VideoF2B

Alberto Solera

Apr 25, 2024

TABLE OF CONTENTS

Hello everyone!

VideoF2B is a **desktop application** that draws figures in videos of **Control Line F2B** stunt flights.

To get started, navigate the **table of contents** or follow one of these links:

If you find any errors, omissions, opportunities for improvement, etc., let us know via email or GitHub!

CHAPTER

ONE

INSTALLING VIDEOF2B

VideoF2B is a single-file executable. Installation is simple: just download the appropriate file for your operating system.

Choose instructions below based on your operating system:

1. Download the latest `VideoF2B.exe` from here.
2. Move the file to any folder of your choice.
3. Run the application by double-clicking it.
4. Enjoy!

1. Download the latest `videof2b` binary from here.
2. Move the file to any directory of your choice.
3. Run the application.
4. Enjoy!

Having trouble with installation? See *Getting help* or *FAQ* for guidance.

CHAPTER

TWO

USING VIDEOF2B

The most prominent feature of VideoF2B is the drawing (tracing) of a path behind a Control Line aircraft in video. These traces help us to visualize the figures that a Stunt pilot performed during a flight. There are two general ways in which you can use this capability in VideoF2B: *traces only*, and traces with *Augmented Reality*.

TODO: perhaps these two are better as glossary terms?

Traces only

TODO: overview of use case with uncalibrated videos.

Augmented Reality (AR)

TODO: overview of use case with calibrated videos.

2.1 Camera calibration

Before you go to the field…

TODO: describe how to calibrate a camera for use with AR.

2.2 Field setup

When you get to the flying field…

TODO: describe how to arrange field equipment for best results.

2.3 Processing your videos

After you recorded your videos…

TODO: describe full details of app usage here.

CHAPTER

THREE

GETTING HELP

TODO: guidance on what to do when things go wrong.

CHAPTER

FOUR

FAQ

TODO: list answers to the most common questions and issues that we are aware of.

CHAPTER

FIVE

VIDEOF2B

5.1 videof2b package

5.1.1 Subpackages

videof2b.core package

Subpackages

videof2b.core.common package

Submodules

videof2b.core.common.path module

Path-related utilities.

videof2b.core.common.path.**files_to_paths**(*file_names*)

Convert a list of file names to a list of file Path objects.

Parameters
file_names (*list[str]*) – The list of file names to convert.

Returns
The list converted to file paths

Return type
list[Path]

videof2b.core.common.path.**path_to_str**(*path=None*)

Convert a Path object or NoneType to a string equivalent.

Parameters
path (*Path | None*) – The value to convert to a string

Returns
An empty string if *path* is None, else a string representation of the *path*

Return type
str

videof2b.core.common.path.**replace_params**(*args*, *kwargs*, *params*)

Transform the specified args or kwargs

Parameters

- **args** (*tuple*) – Positional arguments
- **kwargs** (*dict*) – Keyword arguments
- **params** – A tuple of tuples with the position and the keyword to replace

Returns
The modified positional and keyword arguments

Return type
tuple[tuple, dict]

Usage: Given a method with the following signature, assume we want to apply the *str* function to *arg2*: *def method(arg1=None, arg2=None, arg3=None)*

Since *arg2 can be specified positionally as the second argument (1 with a zero index) or as a keyword, we would call this function as follows: `replace_params(args, kwargs, ((1, 'arg2', str),))*

videof2b.core.common.path.**str_to_path**(*string*)

Convert a str object to a Path or NoneType.

This is especially useful because constructing a Path object with an empty string causes the Path object to point to the current working directory, which is not desirable.

Parameters
string (*str*) – The string to convert

Returns
None if *string* is empty, or a Path object representation of *string*

Return type
Path | None

videof2b.core.common.settings module

Module for handling persistent settings.

class videof2b.core.common.settings.**Settings**(**args*, ***kwargs*)

Bases: QSettings

Simple wrapper around QSettings. Contains core definitions of all known keys and their default values. Does not contain a strategy for versioning of settings. Handles lookup of default values in the most basic manner.

staticMetaObject = PySide6.QtCore.QMetaObject("Settings" inherits "QSettings":)

value(*key*)

Return the value for the given key. The key must exist in *Settings.__defaults__*. If value not found, return the value from *Settings.__defaults__*

videof2b.core.common.singleton module

Contains the Singleton class definition.

class videof2b.core.common.singleton.**Singleton**

Bases: type

Implementation of a Singleton metaclass.

videof2b.core.common.store module

This module contains objects that enable persistence of custom data.

class videof2b.core.common.store.**Store**(**args*, ***kwargs*)

Bases: object

An object store. This is a singleton that provides access to object references that are shared within a process.

add(*key*, *item*)

Add an item to the store.

classmethod create()

The constructor for the Store.

get(*key*)

Get the specified object from the store.

remove(*key*)

Remove an item from the store.

class videof2b.core.common.store.**StoreProperties**

Bases: object

Adds shared components to classes for use at run time.

property application

Dynamically-added application object.

property settings

Dynamically-added settings object.

Module contents

Common definitions and constants for VideoF2B.

class videof2b.core.common.**FigureTypes**(*value*)

Bases: Enum

All F2B figures in sequence.

FOUR_LEAF_CLOVER = 14

HORIZONTAL_EIGHTS = 9

HORIZONTAL_SQUARE_EIGHTS = 10

HOURGLASS = 12

INSIDE_LOOPS = 3

INSIDE_SQUARE_LOOPS = 6

INSIDE_TRIANGULAR_LOOPS = 8

INVERTED_FLIGHT = 4

LANDING = 15

OUTSIDE_LOOPS = 5

OUTSIDE_SQUARE_LOOPS = 7

OVERHEAD_EIGHTS = 13

REVERSE_WINGOVER = 2

TAKEOFF = 1

VERTICAL_EIGHTS = 11

class videof2b.core.common.**SphereManipulations**(*value*)

Bases: Enum

Possible manipulations of the AR sphere during processing.

