Left Luggage Detection Documentation

Release 0.1.1

Andrea Rizzo, Matteo Bruni

June 08, 2015

Contents

Authors:

- *Andrea Rizzo*, andrearizzo[at]outlook.com
- *Matteo Bruni*, matteo.bruni[at]gmail.com

CHAPTER 1

Abstract

This wiki describes the method used to detect abandoned items in a public space. Today, video surveillance is used airports, train stations and public spaces where it is essential guarantee a high security level. The video stream is obtained through the use of a Kinect device. The the RGB (*intensity*) and depth video streams are analyzed independently. From each stream we obtain a set of proposal, i.e. left luggage item, that are combined in the final step of our pipeline.

CHAPTER 2

Contents

2.1 Left luggage detection

This wiki describes the method used to detect abandoned items in a public space. Today, video surveillance is used airports, train stations and public spaces where it is essential guarantee a high security level. The video stream is obtained through the use of a Kinect device. The the RGB (*intensity*) and depth video streams are analyzed independently. From each stream we obtain a set of proposal, i.e. left luggage item, that are combined in the final step of our pipeline.

2.1.1 Introduction

In this section we briefly describe the proposed approach.

Image data: we use the Kinect device. Kinect sensor is a horizontal bar connected to a small base with a motorized pivot. The major device features are RGB camera and depth sensor. The device has a USB2 interface and the resolution of the RGB camera is 640×480 with 8 bit quantization. The depth camera instead has a resolution of 640×480 with 11 bit quantization.

Pipeline: our detection pipeline analyzes the RGB (*intensity*) and depth video streams independently. This means that the RGB left object proposals are found without considering the depth data and the depth proposals are found without considering the RGB data. Both sets of proposal are combined later in a processing stage. The independent processing warranted because the RGB video stream is defined everywhere, i.e. for each pixel of a stream frame the intensity value is defined, but it is liable to photometric variations. Instead the depth video stream is not defined everywhere. The depth value is only available for the image regions that are close enough to the device. Also for black objects the sensor can't measure the depth value.

By using the two video streams a background models for depth and RGB are computed. To extract left luggage proposals the spatial changes over time are accumulated in an image aggregator. For the depth aggregator we provide more than one method to accumulate the depth changes. If the aggregator exceeds a threshold is segmented with a bounding box and we mark the spatial region as left item proposal. The depth and intensity proposal are compared using the PASCAL criterion. The bounding boxes that satisfy the criterion are considered left objects.

2.1.2 Background modeling

In this section we describe the methods used to model the background. Then we describe the methods used to accumulate the spatial changes and how the aggregators are processed to extract the proposals.

Depth background model and proposals

The depth background model is computed by using an high-resolution (11-bit) depth matrix. The method used to model the background is the accumulate running average. At time t the model is updated with the following function:

$model_t = (1 - \alpha) \cdot model_{t-1} + \alpha \cdot frame_t$

where the coefficient α is the learning rate. For a proper background modelling we have to detect the holes in depth map, i.e. the pixels where the sensor didn't measure the depth. The value of these pixels is $2^{11} - 1$. Then the foreground is extracted. Since the depth image is very noisy, applying an opening is suggested.

To extract the proposal, we accumulate the spatial changes in depth. The methods provided are three.

Image accumulator

The first method is a simple **image accumulator** and it is quicker than the other methods. By using a matrix with the same size of depth frame, the pixels that are in current foreground are incremented by an unit value. Instead the pixels that were in the foreground but now are not in current foreground the correspondent entries are decremented. To generate a proposal from the accumulator we consider only the entry that have a number of observations above a threshold. The proposals are extracted by using [1]. The proposals with area less than 50 pixels are not considered.

Bounding box accumulator

The **bounding box accumulator** method is slower than *image accumulator* but is more accurate. The proposals are generated as in the previous method by using the mask of current foreground. So the current set of bounding box are compared with the set of accumulated bounding box. We consider two bounding box similar if the distance between two centers and the area ratio are under a threshold. For each bounding box that has a match the correspondent entries in the accumulator are updated with the new spatial coordinates and the counter are incremented by an unit value. For each bounding box that hasn't a match it's temporarily stored in the accumulator with counter set to 1. For each bounding box in the accumulator that hasn't a match the correspondent count are decrement. To generate a proposal from the accumulator we consider only the bounding boxes that have a number of observations above a threshold. Note that if more than one bounding box match with a bounding box in the accumulator it considers the first match found.

Best bounding box accumulator

The **best bounding box accumulator** method is the slowest. It's works as the previous method but if more than one bounding box match with a bounding box in the accumulator it updates the accumulator by using the best match found.

Intensity background model and proposals

The intensity background model is computed by using the method of Zivkovic et al [2]. The intensity-based proposal are generated with the dual foregrouonds model of Porikli et al. [3].

