SE 3XA3: MIS
OpenCameraRefined

Team #211, CAMERACREW
Faisal Jaffer, jaffem1
Dominik Buszowiecki, buszowid
Pedram Yazdinia, yazdinip
Zayed Sheet, sheetz

April 7, 2020

Table 1: **Revision History**

Date	Version	Notes
March 14, 2020	1.0	Initial document
April 4, 2020	2.0	Filter Controller Updated

Gesture Controller Module

Module

GestureController

Uses

TensorflowObjectDetectionAPI
ClassifierConstants
CameraController
Filter
Classifier

Syntax

Exported Constants

N/A

Exported Types

GestureController = ?

Exported Access Programs

Routine name	In	Out	Exceptions
GestureController		GestureController	classifier_initialize_failure
processImage			
setFrame	$byte[]$		
captureImage			
showFilter			

Semantics

State Variables

imageFrame: $byte[]$ # current camera frame
classifier: $Classifier$ # TensforFlow model
filter: $Filter$

recognition: *Recognition*[] # classification on the current frame

Environment Variables

None

State Invariant

None

Assumptions

- The mathematical operator \ represents integer division. For example $8\backslash 5 = 1$.

Access Routine Semantics

GestureController():

- transition: $classifier, filter := Classifier(ClassifierConstants.getInferenceName(),$
  $ClassifierConstants.getLabelName()), Filter()$

- output: $out := self$

- exception: $exc := ((ClassifierConstants.getInferenceName() == null |$
  $ClassifierConstants.getLabelName() == null) \Rightarrow classifier\_initialize\_failure)$

processImage():

- transition: $recognition := classifier.recognizeImage(bitmapFromByte(imageFrame)) \Rightarrow$
  $(recognition.title == ClassifierConstants.Smile \rightarrow captureImage() | recognition.title ==$
  $ClassifierConstants.Thumb \rightarrow filter.changeFilter())$

- output: None

- exception: None

setFrame(*byte*[] data):

- transition: imageFrame:= data

- output: None

- exception: None

captureImage():

- transition: *CameraController.captureImage*()

- output: None

- exception: None

showFilter():

- transition: None

- output: None

- exception: None

Local Functions

bitmapFromByte(*byte*[] data):

- transition: Matrix mat := ImageUtils.convertYUV420SPToARGB8888(data)

- output: *out* := mat

- exception: None

Recognition Module

Module

Recognition

Uses

RectF

Syntax

Exported Constants

N/A

Exported Types

Recognition = ?

Exported Access Programs

Routine name	In	Out	Exceptions
Recognition	$String, String, \mathbb{Q}, RectF$	Recognition	
getID		$String$	
getTitle		$String$	
getConfidence		$\mathbb{Q}$	
getLocation	$Bitmap$	$RectF$	

Semantics

State Variables

id: $String$ # unique id assignment since multiple recognitions possible
title: $String$ # label of the recognition
confidence: $\mathbb{Q}$ # confidence level of the classification
location: RectF # pixel location of the recognition

Environment Variables

None

State Invariant

None

Assumptions

- Invalid arguments will not be provided into the Recognition and getLocation routines.

Access Routine Semantics

Recognition(id, title, confidence, location):

- transition: $id, title, confidence, location := id, title, confidence, location$

- output: $out := self$

- exception:None

getID():

- transition: None

- output: $out := id$

- exception: None

getTitle():

- transition: None

- output: $out := title$

- exception: None

getConfidence():

- transition: None

- output: $out := confidence$

- exception: None

getLocation():

- transition: None

- output: $location$

- exception: None

Local Functions

None

Classifier Module

Module

Classifier

Uses

Recognition
ClassifierConstants
Bitmap
TF

Syntax

Exported Constants

N/A

Exported Types

None

Exported Access Programs

Routine name	In	Out	Exceptions
Classifier	$String, String, (\mathbb{Z}, \mathbb{Z})$	Classifier	classifier_initialize_failure
recognizeImage	$Bitmap$	$Recognition$	

Semantics

State Variables

modelFilename: $String$ # link to the model inference
labelFilename: $String$ # link to the model labels
inputSize: $(\mathbb{Z}, \mathbb{Z})$ # size of model input
model: TF.model# stored model

Environment Variables

None

State Invariant

None

Assumptions

- Invalid arguments will not be provided to the Classifier and recognizeImage routines.

Access Routine Semantics

Classifier(modelFilename, labelFilename, inputSize):

- transition: $model := newTF.model(modelFilename, labelFilename)$

- output: $out := self$

- exception: $exc := newTF.model(modelFilename, labelFilename) == null \Rightarrow (classifier\_initialize\_failure)$

recognizeImage(b):

- transition: None

- output: $out := model.detect(b)$

- exception: None

Local Functions

None

Classifier Constants Module

Module

ClassifierConstants

Uses

Syntax

Exported Constants

inferenceName: *String*
labelName: *String*

Exported Types

None

Exported Access Programs

Routine name	In	Out	Exceptions
getInferenceName		*String*	
getLabelName		*String*	

Semantics

State Variables

Environment Variables

None

State Invariant

None

Assumptions

None

Access Routine Semantics

getInferenceName():

- transition: None

- output: $out :=$ inferenceName

- exception: None

getLabelName():

- transition: None

- output: $out :=$ labelName

- exception: None

Local Functions

None

~~Filter Module~~ ImgFilterController Module

Module

Filter

Uses

Filter Constants

Syntax

Exported Constants

N/A

Exported Types

N/A

Exported Access Programs

Routine name	In	Out	Exceptions
changeFilter			
getFilter		$\mathbb{Z}$	
setFrame	*byte*[]		
processImage			
getFiltered		*Bitmap*	

Semantics

State Variables

filterIndex : $\mathbb{Z}$ *#Which index in the FILTERS constant from the filters module is selected*
imageFrame: *byte*[] # current camera frame
rgbFrameBitmap: *Bitmap* # bitMap version of camera frame
filtered: *Bitmap* # Filtered version of rgbFrameBitmap

Environment Variables

preview: *# Android Camera Preview object*

State Invariant

filterIndex $< |FILTERS|$

Assumptions

The % operator represents the mathematical modulus operator

Access Routine Semantics

changeFilter():

- transition: filterIndex := (filterIndex + 1)%$|FILTERS + 1|$

- output: None

getFilter():

- transition: None

- output: out := filterIndex

setFrame($byte[]$ data):

- transition: imageFrame:= data

- output: None

- exception: None

processImage():

- transition:
  filterIndex $= 0 \implies$ filtered := $null$
  filterIndex $= 0 \implies$ filtered := setFiltered(rgbFrameBitmap)

- output: None

- exception: None

getFiltered():

- transition: None

- output: out := filtered

Filter Constants Module

Module

Constants

Uses

OpenCV

Syntax

Exported Constants

GRAYSCALE: ~~($\mathbb{Z}$[], $\mathbb{Z}$[])~~ ColorMatrixColorFilter
RED_FILTER: ~~($\mathbb{Z}$[], $\mathbb{Z}$[])~~ ColorMatrixColorFilter
BLUE_FILTER: ~~($\mathbb{Z}$[], $\mathbb{Z}$[])~~ ColorMatrixColorFilter
~~# The values of the filters will be obtained from the OpenCV lookup tables. They are 2xN matrices which~~
# The values of the filters are obtained from code snippets found online, the type Color-MatrixColorFilter is a type built into android
FILTERS = [GRAYSCALE, RED_FILTER, BLUE_FILTER]

Exported Types

None

Exported Access Programs

None

Semantics

State Variables

None

State Invariant

None