MOVE_EAST = 4

MOVE_NORTH = 5

MOVE_SOUTH = 6

MOVE_WEST = 3

RESET_CENTER = 0

ROTATE_CCW = 1

ROTATE_CW = 2

videof2b.core.common.**get_app_metadata**()

Get basic app information. Returns (name, version) as a tuple of strings.

Return type
Tuple

videof2b.core.common.**get_bundle_dir**()

Return the path of the bundle directory. When frozen as a one-file app, this is the _MEI### dir in temp. When frozen as a one-dir app, this is that dir. When running as a script, this is the project's root dir.

videof2b.core.common.**get_frozen_path**(*path_when_frozen*, *path_when_non_frozen*)

Return one of the given paths based on status of *sys.frozen*.

videof2b.core.common.**get_lib_versions**()

Get User-friendly names and versions of libraries that we care about for bug reports. This is just a sub-list of *install_requires* items in our *setup.cfg*.

videof2b.core.common.**is_linux**()

Returns True if running on a Linux OS.

Return type
bool

Returns
True if running on a Linux OS.

videof2b.core.common.**is_win**()

Returns True if running on a Windows OS.

Return type
bool

Returns
True if running on a Windows OS.

videof2b.core.common.**launch_document**(*path*)

Open the specified document using the default application.

Return type
None

Submodules

videof2b.core.calibration module

Module for calibrating cameras.

class videof2b.core.calibration.**CalibratorReturnCodes**(*value*)

Bases: IntEnum

Definition of the return codes from CameraCalibrator's processing loop.

EXCEPTION_OCCURRED = -2

INSUFFICIENT_VALID_FRAMES = 3

NORMAL = 0

NO_VALID_FRAMES = 2

UNDEFINED = -1

USER_CANCELED = 1

class videof2b.core.calibration.**CameraCalibrator**(*path*, *is_fisheye=False*)

Bases: QObject

Calibrates a camera.

error_occurred

finished

new_frame_available

progress_updated

run()

Calibrate the camera using the chessboard pattern.

staticMetaObject = PySide6.QtCore.QMetaObject("CameraCalibrator" inherits "QObject":
Methods: #5 type=Signal, signature=new_frame_available(QImage), parameters=QImage
#6 type=Signal, signature=progress_updated(PyObject), parameters=PyObject #7
type=Signal, signature=finished(int), parameters=int #8 type=Signal,
signature=error_occurred(QString,QString), parameters=QString, QString)

stop()

Cancel the calibration procedure.

videof2b.core.camera module

This module contains representations of cameras.

class videof2b.core.camera.**CalCamera**(*frame_size*, *flight*)

Bases: QObject

Represents a real-world camera whose intrinsic and extrinsic optical properties are known.

load_calibration(*path*)

Load a camera calibration from the specified path.

locate(*flight*)

Locate a given Flight instance using this camera.

Return type
bool

staticMetaObject = PySide6.QtCore.QMetaObject("CalCamera" inherits "QObject":)

undistort(*img*)

Undistort a given image according to the camera's calibration.

videof2b.core.camera_director module

Calculator for placing a camera in the field.

class videof2b.core.camera_director.**CamDirector**

Bases: object

The core calculator of cam placement geometry.

property A: float

Maximum vertical viewing angle of the camera, in degrees.

property C: float

Signed camera height relative to flight equator. Up is positive.

DEFAULT_A = 71.75

DEFAULT_C = -1.0

DEFAULT_G = -1.5

DEFAULT_R = 21.0

property G: float

Signed ground level relative to flight equator. Up is positive.

property R: float

Sphere radius. Must be positive and nonzero.

property cam_distance_limits: Iterable

The limits (min, max) of camera distance from flight center.

property cam_tangent_elev_limits: Iterable

The limits (min, max) of the elevation of the tangent point from camera to sphere at the respective *cam_distance_limits*. The point's elevation is measured from flight center in degrees.

property cam_view_limits: Iterable

The limits (min, max) of camera vertical viewing angle, in degrees, at the respective *cam_distance_limits*.

solve()

For the current state, solve for the following:
d_min and *d_max* such that *alpha* <= *max_alpha*

Subject to the constraint:

- maximum vertical view angle must include the 45-degree latitude and the center field marker on the ground.

videof2b.core.detection module

This module performs motion detection in video.

class videof2b.core.detection.**Detector**(*maxlen*, *scale*)

Bases: object

The primary motion detector in VideoF2B.

clear()

Clear the currently detected track.

process(*img*)

Detect a moving object in a given image frame.

videof2b.core.drawing module

Module for drawing flight track and Augmented Reality in video.

class videof2b.core.drawing.**Colors**

Bases: object

Shortcuts for OpenCV-compatible colors.

BLACK = (0, 0, 0)

BLUE = (255, 0, 0)

CYAN = (255, 255, 0)

DarkGreen = (0, 204, 0)

GRAY20 = (50, 50, 50)

GREEN = (0, 255, 0)

MAGENTA = (255, 0, 255)

RED = (0, 0, 255)

WHITE = (255, 255, 255)

YELLOW = (0, 255, 255)

class videof2b.core.drawing.**DashedPolyline**(*obj_pts*, ***kwargs*)

Bases: *Plot*

Defines a polyline that is drawn dashed.

draw(*img*, *key*, ***kwargs*)

Draw this polyline using its attributes.

class videof2b.core.drawing.**Drawing**(*detector*, ***kwargs*)

Bases: object

Container that performs all the drawing of AR sphere, track, figures, etc., in any given image frame.

DEFAULT_N = 100

draw(*img*)

Draw all relevant geometry in the given image frame.

property draw_diags

Controls the drawing of diagnostics.

property draw_endpts

Controls the drawing of maneuver endpoints.

locate(*cam*, *flight=None*, ***kwargs*)

Locate a new Flight or relocate an existing one using the given camera.

