Social Groups Detection

Silvia-Laura Pintea *(6109969)*

<S.L.Pintea@student.uva.nl>

Contents

Chapter 1

Data Structure Documentation

1.1 annotationsHandle::ANNOTATION Struct Reference

A structure that stores a single annotation for a specific person.

Data Fields

- short int **id**
- cv::Point **location**
- vector< unsigned int > **poses**

1.2 annotationsHandle Class Reference

Class for annotating both positions and poses of the people in the images.

Data Structures

- struct ANNOTATION

  A structure that stores a single annotation for a specific person.

- struct ASSIGNED

  Shows which id from the old annotations is assigned to which id from the new annotations based on what minimal distance.

- struct FULL_ANNOTATIONS

  Structure containing a vector of annotations for each image.

Public Types

- enum POSE { SITTING, STANDING, BENDING, ORIENTATION }

  All considered poses.

Static Public Member Functions

- static void mouseHandlerAnn (int event, int x, int y, int flags, void ∗param)

  Mouse handler for annotating people's positions and poses.

- static void showMenu (cv::Point center)

  Draws the "menu" of possible poses for the current position.

- static void plotHull (IplImage ∗img, vector< CvPoint > &hull)

  Plots the hull indicated by the parameter `hull` on the given image.

- static int runAnn (int argc, char ∗∗argv)

  Starts the annotation of the images.

- static void trackbar_callback (int position, void ∗param)

  The "on change" handler for the track-bars.

- static void trackBarHandleFct (int position, void ∗param)

  A function that starts a new thread which handles the track-bar event.

- static void loadAnnotations (char ∗filename, vector< FULL_ANNOTATIONS > &loadedAnno)

  Load annotations from file.

- static void annoDifferences (vector< FULL_ANNOTATIONS > &train, vector< FULL_ANNOTATIONS > &test, double &avgDist, double &Ndiff, double avgOrientDiff, double poseDiff)

  Computes the average distance from the predicted location and the annotated one, the number of unpredicted people in each image and the differences in the pose estimation.

- static void correltateLocs (vector< ANNOTATION > &annoOld, vector< ANNOTATION > &annoNew, vector< ASSIGNED > &idAssignedTo)

  Correlate annotations' from locations in `annoOld` to locations in `annoNew` through IDs.

- static bool canBeAssigned (vector< ASSIGNED > &idAssignedTo, short int id, double newDist, short int to)

  Checks to see if a location can be assigned to a specific ID given the new distance.

- static void displayFullAnns (vector< FULL_ANNOTATIONS > &fullAnns)

  Displays the complete annotations for all images.

- static int runEvaluation (int argc, char ∗∗argv)

  Starts the annotation of the images.

- static void drawOrientation (cv::Point center, unsigned int orient)

  Shows how the selected orientation looks on the image.

Static Protected Attributes

- static IplImage ∗ image

  The currently processed image.

- static vector< ANNOTATION > **annotations**
- static char choice

  Indicates if the pose was defined for the current frame.
- static boost::mutex trackbarMutex

  A mutex for controlling the access to the annotations.

1.2.1 Member Function Documentation

1.2.1.1 int runAnn (int *argc,* char ** *argv*) [static]

The parameters that need to be indicated are:

- argv[1] -- name of directory containing the images
- argv[2] -- the file contains the calibration data of the camera
- argv[3] -- the file in which the annotation data needs to be stored

1.2.1.2 int runEvaluation (int *argc,* char ** *argv*) [static]

The parameters that need to be indicated are:

- argv[1] -- train file with the correct annotations;
- argv[2] -- test file with predicted annotations;

1.2.2 Field Documentation

1.2.2.1 image [static, protected]

An instance of the structure ANNOTATIONS storing the annotations for each image.

1.3 annotationsHandle::ASSIGNED Struct Reference

Shows which id from the old annotations is assigned to which id from the new annotations based on what minimal distance.

Data Fields

- short int **id**
- short int **to**
- double **dist**

1.4 classifyImages Class Reference

Public Member Functions

- **classifyImages** (int argc, char ∗∗argv)
- void createData (std::vector< std::string > options)

  Creates the training data/test data.

- void classifySVM ()

  Regression SVM classification.

Protected Attributes

- featureDetector ∗ testFeatures

  An instance of `featureDetector` class.

- featureDetector ∗ trainFeatures

  An instance of `featureDetector` class.

- cv::Mat trainData

  The training data matrix.

- cv::Mat testData

  The test data matrix.

- std::string trainFolder

  The folder containing the training images.

- std::string testFolder

  The folder containing the test images.

1.5 featureDetector Class Reference

Class used for detecting useful features in the images that can be later used for training and classifying.

