Camera Data Annotation Interface: Documentation

Table of Contents

- Views
  - Main Window
  - Timeline View
- Annotating & Labeling
  - Annotations
  - Clickable Label
  - Label Area
- Dialog Boxes
  - Import Dialog Box
  - Export Dialog Box
  - Settings Dialog Box
- Resizing
  - Resizable Rectangle Item
  - Resizable Rectangle Item Settings
  - Resize Directions

Last Updated: 12/14/2021

annotation.h/annotation.cpp

void setClassName(QString _class_name);

Changes name of annotation labels

QString getClassName();

Returns label name

void setColor(QColor _color);

Changes color of annotation boxes

QColor getColor();

Returns color

void printList(); *Not used*

Prints to the console the addresses of the stored QGraphicItems under this specific annotation class

void addBand(QGraphicsItem* box);

Appends *box* to the stored QGraphicsItem list

void removeBand(QGraphicsItem* box);

Removes *box* from the stored QGraphicsItem list

QGraphicsItem* removeBandAt(int i);

Removes QGraphicsItem from the stored list at index *i*

QGraphicsItem* getBandAt(int i);

Returns QGraphicsItem from the stored list at index *i*

QList<QGraphicsItem*> getList();

Returns the QGraphicsItem list

QString whoAmI(QGraphicsItem* box);

Checks and returns the class_name if *box* is found within the QGraphicsItem list. Returns an empty QString if it is not found

clickablelabel.h/clickablelabel.cpp

void setFrameNum(int _frame_num);

Changes frame number

int getFrameNum();

Returns frame number

void setImagePath(QString _image_path);

Changes system path to image

QString getImagePath();

Returns system path to image

void clicked();

clicked() signal

void mousePressEvent(QMouseEvent* event);

Emits clicked() signal on mouse press

exportdialog.h/exportdialog.cpp

QString get_path();

Returns path to user selected directory

void on_pushButton_clicked();

When pushButton is clicked a QFileDialog will pop up allowing the user to browse their system's file system and select a directory to export to. Selected directory will be set to path.

importdialog.h/importdialog.cpp

QString get_path();

Returns system path to user selected ros bag file

QString get_topic();

Returns the currently selected topic from the ui's comboBox

void on_pushButton_clicked();

When pushButton is clicked a QFileDialog will pop up allowing the user to browse their system's file system and select a file to import from. The file name will then be displayed to the ui and the system path stored to path. A QProcess then runs on the script "topic_extraction.py" using parameters path to extract the topic parameters to a file.This file is then read and the topics are added to the ui's comboBox.

void on_buttonBox_accepted();

Closes the import ui

void on_buttonBox_rejected();

Closes the import ui

QString get_box_path();

Returns system path to user selected file containing bounding box data

void on_pushButton_2_clicked();

When pushButton is clicked a QFileDialog will pop up allowing the user to browse their system's file system and select a file to pull bounding box data from. The selected file's system path is stored in box_path and the file name is displayed to the ui.

labelarea.h/labelarea.cpp

QGraphicsScene* setupScene();

Used for setting a QGraphicScene's initial settings. Sets default size to 600x800 and adds the class's eventfilter. Finally a cross patterned black grid is added to the back of the scene.

void displayImg(const QImage &image, QGraphicsScene* _scene);

Takes in QImage and scene. Creates a QGraphicsPixmapItem from the QImage and adds it to the scene. The scene is then scaled to the image's size and its view is centered on the image.

bool eventFilter(QObject *watched, QEvent *event);

Used for tracking mouse events. Starts by setting the flags of all ResizableRectItems to false if in draw mode or true if not. This prevents selecting a box while in draw mode. Then checks if the mouse is pressed, and if so, stores the point where clicked relative to the scene position. Next it checks for a mouse release event, and if so draws a box. This is done by first checking if it is in draw mode, and If not, emits a boxMoved() signal. Next it calculates the bounding rect of the drawn rectangle using the point stored on mouse click and the point on mouse release. Finally the ResizableRectItemSettings and passed in to a new ResizeableRectItem along with the drawn rectangle coordinates and is added to the current scene along with a boxMade(item) signal.