Return type
None

move_center_x(*delta*)

Move sphere center by *delta* along world X direction, in meters.

move_center_y(*delta*)

Move sphere center by *delta* along world Y direction, in meters.

reset_center()

Reset sphere center to default.

set_azimuth(*azimuth*)

Set the aximuth of the AR sphere, in degrees.

class videof2b.core.drawing.**DummyScene**

Bases: object

Placeholder object for an empty scene.

draw(**args*, ***kwargs*)

No-op method.

class videof2b.core.drawing.**EdgePolyline**(*R*, *cam_pos*, ***kwargs*)

Bases: *Polyline*

Defines a special polyline that represents the visible edge of the sphere. This polyline is aware of the camera's position.

class videof2b.core.drawing.**ManeuverPoint**(*point*, *size*, *color*)

Bases: object

Represents either endpoint of a maneuver (start or end).

class videof2b.core.drawing.**Plot**(*obj_pts*, ***kwargs*)

Bases: object

Base class for plotting primitives. * Call *draw()* to draw the Plot instance in your image. kwargs:

size: the line thickness or point radius.

color: the color of the primitives in this Plot.

is_fixed: bool indicating whether this Plot is fixed in object space or not.

If True, world transforms do not affect the object coordinates. If False (default), then world transforms will rotate, scale, and translate the object coordinates according to the rules in the *_calculate()* method.

draw(*key*, ***kwargs*)

Call this method from derived classes before drawing the image points.

class videof2b.core.drawing.**Polyline**(*obj_pts*, ***kwargs*)

Bases: *Plot*

Defines a polyline.

draw(*img*, *key*, ***kwargs*)

Draw this polyline using its attributes.

class videof2b.core.drawing.**Scatter**(*obj_pts*, ***kwargs*)

Bases: *Plot*

Defines a collection of scattered points.

draw(*img*, *key*, ***kwargs*)

Draw scatter points as solid-filled circles using their attributes.

class videof2b.core.drawing.**Scene**(**items*)

Bases: object

A scene consists of a collection of Plot-like objects.

add(*item*)

Add an item to this scene.

add_diag(*item*)

Add a diagnostic item to this scene.

add_endpt(*item*)

Add an endpoint item to this scene.

property diags_on

Boolean flag that controls drawing of diagnostics.

draw(*img*, *key*, ***kwargs*)

Draw this scene in the given image.

property endpts_on

Boolean flag that controls drawing of maneuver endpoints.

videof2b.core.figure_tracker module

F2B Figure tracker.

class videof2b.core.figure_tracker.**FigureTracker**(*callback=<function FigureTracker.<lambda>>*, ***kwargs*)

Bases: object

Container that tracks F2B figures. May be used for fitting the actual flight path to the nominal figure to determine a score.

All FAI figures, per "F2, Annex 4J – F2B Manoeuvre Diagrams" for reference.

Not all may be easy to track:

- 4.J.1 Take-off (Rule 4.2.15.3)
- 4.J.2 Reverse wingover (Rule 4.2.15.4)
- 4.J.3 Three consecutive inside loops (Rule 4.2.15.5)
- 4.J.4 Two consecutive laps of inverted level flight (Rule 4.2.15.6)
- 4.J.5 Three consecutive outside loops (Rule 4.2.15.7)
- 4.J.6 Two consecutive inside square loops (Rule 4.2.15.8)
- 4.J.7 Two consecutive outside square loops (Rule 4.2.15.9)
- 4.J.8 Two consecutive inside triangular loops (Rule 4.2.15.10)
- 4.J.9 Two consecutive horizontal eight (Rule 4.2.15.11)
- 4.J.10 Two consecutive horizontal square eight (Rule 4.2.15.12)
- 4.J.11 Two consecutive vertical eight (Rule 4.2.15.13)
- 4.J.12 Hourglass (Rule 4.2.15.14)
- 4.J.13 Two consecutive overhead eight (Rule 4.2.15.15)
- 4.J.14 Four-leaf clover manoeuvre (Rule 4.2.15.16)
- 4.J.15 Landing manoeuvre (Rule 4.2.15.17)

FIGURE_MAP = {0: FigureTypes.INSIDE_LOOPS}

add_actual_point(*idx*, *point*)

Add a measured (actual) point to the currently tracked figure at a given index. If no figure is currently being tracked, this call has no effect.

export(*path*)

Export all tracked figures as numpy arrays to the specified file. Arrays are labeled "fig0", "fig1", etc.

finish_all()

Clean-up method. Finish current figure, if any figure is in progress.

finish_figure()

Finish trfig_type_constructorhe currently tracked figure.

start_figure()

Start tracking a new figure.

exception videof2b.core.figure_tracker.**UserError**

Bases: Exception

Class for exception that occur during Figure tracking due to user errors.

videof2b.core.figures module

Geometric definitions for F2B figures.

class videof2b.core.figures.**Figure**(*R=None*, *actuals=None*, ***kwargs*)

Bases: object

Base class for all F2B figures.

static create(*which_figure*, *R=None*, *actuals=None*, ***kwargs*)

Factory method for creating a given Figure on a sphere of radius R with specified actual path points.

fit()

Perform best fit of actual points against the nominals of the figure.

get_nom_point(*a*, *b*, *c*, **t*)

Returns the nominal point at a given 0.0 < t < 1.0 using the figure's parameters.

u

Suggested count of nominal points for a reasonably smooth-looking figure. Override in subclasses as needed. This is primarily used for drawing.

class videof2b.core.figures.**FigureDiagnostics**(*enabled=False*)

Bases: object

Contains the diagnostic settings in a figure.

class videof2b.core.figures.**InsideLoops**(*R=None*, *actuals=None*, ***kwargs*)

Bases: *Figure*

Represents three consecutive inside loops per F2B Rule 4.2.15.5 and Diagram 4.J.3 in the Annex. kwargs:

Parameters
enable_diags – enables diagnostic output and plotting of various behind-the-scenes stuff.

fit()

Fit actual flight path to this Figure's nominal path.

videof2b.core.figures.**find_min_gss**(*f*, *a*, *b*, *eps=0.0001*)

Find Minimum by Golden Section Search Method Returns the value of x that minimizes the function f(x) on interval [a, b]

videof2b.core.figures.**test**()

Test cases.

videof2b.core.flight module

Defines a recorded flight.

class videof2b.core.flight.**Flight**(*vid_path*, *is_live=False*, *cal_path=None*, ***kwargs*)

Bases: QObject

Contains information about a flight to be processed.

add_locator_point(*point*)

Add a potential locator point.

