Default gateway: 192.168.**1**.64
  - Host adapter configuration:
    * POE gigabit card IP: 192.168.**1**.100
    * Subnet mask: 255.255.255.0
    * DNS server IP: 192.168.**1**.1

### Example Camera Configuration

Below is an example network adapter and camera configuration for a two-camera imaging system provided by Point Grey. It consists of two Blackfly GigE cameras (Point Grey part number: BFLY-PGE-09S2C) and a single dual-port POE GigE adapter card (Point Grey part number: GIGE-PCIE2-2P02).

### Camera 0

- Adapter physical connection (looking at back of computer)

  ```
  RJ45 ------------
          |       |
  L [   [ ]     [x]   ] R
  ```

- Adapter Settings
  - Model: Intel 82574L Gigabit Network Connection
  - MAC: 00:B0:9D:DB:D9:63
  - MTU: 9000
  - DHCP: Disabled
  - IP: 192.168.0.100
  - Subnet mask: 255.255.255.0

- Camera Settings
  - Model: Blackfly BFLY-PGE-09S2C
  - Serial No.: 14395177
  - IP: 192.168.0.1 (Static)
  - Subnet mask: 255.255.255.0
  - Default GW: 0.0.0.0
  - Persistent IP: Yes

**Camera 1**

- Adapter physical connection (looking at back of computer)

```
RJ45 ------------
      |      |
L [  [x]    [ ]  ] R
```

- Adapter Settings
  - Model: Intel 82574L Gigabit Network Connection
  - MAC: 00:B0:9D:DB:A7:29
  - MTU: 9000
  - DHCP: Disabled
  - IP: 192.168.1.100
  - Subnet mask: 255.255.255.0
- Camera Settings
  - Model: Blackfly BFLY-PGE-09S2C
  - Serial No.:
  - IP: 192.168.1.1 (Static)
  - Subnet mask: 255.255.255.0
  - Default GW: 0.0.0.0
  - Persistent IP: Yes

## TODO

- [ ] Unit and stress testing
  - Unit tests for `libshmemdf`
    - ~~Nominal data types, `T`~~
    - Specializations for `Frames`
  - Stress tests for data processing chains
    - I need to come up with a series of scripts that configure and run components in odd and intensive, but legal, ways to ensure sample sychronization is maintained, graceful exits, etc
- [ ] Position type correction and generalization to 3D pose
  - ~~It might be a good idea to generalize the concept of a position to a multi-positional element~~
  - ~~For things like the `oat-decorate`, `oat-posicom`, and potentially `oat-detect`, this could increase performance and decrease user script complexity if multiple targets common detection features needed to be tracked at once.~~
  - ~~Down side is that it potentially increases code complexity and would require a significant refactor.~~
  - ~~Additionally, position detection might no longer be stateless. E.g. think of the case when two detected objects cross paths. In order to ID the objects correctly in subsequent detections, the path of the objects would need to be taken into account (and there is not guarantee this result will be correct…). A potential work around is to have IDed 'position groups' with annoymous position members. This would get us back to stateless detection. However, it would make the concept of position combining hard to define (although that is even true now is just a design choice, really).~~
  - EDIT: Additionally, there should certainly not be `Position2D` vs `Position3D`. Only `Position` which provides 3d specificaiton with Z axis defaulting to 0.
  - EDIT: In fact, positions should simply be generalize two a 3D pose. I've started a branch to do this.
- [ ] `oat-framefilt undistort`
  - Very slow. Needs an OpenGL or CUDA implementation
  - User supplied frame rotation occurs in a separate step from undistortion. Very inefficient. Should be able to combine rotation with camera matrix to make this a lot faster.
  - EDIT: Also should provide an `oat-posifilt` version which only applies undistortion to position rather than the entire frame.
- [ ] Type deduction in shmem Tokens
  - Sources should have a static method for checking the token type of a given address.