Zoloto

Release 0.9.0

Jake Howard

Nov 30, 2022

CONTENTS:

A fiducial marker system powered by OpenCV - Supports ArUco and April

Documentation

CHAPTER

ONE

INSTALLATION

```
pip install zoloto
```

1.1 OpenCV

OpenCV should be installed manually, ideally through your system package manager. This makes it easier to customize your OpenCV installation for your system, or use the optimal settings for your OS / hardware. Note that you may need to install `opencv-contrib` as well as `opencv`.

If you'd rather have one installed automatically, install the extra `opencv`:

```
pip install zoloto[opencv]
```

Note that this version lacks hardware acceleration. See the README for more details.

For storage-constrained environments, there's also `opencv-contrib-python-headless`, which should be installed manually.

CHAPTER

TWO

EXAMPLES

```
from pathlib import Path

from zoloto import MarkerType
from zoloto.cameras import ImageFileCamera


with ImageFileCamera(Path("my-image.png"), marker_type=MarkerType.ARUCO_6X6) as camera:
    camera.save_frame("my-annotated-image.png", annotate=True)
    print("I saved an image with {} markers in.".format(len(camera.get_visible_
↪markers())))
```

More examples

Zoloto ships with a CLI (aptly named `zoloto`), which contains some helpful utils for working with Zoloto and fiducial markers.

CHAPTER

THREE

DEVELOPMENT SETUP

`./scripts/setup.sh` will create a virtual environment, and install all the required development dependencies into it.

Note that this will not install a version of OpenCV for you. For that, run `./scripts/setup.sh opencv`.

There are some additional useful scripts to assist:

- `./scripts/test.sh`: Run the unit tests and linters
- `./scripts/fix.sh`: Automatically fix issues from `black` and `isort`
- `./scripts/benchmark.sh`: Run benchmarks (these can take a couple minutes depending on your hardware)

3.1 Cameras

3.1.1 File Cameras

Image File Camera

class zoloto.cameras.file.**ImageFileCamera**(*image_path*, ***, *marker_size=None*, *marker_type*, *calibration_file=None*)

__init__(*image_path*, ***, *marker_size=None*, *marker_type*, *calibration_file=None*)

capture_frame()

Return type ndarray[Any, dtype[+ScalarType]]

close()

Return type None

get_detector_params()

Note: We modify the default parameters slightly to improve detection on markers with hard borders (like the ones generated from *zoloto marker-pdfs*)

Return type *aruco_DetectorParameters*

get_marker_size(*marker_id*)

Return type int

get_visible_markers(*, *frame=None*)

process_frame(*, *frame=None*)

process_frame_eager(*, *frame=None*)

save_frame(*filename*, *, *annotate=False*, *frame=None*)

Video File Camera

class zoloto.cameras.file.**VideoFileCamera**(*video_path*, *, *marker_size=None*, *marker_type*, *calibration_file=None*)

__init__(*video_path*, *, *marker_size=None*, *marker_type*, *calibration_file=None*)

__iter__()

Return type Generator[ndarray[Any, dtype[+ScalarType]], None, None]

capture_frame()

Return type ndarray[Any, dtype[+ScalarType]]

close()

Return type None

get_detector_params()

Note: We modify the default parameters slightly to improve detection on markers with hard borders (like the ones generated from *zoloto marker-pdfs*)

Return type *aruco_DetectorParameters*

get_marker_size(*marker_id*)

Return type int

get_resolution()

get_visible_markers(*, *frame=None*)

process_frame(*, *frame=None*)

process_frame_eager(*, *frame=None*)

save_frame(*filename*, *, *annotate=False*, *frame=None*)

show(*annotate=False*)

Return type None

3.1.2 Cameras

Camera

class zoloto.cameras.camera.**Camera**(*camera_id*, *, *marker_size=None*, *marker_type*, *calibration_file=None*, *resolution=None*)

__init__(*camera_id*, *, *marker_size=None*, *marker_type*, *calibration_file=None*, *resolution=None*)

__iter__()

Return type Iterator[ndarray[Any, dtype[+ScalarType]]]

capture_frame()