Return type
None

clear_locator_points()

Clear all locator points.

locator_points_changed

locator_points_defined

obj_point_names = ('circle center', 'front marker', 'left marker', 'right marker')

on_locator_points_changed()

Signals that locator points changed, and the new instruction message.

pop_locator_point()

Remove the last added locator point, if it exists.

Return type
None

restart()

Restart this flight's video stream.

staticMetaObject = PySide6.QtCore.QMetaObject("Flight" inherits "QObject": Methods: #5 type=Signal, signature=locator_points_changed(PyObject,QString), parameters=PyObject, QString #6 type=Signal, signature=locator_points_defined())

videof2b.core.geometry module

General geometry related to F2B figures.

exception videof2b.core.geometry.**ArgumentError**

Bases: Exception

Thrown when the provided combination of arguments is invalid.

class videof2b.core.geometry.**Fillet**(*R*, *r*, *psi*, *is_degrees=False*)

Bases: object

Represents a spherical fillet.

On a sphere of radius R, a fillet is defined as an arc of a small circle of radius r between two great arcs of the sphere with angle psi (ψ) between the arcs such that the small circle is tangent to both arcs. The constructor of this class tries to calculate the parameters of the fillet via the *calculate()* method.

Parameters

- **R** (float) – radius of the sphere.
- **r** (float) – radius of the fillet.
- **psi** (float) – angle between two great arcs that define the fillet.
- **is_degrees** (Optional[bool], default: False) – If True, psi is given in degrees, otherwise it is given in radians.

calculate()

Calculate the fillet as follows:

Given two intersecting planes with angle ψ between them, a cone of slant height R and base radius r whose apex rests on the intersection line of the planes will rest tangent to both planes when the cone's axis makes an angle θ with the intersection line of the planes in the bisecting plane.

If we set up a coordinate system on the cone's base such that: :rtype: None

- the origin is at the cone's apex;
- the $-z$ axis is along the cone's axis toward the cone's base; and
- the $+x$ axis is toward the intersection line,

then the coordinates of the points of tangency between the cone and the planes are $(x_p, y_p, -d)$ and $(x_p, -y_p, -d)$.

Let β be the central angle of the arc along the cone's base that joins the two tangency points on the side of the intersection (the shorter of the two possible arcs).

Perform the following:

- Ensure that tangency is possible. This requires that

$$2\alpha \leqslant \psi \leqslant \pi$$

  where $\alpha = \arcsin\left(\frac{r}{R}\right)$ is the half-angle of the cone's apex.

  If this condition fails, the instance attribute is_valid is set to False and this method returns early.
- Find angle θ (Gorjanc solution). Store it in the instance attribute theta.
- Find x_p, y_p, and d. Store them in instance attributes x_p, y_p, and d, respectively.
- Find angle β. Store it in the instance attribute beta.
- Set is_valid to True and return.

See also:

method *get_fillet_theta()*.

static get_fillet_theta(*R*, *r*, *psi*)

Calculate the angle θ between cone axis and intersection line of the planes that are tangent to a *Fillet*.

Implements the Gorjanc solution. Values of x_p and y_p follow from this as well.

Parameters

- **R** (float) – radius of the sphere on which the fillet is defined.
- **r** (float) – radius of the fillet, which is also equal to the base radius of the fillet's cone.
- **psi** (float) – angle between the planes that are tangent to the fillet's cone.

Return type
float

Note: This method is provided as a static method for optimizer use in addition to use by *Fillet* here.

See also:

The *calculate()* method.

videof2b.core.geometry.**angle**(*a*, *b*)

Calculate the angle between two vectors in radians.

Parameters

- **a** (Union[_SupportsArray[dtype], _NestedSequence[_SupportsArray[dtype]], bool, int, float, complex, str, bytes, _NestedSequence[Union[bool, int, float, complex, str, bytes]]]) – any vector in 2D or 3D.
- **b** (Union[_SupportsArray[dtype], _NestedSequence[_SupportsArray[dtype]], bool, int, float, complex, str, bytes, _NestedSequence[Union[bool, int, float, complex, str, bytes]]]) – any vector in 2D or 3D.

Return type
float

Returns
the angle between a and b in radians.

videof2b.core.geometry.**calc_equilateral_sigma**(*height=0.7853981633974483*)

Solve for the side of an equilateral spherical triangle whose height is given.

This is the companion to the *get_equilateral_height()* function.

Parameters
height (Optional[float], default: 0.7853981633974483) – height of the equilateral spherical triangle, in radians. Defaults to $\frac{\pi}{4}$.

Return type
float

Returns
side angle σ, in radians.

videof2b.core.geometry.**calc_tri_loop_params**(*R*, *r*, *target_elev=0.7853981633974483*)

Calculate the basic parameters of a triangular loop figure on a sphere.

Given an equilateral spherical triangle on the surface of a sphere of radius R such that the top of a corner turn of radius r is located at target_elev on the sphere, calculate:

- The central angle σ of the side of the triangle,
- The angle ϕ between adjacent sides of the triangle.