Data Structures

- struct people

  Structure containing images of the size of the detected people.

Public Types

- enum FEATURE {
  BLOB, ELLIPSE, CORNER, EDGES,
  GABOR, SURF }

  All available feature types.

Public Member Functions

- **featureDetector** (int argc, char ∗∗argv)
- **featureDetector** (int argc, char ∗∗argv, bool plot)
- void upperLowerROI (featureDetector::people someone, double variance, cv::Mat &upperRoi, cv::Mat &lowerRoi)

  Function that gets the ROI corresponding to a head/feet of a person in an image.

- bool doFindPerson (unsigned imgNum, IplImage ∗src, const vnl_vector< FLOAT > &imgVec, vnl_vector< FLOAT > &bgVec, const FLOAT logBGProb, const vnl_vector< FLOAT > &logSumPixelBGProb)

  Overwrites the `doFindPeople` function from the `Tracker` class to make it work with the feature extraction.

- bool imageProcessingMenu ()

  Simple "menu" for skipping to the next image or quitting the processing.

- void gaussianKernel (cv::Mat &gauss, cv::Size size, double sigma, cv::Point offset)

  Creates a symmetrical Gaussian kernel.

- void allForegroundPixels (std::vector< featureDetector::people > &allPeople, std::vector< unsigned > existing, IplImage ∗bg, double threshold)

  Get the foreground pixels corresponding to each person.

- double getDistToTemplate (int pixelX, int pixelY, std::vector< CvPoint > templ)

  Gets the distance to the given template from a given pixel location.

- bool isInTemplate (unsigned pixelX, unsigned pixelY, std::vector< CvPoint > templ)

  Checks to see if a given pixel is inside a template.

- void showROI (cv::Mat image, cv::Point top_left, cv::Size ROI_size)

  Shows a ROI in a given image.

- void getLinePerpendicular (cv::Point A, cv::Point B, cv::Point C, double &m, double &b)

  Get perpendicular to a line given by 2 points A, B in point C.

- bool sameSubplane (cv::Point test, cv::Point point, double m, double b)

  Checks to see if a point is on the same side of a line like another given point.

- void getCornerPoints (std::vector< cv::Point2f > &corners, cv::Mat image)

  Gets strong corner points in an image.

- void getEdges (cv::Mat_< uchar > &edges, cv::Mat image)

  Gets the edges in an image.

- void getSURF (std::vector< float > &descriptors, cv::Mat image)

  SURF descriptors (Speeded Up Robust Features).

- void blobDetector (cv::Mat &feature, cv::Mat image, std::vector< unsigned > borders)

  Blob detector in RGB color space.

- void showZoomedImage (cv::Mat image, std::string title="zoomed")

  Just displaying an image a bit larger to visualize it better.

- void skinEllipses (cv::RotatedRect &finalBox, cv::Mat img, cv::Point templateCenter, cv::Point offset=cv::Point(0, 0), double minHeadSize=20, double maxHeadSize=40)

  Head detection by fitting ellipses (if templateCenter is relative to the img the offset needs to be used).

- void getGabor (cv::Mat &response, cv::Mat image, float ∗params=NULL)

  Convolve an image with a Gabor filter with the given parameters and returns the response image.

- void setFeatureType (FEATURE type)

  Set what kind of features to extract.

- void extractDataRow (std::vector< unsigned > existing, IplImage ∗bg)

  Creates on data row in the final data matrix by getting the feature descriptors.

- void templateWindow (cv::Size imgSize, unsigned &minX, unsigned &maxX, unsigned &minY, unsigned &maxY, std::vector< CvPoint > &templ, unsigned tplBorder=100)

  Returns the size of a window around a template centered in a given point.

- void init (std::string dataFolder)

  Initializes the parameters of the tracker.

Protected Attributes

- bool plotTracks

  If it is true it displays the tracks of the people in the images.

- FEATURE featureType

  Can have one of the values indicating the feature type.

- cv::Mat data

  The training data obtained from the feature descriptors.

1.5.1 Member Function Documentation

1.5.1.1 void getGabor (cv::Mat & *response*, cv::Mat *image*, float ∗ *params* = ***NULL***)

Convolves an image with a Gabor filter with the given parameters and returns the response image.

1.6 annotationsHandle::FULL_ANNOTATIONS Struct Reference

Structure containing a vector of annotations for each image.

Data Fields

- string **imgFile**
- vector< ANNOTATION > **annos**

1.7 featureDetector::people Struct Reference

Structure containing images of the size of the detected people.

Data Fields

- cv::Point **absoluteLoc**
- cv::Point **relativeLoc**
- std::vector< unsigned > **borders**
- cv::Mat_< cv::Vec3b > **pixels**

Index