QGraphicsScene* getScene();

Returns widget's scene

void setColor(QColor _color);

Changes color to _color

QColor getColor();

Returns color

void unselectAll();

Clears the scene's selection

void setCurrentClass(QString _class);

Changes currentClass to _class

void wheelEvent(QWheelEvent *event);

Adds scroll functionality. When the user scrolls using the mouse's scroll wheel, the scene is scaled relative to the scroll event.

void boxMade(QGraphicsItem* band);

BoxMade(QGraphicsItem* band) signal

Void boxMoved()

boxMoved() signal

mainwindow.h/mainwindow.cpp

void on_actionImport_triggered();

Calls import dialog gui, it's getters to obtain file path and parameters
Calls extraction script to get .bag location and image destination
Sets up annotation tree for inclusion of a formatted .txt file
Draws images to the scenes in the frame viewer and timeline
Readds bounding boxes specified in .txt file to the scenes

void on_actionExport_triggered();

Calls export gui
Creates a file at specified location that includes image number and the specifics for each of its boxes, including its class label, its position, and dimensions
Changes to the export format can be done here

void on_newclassButton_clicked();

Creates and adds a new Annotation object and appends it to the class_list
New Annotations have a "New_" name and are assigned a random color
Class list is updated to show the new class under already created ones
Current label is changed to the newly created class

void on_nextimgButton_clicked();

Find next timelineView and pass it to update_timeline()

void on_previmgButton_clicked();

Find previous timelineView and pass it to update_timeline()

void processExit(int , QProcess::ExitStatus);

No longer in use.
Acts as a slot function that detects ExitStatus from a QProcess to allow for asynchronous script running.

void update_timeline(QWidget*);

Cast passed in Widget to timelineView
Change stylesheet to highlight newly selected scene on the timeline
Check if the newly selected scene is 1 after the previous scene, in which case copyBoxes() is called.
See in the newly selected scene is after or before the previous scene
Make sure newly selected scene is not one of the first 2 or last 3 scenes, then scale the scrollbar by 171 * newly selected scene's frame number-2
(171 is used as it scales well with the size of the timeline)

void on_classTreeWidget_currentItemChanged(QTreeWidgetItem *current, QTreeWidgetItem *previous);

Called when when another item other than the one currently selected on the class tree is selected
If the item is a class then the current label is changed and drawing color is updated to reflect the current class
If the item refers to a box, the QGraphicsItem is highlighted on the current scene

void on_classTreeWidget_itemChanged(QTreeWidgetItem *item, int column);

Called when the currently selected item is edited
Updates the associated annotation class based on the changes made to the item

void on_classTreeWidget_itemDoubleClicked(QTreeWidgetItem *item, int column);

Unused
Console debug

void on_labelarea_boxMade(QGraphicsItem* band);

Called when a QGraphicsItem is added to the current scene
Attaches the item to the current label by appending the item to the annotation list
Class tree is updated to include the item underneath its associated label
Creates a ResizableRectItem associated with the GraphicsItem and sets its parameters with the annotation's assigned color

void on_deleteclassButton_clicked();

Called when the delete button is pressed
Checks that the current class tree item refers to a QGraphicsItem
Deletes the item updates the class tree list
Does nothing otherwise

void on_labelarea_selectionChanged();

Disconnected signal slot
Could be called to update the current selected class tree item based on which QGraphicsItem is selected

void on_drawToolButton_clicked();

Changes labelarea's DragMode to RubberBandDrag

Sets stylesheet of buttons to highlight the draw button

void on_selectionToolButton_clicked();

Changes labelarea's DragMode to NoDrag
Sets stylesheet of buttons to highlight the tool button

void on_moveToolButton_clicked();

Changes labelarea's DragMode to ScrollHandDrag
Sets stylesheet of buttons to highlight the tool button

void on_editclassButton_clicked();

Create and initiate settingsdialog instance
Update all scene's boxes to the new color from the settingsdialog
Update all timeline's scene's boxes to the new color from the settingsdialog

void draw_picture(QString,int);