Return type ndarray[Any, dtype[+ScalarType]]

close()

Return type None

classmethod discover(***kwargs*)

Return type Generator[*Camera*, None, None]

get_detector_params()

Note: We modify the default parameters slightly to improve detection on markers with hard borders (like the ones generated from *zoloto marker-pdfs*)

Return type *aruco_DetectorParameters*

get_marker_size(*marker_id*)

Return type int

get_resolution()

get_video_capture(*camera_id*)

Return type *VideoCapture*

get_visible_markers(*, *frame=None*)

process_frame(*, *frame=None*)

process_frame_eager(*, *frame=None*)

save_frame(*filename*, *, *annotate=False*, *frame=None*)

show(*annotate=False*)

Return type None

Snapshot Camera

class zoloto.cameras.camera.**SnapshotCamera**(*camera_id*, *, *marker_size=None*, *marker_type*, *calibration_file=None*, *resolution=None*)

A modified version of Camera optimised for single use.

- Doesn't keep the camera open between captures

__init__(*camera_id*, *, *marker_size=None*, *marker_type*, *calibration_file=None*, *resolution=None*)

capture_frame()

Return type ndarray[Any, dtype[+ScalarType]]

close()

Return type None

classmethod discover(***kwargs*)

Return type Generator[*SnapshotCamera*, None, None]

get_detector_params()

Note: We modify the default parameters slightly to improve detection on markers with hard borders (like the ones generated from *zoloto marker-pdfs*)

Return type *aruco_DetectorParameters*

get_marker_size(*marker_id*)

Return type int

get_resolution()

get_video_capture(*camera_id*)

Return type *VideoCapture*

get_visible_markers(*, *frame=None*)

process_frame(*, *frame=None*)

process_frame_eager(*, *frame=None*)

save_frame(*filename*, *, *annotate=False*, *frame=None*)

3.1.3 Raspberry Pi

Pi Camera

class zoloto.cameras.rpi.**PiCamera**(*, *marker_size=None*, *marker_type*, *calibration_file=None*)

__init__(*, *marker_size=None*, *marker_type*, *calibration_file=None*)

__iter__()

Return type Iterator[ndarray[Any, dtype[+ScalarType]]]

capture_frame()

Return type ndarray[Any, dtype[+ScalarType]]

close()

Return type None

get_detector_params()
Note: We modify the default parameters slightly to improve detection on markers with hard borders (like the ones generated from *zoloto marker-pdfs*)

Return type *aruco_DetectorParameters*

get_marker_size(*marker_id*)

Return type int

get_visible_markers(*, *frame=None*)

process_frame(*, *frame=None*)

process_frame_eager(*, *frame=None*)

save_frame(*filename*, *, *annotate=False*, *frame=None*)

show(*annotate=False*)

Return type None

Pi Snapshot Camera

class zoloto.cameras.rpi.**PiSnapshotCamera**(*, *marker_size=None*, *marker_type*, *calibration_file=None*)
A modified version of PiCamera optimised for single use.

- Doesn't keep the camera open between captures

__init__(*, *marker_size=None*, *marker_type*, *calibration_file=None*)

capture_frame()

Return type ndarray[Any, dtype[+ScalarType]]

close()

Return type None

get_detector_params()
Note: We modify the default parameters slightly to improve detection on markers with hard borders (like the ones generated from *zoloto marker-pdfs*)

Return type *aruco_DetectorParameters*

get_marker_size(*marker_id*)

Return type int

get_visible_markers(*, *frame=None*)

process_frame(*, *frame=None*)

process_frame_eager(*, *frame=None*)

save_frame(*filename*, *, *annotate=False*, *frame=None*)

3.1.4 Marker Camera

class zoloto.cameras.marker.**MarkerCamera**(*marker_id*, *marker_size*, *, *marker_type*, *border_size=40*)

A camera which always returns a single, full-screen marker

MIN_BORDER_SIZE = 3

__init__(*marker_id*, *marker_size*, *, *marker_type*, *border_size=40*)

capture_frame()

Return type ndarray[Any, dtype[+ScalarType]]

close()