Parameters

- **R** (float) – radius of the sphere.
- **r** (float) – radius of the loop's corner turns.
- **target_elev** (Optional[float], default: 0.7853981633974483) – elevation of the highest point in the top turn. Defaults to $\frac{\pi}{4}$.

Return type
Tuple[float]

Returns
sigma, phi

All angles are in radians.

videof2b.core.geometry.**cartesian_to_spherical**(*p*)

Convert a point from Cartesian coordinates to elevation-based spherical coordinates.

Parameters
p (Union[_SupportsArray[dtype], _NestedSequence[_SupportsArray[dtype]], bool, int, float, complex, str, bytes, _NestedSequence[Union[bool, int, float, complex, str, bytes]]]) – an array or sequence representing a point `(x, y, z)` in Cartesian space.

Return type
ndarray

Returns

an array containing `(r, theta, phi)` where

- `r` = radius,
- `theta` = elevation angle θ,
- `phi` = azimuth angle ϕ.

Seealso
the inverse conversion function is *spherical_to_cartesian()*.

All angles are in radians.

videof2b.core.geometry.**get_arc**(*r*, *alpha*, *rho=100*)

Create an array of 3D points that represent a circular arc.

Return 3D points for an arc of radius `r` and included angle `alpha` with point density `rho`, where `rho` is the number of points per 2π. Arc center is `(0, 0, 0)`. The arc lies in the xy plane. The arc starts at zero angle, i.e., at `(r, 0, 0)`, and proceeds counterclockwise until it ends at `alpha`. Angle measurements are in radians. The endpoint is always included.

Parameters

- **r** (float) – radius of the arc.
- **alpha** (float) – included angle of the arc in radians.
- **rho** (Optional[int], default: 100) – angular density of generated points. Defaults to 100.

Return type
ndarray

Returns
`(N, 3)` array of points, where `N >= 3` and is proportional to `alpha` and `rho`.

Warning: The meaning of the `rho` parameter may change in the future from angular density to circumferential (linear) density to provide more consistent point spacing on arcs of different radii in the same scene.

videof2b.core.geometry.**get_cone_alpha**(*R*, *r*)

Calculates the half-aperture of a cone with slant height `R` and base radius `r`.

Parameters

- **R** (`float`) – slant height of cone.
- **r** (`float`) – base radius of cone.

Return type
`float`

Returns
angle α in radians.

videof2b.core.geometry.**get_cone_d**(*R*, *r*)

Perpendicular height of a cone with slant height `R` and base radius `r`.

Parameters

- **R** (`float`) – slant height of cone.
- **r** (`float`) – base radius of cone.

Raises
ValueError if `R < r`

Return type
`float`

Returns
height `d` of the cone.

videof2b.core.geometry.**get_cone_delta**(*alpha*, *theta=None*, *beta=None*)

Calculate the properties of a cone rotated from the flight base to a certain elevation.

Consider a base cone whose axis lies in the xy plane, and whose ruled surface contains the y axis. Rotate this cone around the x axis by an angle β such that the elevation of the cone's axis is at angle θ. This result is important because it preserves the cone's tangency point with the yz plane after the rotation. In the cone's base plane, a line segment from the cone axis to this point of tangency lies in the xy plane when the cone is unrotated (the "base" cone). After rotation of the cone by β, this same line segment is no longer parallel to the xy plane. Effectively, it has been rotated around the cone's axis by an angle that we hereby call δ. The goal is to calculate δ and one of the other angles such that the caller has all three angles δ, θ, and β at its disposal.

Parameters

- **alpha** (`float`) – the cone's half-aperture, in radians. This is required.
- **theta** (`Optional[float]`, default: `None`) – the elevation of the cone's axis, in radians.
- **beta** (`Optional[float]`, default: `None`) – the rotation of the cone around the x axis from the base, in radians.

Raises
ArgumentError – when `theta` and `beta` arguments are supplied inconsistently, i.e., when both are given or neither is given.

Return type
`Tuple[float]`

Returns

angle δ and one of the missing angles θ or β. Two mutually exclusive cases are possible:

- **theta is known (as in top corners of square loops)**
  –> return `(delta, beta)`
- **beta is known (as in clover loops).**
  –> return `(delta, theta)`

videof2b.core.geometry.**get_equilateral_height**(*sigma*)

Calculate the height of an equilateral spherical triangle as a function of its side angle σ. Takes advantage of the cosine rule in spherical trigonometry.

This is the companion to the *calc_equilateral_sigma()* function.

Parameters
sigma (`float`) – the side angle σ of the spherical triangle.

Return type
`float`

Returns
the height of the equilateral spherical triangle in radians.

videof2b.core.geometry.**get_equilateral_phi**(*sigma*)

Calculate the angle between sides of an equilateral spherical triangle as a function of side angle *sigma*. See the derivation here.

Parameters
sigma (`float`) – the side angle σ of the spherical triangle.

Return type
`float`

Returns
the angle ϕ between the sides of the equilateral spherical triangle.

videof2b.core.geometry.**spherical_to_cartesian**(*p*)

Convert a point from elevation-based spherical coordinates to Cartesian coordinates.

Parameters
p (`Union[_SupportsArray[dtype], _NestedSequence[_SupportsArray[dtype]], bool, int, float, complex, str, bytes, _NestedSequence[Union[bool, int, float, complex, str, bytes]]]`) – an array or sequence like `(r, theta, phi)` where

- `r` = radius,
- `theta` = elevation angle θ,
- `phi` = azimuth angle ϕ.

Return type
`ndarray`

Returns
an array containing `(x, y, z)`.

Seealso
the inverse conversion function is *cartesian_to_spherical()*.

All angles are in radians.

videof2b.core.imaging module

Imaging functions.

videof2b.core.imaging.**cv_img_to_qimg**(*cv_img*)

Convert a cv2 image to a QImage for display in QPixmap objects.