Sets the labelarea scene to selected scene
Create connection between boxMade() and on_labelarea_boxMade()
Create connection between boxMoved() and on_labelarea_boxMoved()
Call updateTreeList() to update the tree list to the new scene

void timeline(QString directory);

Get number of images in directory
Setup all of the QGraphicScenes
Initialize QSignalMapper to create connection between slot() and update_timeline()
For each img:
Initialize new timelineView
Get next image in directory and scale in to 165x165
Initialize new QGraphicsScene
Create QGraphicsPixmapItem, add it to scene, and scale the view to the image
Create connection between clicked() signal and QSignalMapper
Set view's frame_number and image_path
Add view to QHBoxLayout
Once all views are added, add QHBoxLayout to scrollArea on the ui

void get_num_images(QString directory);

Return number of images located in given directory

Annotation* getAnnotationFromCurrentClass();

Return Annotation of currently selected class

Annotation* getAnnotationFromClass(QString className);

Return Annotation with the same name as className

QTreeWidgetItem* findItemInList(QGraphicsItem* item);

Searches the class list and the annotation lists for *item* and returns the associated class tree item

void readClassList();

Puts to console the current name of all the labels

void keyPressEvent(QKeyEvent *e);

If S clicked -> set labelarea DrawMode to ScrollHandDrag and forward to slot function
If A clicked -> set labelarea DrawMode to NoDrag and forward to slot function
If D clicked -> set labelarea DrawMode to RubberBandDrag and forward to slot function

void setupGraphicScenes();

Create scenes using labelarea's setupScene()

resizablerectitem.h/resizablerectitem.cpp

void changeColor(QColor _color);

Change color to _color

void setClass(QString _class_name);

Change class_name to _class_name

QString getClass();

Returns class_name

void mousePressEvent(QGraphicsSceneMouseEvent *event);

Get the resize-directions. Then, if not a resize event, pass it to base class so the move event can be implemented. ELse, resizeRect()

void mouseMoveEvent(QGraphicsSceneMouseEvent *event);

If not a resize event, pass it to the base class so the move event can be implemented. Else, resizeRect()

void mouseReleaseEvent(QGraphicsSceneMouseEvent *event);

If not a resize event, pass it to the base class so the move event can be implemented. Else, resizeRect()

void paint(QPainter *painter, const QStyleOptionGraphicsItem *option, QWidget *widget);

Paint override to allow the inner-rect to be drawn before the outer-rect

QRectF getInnerRect() const;

Returns rect() adjusted to settings resizableBorderSize

void resizeRect(QGraphicsSceneMouseEvent *event);

Resize based on QGraphicsSceneMouseEvent

resizablerectitemsettings.h/resizablerectitemsettings.cpp

void validateRect(QRectF *r, const ResizeDirections &resizeDirections) const;

Enforcement of the minimum and maximum sizes

resizedirections.h

bool any();

Return either horizontal or vertical enum

settingsdialog.h/settingsdialog.cpp

void on_pushButton_color_clicked();

When pushButton_color is clicked, prompt a QColorDialog that lets the user select a color. This color is then set to the current select class along with the current set alpha. Finally, the color code is displayed to the ui.

void on_comboBox_class_currentIndexChanged(int index);

Display newly selected class's color and alpha to the ui.

void on_buttonBox_accepted();

If accepted, update the selected class with the currently selected alpha value then close the ui.

Bool get_checked();

Return the checkstate of the checkBox

timelineview.h/timelineview.cpp

void resizeEvent(QResizeEvent *event);

Override of the QGraphicsView resizeEvent to fit the view to its QGraphicPixmapItem.

void setFrameNum(int _frame_num);

Change frame_num to _frame_num

int getFrameNum();

Returns frame_num.

void setImagePath(QString _image_path);

Change image_path to _image_path.

QString getImagePath();

Returns image_path.

void clicked();

clicked() signal

void mousePressEvent(QMouseEvent* event);

Emit clicked() signal on mousePressEvent