Return type None

get_detector_params()

Note: We modify the default parameters slightly to improve detection on markers with hard borders (like the ones generated from *zoloto marker-pdfs*)

Return type *aruco_DetectorParameters*

get_marker_size(*marker_id*)

Return type int

get_resolution()

get_visible_markers(*, *frame=None*)

process_frame(*, *frame=None*)

process_frame_eager(*, *frame=None*)

save_frame(*filename*, *, *annotate=False*, *frame=None*)

3.1.5 Base Camera

class zoloto.cameras.base.**BaseCamera**(*, *marker_size=None*, *marker_type*, *calibration_file=None*)

__init__(*, *marker_size=None*, *marker_type*, *calibration_file=None*)

abstract capture_frame()

Return type ndarray[Any, dtype[+ScalarType]]

close()

Return type None

get_detector_params()
Note: We modify the default parameters slightly to improve detection on markers with hard borders (like the ones generated from *zoloto marker-pdfs*)

Return type *aruco_DetectorParameters*

get_marker_size(*marker_id*)

Return type int

get_visible_markers(*, *frame=None*)

process_frame(*, *frame=None*)

process_frame_eager(*, *frame=None*)

save_frame(*filename*, *, *annotate=False*, *frame=None*)

3.2 Markers

class zoloto.marker.**Marker**(*marker_id*, *corners*, *size*, *marker_type*, *calibration_params*)

__init__(*marker_id*, *corners*, *size*, *marker_type*, *calibration_params*)

class zoloto.marker.**EagerMarker**(*marker_id*, *corners*, *size*, *marker_type*, *precalculated_vectors*)

class zoloto.marker.**UncalibratedMarker**(*marker_id*, *corners*, *size*, *marker_type*)

3.3 Marker Type

class zoloto.marker_type.**MarkerType**(*value*)
An enumeration.

APRILTAG_16H5 = 17

APRILTAG_25H9 = 18

APRILTAG_36H10 = 19

APRILTAG_36H11 = 20

ARUCO_4X4 = 3

ARUCO_5X5 = 7

ARUCO_6X6 = 11

ARUCO_7X7 = 15

ARUCO_ORIGINAL = 16

property dictionary: *cv2.aruco_Dictionary*
The underlying OpenCV marker dictionary

Return type *aruco_Dictionary*

property marker_count: int
The total number of markers available

Return type int

property marker_ids: list[int]
All of the possible marker ids

property marker_size: int
Number of bits along 1 size of a marker

Return type int

property max_id: int
The highest id available

Return type int

property min_marker_image_size: int
Minimum size of a marker in pixels

Return type int

3.4 Coordinates

3.4.1 Orientation

class zoloto.coords.Orientation(*e_x*, *e_y*, *e_z*)
The orientation of an object in 3-D space.

__init__(*e_x*, *e_y*, *e_z*)
Construct a quaternion given the components of a rotation vector.

More information: https://w.wiki/Fci

__iter__()
Get an iterator over the rotation angles.

Returns An iterator of floating point angles in order x, y, z.

Return type Iterator[float]

property pitch: float
Get rotation angle around y axis in radians.

Return type float

property quaternion: *pyquaternion.quaternion.Quaternion*
Get the quaternion represented by this orientation.

Return type *Quaternion*

property roll: float
Get rotation angle around x axis in radians.

Return type float

property rot_x: float
Get rotation angle around x axis in radians.

Return type float

property rot_y: float
Get rotation angle around y axis in radians.

Return type float

property rot_z: *float*
Get rotation angle around z axis in radians.

Return type float

rotation_matrix
Get the rotation matrix represented by this orientation.

Returns A 3x3 rotation matrix as a tuple of tuples.

property yaw: *float*
Get rotation angle around z axis in radians.

Return type float

yaw_pitch_roll
Get the equivalent yaw-pitch-roll angles.

Specifically intrinsic Tait-Bryan angles following the z-y'-x" convention.

3.4.2 PixelCoordinates

class zoloto.coords.**PixelCoordinates**(*x: float*, *y: float*)
Coordinates within an image made up from pixels.