Return type
QImage

videof2b.core.processor module

The main flight processor in VideoF2B.

class videof2b.core.processor.**ProcessorReturnCodes**(*value*)

Bases: IntEnum

Definition of the return codes from VideoProcessor's processing loop.

EXCEPTION_OCCURRED = -2

NORMAL = 0

POSE_ESTIMATION_FAILED = 2

TOO_MANY_EMPTY_FRAMES = 3

UNDEFINED = -1

USER_CANCELED = 1

class videof2b.core.processor.**ProcessorSettings**

Bases: object

Stores persistable user settings.

im_width = 960

live_videos = PosixPath('../VideoF2B_videos')

max_track_time = 15

perform_3d_tracking = False

sphere_rot_delta = 0.5

sphere_xy_delta = 0.1

class videof2b.core.processor.**VideoProcessor**

Bases: QObject, *StoreProperties*

Main video processor. Handles processing of a video input from start to finish.

add_locator_point(*point*)

Add a potential locator point.

ar_geometry_available

clear_track()

Clear the aircraft's existing flight track.

error_occurred

finished

load_flight(*flight*)

Load a Flight and prepare for processing.

Parameters
flight (Flight) – a properly populated Flight instance.

Return type
None

locating_started

locator_points_changed

locator_points_defined

manipulate_sphere(*command*)

Manipulate the AR sphere via the specified command.

Return type
None

mark_figure(*is_start*)

Mark the start/end of a tracked figure.

Parameters
is_start (bool) – True to mark the start of the figure, False to mark the end.

Return type
None

new_frame_available

on_locator_points_changed(*points*, *msg*)

Handles changes in locator points during the camera locating procedure.

on_locator_points_defined()

Locator points are completely defined. Let the world know.

pause_resume()

Pause/Resume processing at the current frame. Allows the following functionality with immediate feedback while paused: * To quit processing. * To clear the track. * To manipulate sphere rotation and movement.

paused

pop_locator_point(_)

Remove the last locator point, if one exists.

progress_updated

relocate()

Relocate the flight.

run()

Run the processor.

staticMetaObject = PySide6.QtCore.QMetaObject("VideoProcessor" inherits "QObject": Methods: #5 type=Signal, signature=locating_started() #6 type=Signal, signature=locator_points_changed(PyObject,QString), parameters=PyObject, QString #7 type=Signal, signature=locator_points_defined() #8 type=Signal, signature=ar_geometry_available(bool), parameters=bool #9 type=Signal, signature=new_frame_available(QImage), parameters=QImage #10 type=Signal, signature=progress_updated(PyObject), parameters=PyObject #11 type=Signal, signature=finished(int), parameters=int #12 type=Signal, signature=track_cleared() #13 type=Signal, signature=paused(bool), parameters=bool #14 type=Signal, signature=error_occurred(QString,QString), parameters=QString, QString)

stop()

Respond to a "nice" request to stop our processing loop.

stop_locating()

Cancel the flight locating procedure.

track_cleared

update_figure_diags(*val*)

Update figure diags state in the drawing.

Return type
None

update_figure_state(*figure_type*, *val*)

Update figure state in the drawing.

Return type
None

update_maneuver_endpts(*val*)

Update maneuver endpoints in the drawing.

Return type
None

videof2b.core.projection module

Project image points onto world sphere.

videof2b.core.projection.**project_image_point_to_sphere**(*frame*, *cam*, *radius*, *center*, *img_point*, *data_writer*)

Project image point to world sphere given a calibrated camera and frame.

Parameters

- **frame** – the image frame of interest.
- **cam** – instance of CalCamera.
- **radius** – radius of sphere.
- **center** – XYZ location of sphere center wrt world pose estimation.
- **img_point** – the image point that we wish to project onto the sphere.

- **data_writer** – a handle to a file-like object.

videof2b.core.projection.**project_sphere_points_to_image**(*cam*, *world_pts*, *frame=None*)

Project the given sphere points to image space.

Module contents

Core modules of VideoF2B application.

videof2b.ui package

Submodules

videof2b.ui.about_window module

The dialog window for the "about" information.

class videof2b.ui.about_window.**AboutDialog**(*parent*)

Bases: QDialog

About window.

setup_ui()

Designs the UI.

staticMetaObject = PySide6.QtCore.QMetaObject("AboutDialog" inherits "QDialog":)

videof2b.ui.camera_cal_dialog module

The dialog window for camera calibration.

class videof2b.ui.camera_cal_dialog.**CameraCalibrationDialog**(*parent*)

Bases: QDialog, *StoreProperties*

Camera calibration UI.

on_doc_open()

Open the calibration pattern PDF file.

on_fisheye_changed(*state*)

Update fisheye flag when checkbox state changes.

on_image_display()

Show a lightweight window in full screen with the calibration pattern. Esc key closes this window by default.

on_path_changed(*new_path*)

Update UI when the calibration path changes.

setup_ui()

Designs the UI.

staticMetaObject = PySide6.QtCore.QMetaObject("CameraCalibrationDialog" inherits "QDialog":)

videof2b.ui.camera_director_dialog module

Interactive tool for placing a camera in the field.

class videof2b.ui.camera_director_dialog.**CamDirectorDialog**(*parent*)

Bases: QDialog

Interactive camera placement aid.

on_new_solution(_)

Update results view when a new solution is available.

setup_model()

Create the models.

setup_ui()

Create the UI.

staticMetaObject = PySide6.QtCore.QMetaObject("CamDirectorDialog" inherits "QDialog":)

class videof2b.ui.camera_director_dialog.**CamDirectorInputsModel**(*biz_obj*, *parent=None*)

Bases: QAbstractListModel

Model that represents the cam director's inputs.

data(*index*, *role=Ellipsis*)

Return type
Any

flags(*index*)

Return type
~qtpy.QtCore.