This type allows float values to account for computed locations which are not limited to exact pixel boundaries.

Parameters

- **x** (*float*) – X coordinate
- **y** (*float*) – Y coordinate

x: *float*
Alias for field number 0

y: *float*
Alias for field number 1

3.4.3 CartesianCoordinates

class zoloto.coords.**CartesianCoordinates**(*x: float*, *y: float*, *z: float*)

Parameters

- **x** (*float*) – X coordinate
- **y** (*float*) – Y coordinate
- **z** (*float*) – Z coordinate

x: *float*
Alias for field number 0

y: *float*
Alias for field number 1

z: *float*
Alias for field number 2

3.4.4 SphericalCoordinates

class zoloto.coords.**SphericalCoordinates**(*rot_x: float*, *rot_y: float*, *dist: int*)

Parameters

- **rot_x** (*float*) – Rotation around the X-axis, in radians
- **rot_y** (*float*) – Rotation around the Y-axis, in radians
- **dist** (*float*) – Distance

dist: int
Alias for field number 2

rot_x: float
Alias for field number 0

rot_y: float
Alias for field number 1

3.4.5 Quaternion

class pyquaternion.quaternion.**Quaternion**
See https://kieranwynn.github.io/pyquaternion/

3.5 Discovery

class zoloto.cameras.camera.**Camera**(*camera_id*, *, *marker_size=None*, *marker_type*, *calibration_file=None*, *resolution=None*)

classmethod discover(**kwargs*)

Return type Generator[*Camera*, None, None]

class zoloto.cameras.camera.**SnapshotCamera**(*camera_id*, *, *marker_size=None*, *marker_type*, *calibration_file=None*, *resolution=None*)
A modified version of Camera optimised for single use.

- Doesn't keep the camera open between captures

classmethod discover(**kwargs*)

Return type Generator[*SnapshotCamera*, None, None]

3.6 Calibration

To perform accurate pose estimation, each camera must be calibrated. To calibrate the camera, OpenCV ships with a tool to assist.

The resulting calibration file can be passed into a *zoloto.cameras.camera.Camera*.

Note: Occasionally on Linux, the tool will fail to open the camera. This happens as it uses gstreamer backend by default, whereas Zoloto uses v4l2. To disable gstreamer, set the `OPENCV_VIDEOIO_PRIORITY_GSTREAMER=0` environment variable.

3.6.1 Calibration Parameters

class zoloto.calibration.**CalibrationParameters**(*camera_matrix*, *distance_coefficients*, *resolution*)

__iter__()
Implement iter(self).

camera_matrix: NDArray[floating]
Alias for field number 0

count(*value*, */*)
Return number of occurrences of value.

distance_coefficients: NDArray[floating]
Alias for field number 1

index(*value*, *start=0*, *stop=9223372036854775807*, */*)
Return first index of value.

Raises ValueError if the value is not present.

resolution: tuple[int, int]
Alias for field number 2

3.7 Command Line Interface

Zoloto ships with a CLI (aptly named `zoloto`), which contains some helpful utils for working with Zoloto and fiducial markers.

3.7.1 Save markers

The `save-markers` tool outputs the images of all the fiducial markers in a given type.

Each marker is surrounded by a white boarder, which is not considered part of the marker (it's not counted when working out the marker's size).

When *–raw* is passed, Markers are output as PNG files, at their smallest possible format. They can then be resized as necessary without losing quality.

Without *–raw*, images are saved 500px, plus a border with text identifying which marker is being used.

3.7.2 Marker PDFs

The `marker-pdfs` tool outputs marker images onto A4 PDFs at the required size.

Each marker is surrounded by an extra pixel of white which is boarded by a grey line. When specifying the size of the marker, this white padding is not included. When cutting out the markers, this padding must be included.

3.8 OpenCV Types

This page documents OpenCV types that could not be automatically included in the documentation.

class cv2.**aruco_DetectorParameters**

class cv2.**VideoCapture**

class cv2.**aruco_Dictionary**

INDEX

Symbols

A

B

C

D

E

G

I

M

O

P

Q

R

S

U

V

X

Y

Z