headerData(*section*, *orientation*, *role=Ellipsis*)

Return type
Any

rowCount(*parent=Ellipsis*)

Return type
int

setData(*index*, *value*, *role=Ellipsis*)

Return type
bool

staticMetaObject = PySide6.QtCore.QMetaObject("CamDirectorInputsModel" inherits "QAbstractListModel":)

class videof2b.ui.camera_director_dialog.**CamDirectorResultsModel**(*biz_obj*, *parent=None*)

Bases: QAbstractTableModel

Model that represents the cam director's results.

columnCount(*parent=Ellipsis*)

Return type
int

data(*index*, *role=Ellipsis*)

Return type
Any

flags(*index*)

Return type
~qtpy.QtCore.

headerData(*section*, *orientation*, *role=Ellipsis*)

Return type
Any

rowCount(*parent=Ellipsis*)

Return type
int

staticMetaObject = PySide6.QtCore.QMetaObject("CamDirectorResultsModel" inherits "QAbstractTableModel":)

videof2b.ui.exception_dialog module

The exception dialog form.

class videof2b.ui.exception_dialog.**ExceptionDialog**(*exc_msg*)

Bases: QDialog, *StoreProperties*

User-friendly exception dialog.

exec()

Show the dialog.

on_attach_file_button_clicked()

Attach files to the bug report e-mail.

on_description_updated()

Update the minimum number of characters needed in the description.

on_save_report_button_clicked()

Save exception log and system information to a file.

setup_ui()

Set up the UI.

staticMetaObject = PySide6.QtCore.QMetaObject("ExceptionDialog" inherits "QDialog":)

videof2b.ui.icons module

Provides icons for the VideoF2B application.

class videof2b.ui.icons.**MyIcons**(**args*, ***kwargs*)

Bases: object

Provide application-wide icons.

videof2b.ui.load_flight_dialog module

The dialog that loads the input video.

class videof2b.ui.load_flight_dialog.**LoadFlightDialog**(*parent*)

Bases: QDialog, *StoreProperties*

The dialog window that collects user inputs for a Flight instance.

accept()

Create a Flight instance if all inputs are valid. Store the instance as our *flight* attribute.

Return type
None

on_skip_locate_changed()

Enable/disable the measurement widgets when the "skip locate" checkbox changes.

setup_ui()

Lay out the UI elements

staticMetaObject = PySide6.QtCore.QMetaObject("LoadFlightDialog" inherits "QDialog":)

videof2b.ui.main_window module

The main GUI window of VideoF2B application.

class videof2b.ui.main_window.**MainWindow**

Bases: QMainWindow, *UIMainWindow*, *StoreProperties*

The main UI window of the VideoF2B application.

clear_track

closeEvent(*event*)

Overridden to handle the closing of the main window in a safe manner. Handles all exit/close/quit requests here, ensuring all threads are stopped before we close.

figure_diags_changed

figure_mark

figure_state_changed

load_settings()

Load the settings of MainWindow.

locating_completed

maneuver_endpts_changed

manipulate_sphere

on_ar_geometry_available(*is_available*)

Update UI controls based on availability of AR geometry.

on_cal_finished(*retcode*)

Handle return codes and any possible exceptions that are reported by CameraCalibrator when its processing loop finishes. Update the UI as appropriate.

on_cal_thread_finished()

Handle cleanup when the calibrator thread finishes.

on_calibrate_cam()

Calibrate a camera via a specified video file.

on_chk_diag_changed()

Tell the processor to toggle the drawn state of figure diagnostics.

on_chk_endpts_changed()

Tell the processor to toggle the drawn state of maneuver start/end points.

on_chk_figure_changed()

Tell the processor to toggle the drawn state of a figure.

on_clear_track()

Clear the aircraft's existing flight track.

on_figure_end()

Mark the end of a figure in 3D.

on_figure_start()

Mark the start of a figure in 3D.

on_help_about()

Display About window.

on_load_flight()

Loads a flight via LoadFlightDialog and starts processing it.

on_loc_pts_changed(*points*, *msg*)

Echoes changes in the VideoProcessor's locator points and updates the instruction message.

on_loc_pts_defined()

Present the user with a confirm/redefine choice via messagebox.

on_locating_started()

Prepare UI for the camera locating procedure.

on_move_east()

Move AR sphere East.

on_move_north()

Move AR sphere North.

on_move_reset()

Reset AR sphere's center to world origin.

on_move_south()

Move AR sphere South.

on_move_west()

Move AR sphere West.

on_next_figure()

Advance the current figure checkbox to next figure if appropriate.

Behavior: If multiple figures are checked, do nothing. If the last figure is checked, do nothing. If no figures are checked, check the first one. In all other cases, uncheck the current figure and check the next one.

on_pause_resume()

Pause/resume processing at the current frame.

on_paused_resumed(*is_paused*)

Slot that responds to VideoProcessor's signal.

on_place_cam()

Open the camera placement dialog.

on_proc_finished(*retcode*)

Handle return codes and any possible exceptions that are reported by VideoProcessor when its processing loop finishes. Also update the UI as appropriate.

on_proc_starting()

Handle required preparations when the processing thread starts.

on_proc_thread_finished()

Handle cleanup when the processing thread finishes.

on_progress_updated(*data*)

Display video processing progress.

on_relocate_cam()

Relocate the camera.

on_restart_flight()

Reload the current flight and restart it.

on_rotate_ccw()

Rotate AR sphere CCW.

on_rotate_cw()

Rotate AR sphere CW.

on_stop_proc()

Request to stop the video processor.

on_track_cleared()

Slot that responds to the processor's signal.

pause_resume

relocate_cam

save_settings()

Save the settings of MainWindow.

start_cal_thread()

Starts the calibrator on a worker thread.

start_proc_thread()

Starts the video processor on a worker thread.

staticMetaObject = PySide6.QtCore.QMetaObject("MainWindow" inherits "QMainWindow": Methods: #40 type=Signal, signature=stop_processor() #41 type=Signal, signature=locating_completed() #42 type=Signal, signature=clear_track() #43 type=Signal, signature=figure_state_changed(PyObject,bool), parameters=PyObject, bool #44 type=Signal, signature=figure_diags_changed(bool), parameters=bool #45 type=Signal, signature=maneuver_endpts_changed(bool), parameters=bool #46 type=Signal, signature=relocate_cam() #47 type=Signal, signature=manipulate_sphere(PyObject), parameters=PyObject #48 type=Signal, signature=figure_mark(bool), parameters=bool #49 type=Signal, signature=pause_resume())

stop_processor

class videof2b.ui.main_window.**UIMainWindow**

Bases: object

Define the UI layout.

setup_ui(*main_window*)

Create the UI here.

videof2b.ui.style module

Define UI styles.

videof2b.ui.video_window module

The video window of VideoF2B application.

class videof2b.ui.video_window.**VideoWindow**(*parent*, ***kwargs*)

Bases: QLabel

The window that displays video frames during processing.

clear()

Overridden method to clear our custom pixmap.

Return type
None

property is_mouse_enabled

Indicates whether the video window reacts to mouse events.

mousePressEvent(*event*)

Overridden event so that we react to mouse clicks as needed.

Return type
None

point_added

point_removed

resizeEvent(*event*)

Overridden event so that our window resizes with its parent while maintaining the loaded image's original aspect ratio.

Return type
None

staticMetaObject = PySide6.QtCore.QMetaObject("VideoWindow" inherits "QLabel": Methods: #47 type=Signal, signature=point_added(PyObject), parameters=PyObject #48 type=Signal, signature=point_removed(PyObject), parameters=PyObject)

update_frame(*frame*)

Set a new video frame in the window.

Return type
None

videof2b.ui.widgets module

This module contains various custom widgets for the UI.

class videof2b.ui.widgets.**FileDialog**

Bases: QFileDialog

A wrapped QFileDialog compatible with Path objects.

classmethod getExistingDirectory(**args*, ***kwargs*)

Thin wrapper of *getExistingDirectory* compatible with Path objects as args.

Return type
pathlib.Path

classmethod getOpenFileName(**args*, ***kwargs*)

Thin wrapper of *getOpenFileName* compatible with Path objects as args.

Return type
tuple[pathlib.Path, str]

classmethod getOpenFileNames(**args*, ***kwargs*)

Thin wrapper of *getOpenFileNames* compatible with Path objects as args.

Return type
tuple[list[pathlib.Path], str]

classmethod getSaveFileName(**args*, ***kwargs*)

Thin wrapper of *getSaveFileName* compatible with Path objects as args.

Return type
tuple[pathlib.Path | None, str]

staticMetaObject = PySide6.QtCore.QMetaObject("FileDialog" inherits "QFileDialog":
)

class videof2b.ui.widgets.**PathEdit**(*parent=None*, *path_type=PathEditType.FILES*, *caption=None*, *initial_path=None*)

Bases: QWidget

Custom QWidget subclass for selecting a file or directory.

on_browse_button_clicked()

Shows the QFileDialog when the browse button is clicked. Emits *path_changed* if appropriate.

Return type
None

on_line_edit_editing_finished()

Updates *path* and emits *path_changed* when the line edit has finished being edited.

Return type
None

on_new_path(*path*)

If the given path is different from current *path*, updates *path* and emits the *path_changed* Signal.

Parameters
path (*Path*) – The new path

Return type
None

property path

Returns the selected path.

Returns
The selected path

Return type
Path

path_changed

property path_type

Returns the path type. Path type specifies selecting a file or directory.

Returns
The type selected

Return type
PathType

staticMetaObject = PySide6.QtCore.QMetaObject("PathEdit" inherits "QWidget":
Methods: #34 type=Signal, signature=path_changed(PyObject), parameters=PyObject)

update_button_tool_tips()

Updates the button tooltips during init and when *path_type* changes.

Return type
None

class videof2b.ui.widgets.**PathEditType**(*value*)

Bases: Enum

Specifies the type of browser in a PathEdit.

DIRECTORIES = 2

FILES = 1

class videof2b.ui.widgets.**QHLine**

Bases: QFrame

A horizontal line widget.

staticMetaObject = PySide6.QtCore.QMetaObject("QHLine" inherits "QFrame":)

class videof2b.ui.widgets.**QVLine**

Bases: QFrame

A vertical line widget.

staticMetaObject = PySide6.QtCore.QMetaObject("QVLine" inherits "QFrame":)

Module contents

The ui module provides the core user interface for VideoF2B

5.1.2 Submodules

5.1.3 videof2b.app module

Main VideoF2B application.

class videof2b.app.**VideoF2B**

Bases: QObject

The main application runner for VideoF2B.

hook_exception(*exc_type*, *value*, *traceback*)

Add an exception hook so that any uncaught exceptions are displayed in this window rather than someplace where users cannot see it and cannot report when we encounter these problems.

Parameters

- **exc_type** – The class of exception.
- **value** – The actual exception object.
- **traceback** – A traceback object with the details of where the exception occurred.

run(*app*)

The main method. Makes the necessary preparations, then runs the given app.

static set_busy_cursor()

Sets the Busy Cursor for the Application.

static set_normal_cursor()

Sets the Normal Cursor for the Application.

staticMetaObject = PySide6.QtCore.QMetaObject("VideoF2B" inherits "QObject":)

videof2b.app.**start**()

Programmatic entry point of VideoF2B.

5.1.4 Module contents

The *videof2b* package contains all VideoF2B functionality.

PYTHON MODULE INDEX

V

INDEX

D

E

F

G

H

I

L

M

N

O

P

Q

R

S

T

U

V

W

Y