[1] Suzuki, S. and Abe, K., Topological Structural Analysis of Digitized Binary Images by Border Following. CVGIP 30 1, pp 32-46 (1985).

[2]

26. Zivkovic and F. van der Heijden. Efficient adaptive density estimation per image pixel for the task of background subraction. Pattern Recogn. Lett., 27(7):773–780, May 2006.

[3]

6. Porikli, Y. Ivanov, and T. Haga. Robust abandoned object detection using dual foregrounds. EURASIP J. Adv. Signal Process, 2008, Jan. 2008. 2, 3, 5

Porikli method

Briefly the Porikli method aims to detect an abandoned item. Instead of using a single background approaches the Prikli methods use two backgrounds: long-term background B_L and short-term background B_S. To compute both backgrounds the method of Zivkovic is used. For long-term background the learning rate α_L is lower than the learning rate α_S used to compute the short-term background. Therefore the B_L is more resistant against the temporary changes. In contrast, the B_S adapts to the underlying distribution faster and the changes in the scene are blended more rapidly. For each frame of video stream, the long and short term foregrounds are extracted by substracting from the current frame the background models B_L and B_S. So we obtain a long-term foreground mask F_L and a short-term foreground mask F_S. Let $I(x, y)$ be a pixel of the current frame, we have four cases:

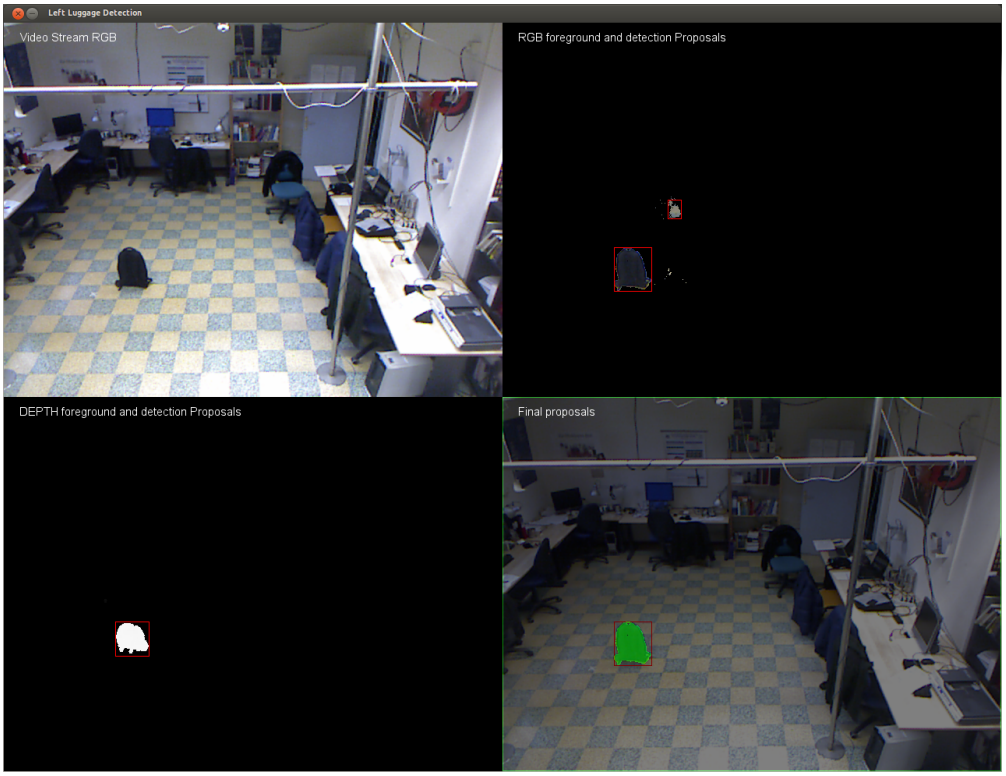
1. if $F_L(x, y) = 1$ and $F_S(x, y) = 1$ then the pixel correspond to a moving object;
2. if $F_L(x, y) = 1$ and $F_S(x, y) = 0$ then the pixel correspond to a **temporarily static object**;
3. if $F_L(x, y) = 0$ and $F_S(x, y) = 1$ then the pixel correspond to scene of background that was accluded before;
4. if $F_L(x, y) = 0$ and $F_S(x, y) = 0$ then the pixel is a background pixel for both backgrounds model.

By using the F_L and F_S an image aggregator is computed. To each pixel correspond an entry in the image aggregator. If a pixel is in F_L but is not in F_S the correspond entry in the image aggregator is increment. Otherwise the image aggregator is decremented.

2.1.3 Combination of proposals

Given the two sets of bounding box obtained through the processing of depth and intensity video streams, we compute the following pairwise overlap ratios:

$$r = \frac{area(B_{curr} \cap B_{acc})}{area(B_{curr} \cup B_{acc})}$$



A possible luggage, obtained through the formula above, is no longer detected because of two possible reasons:

- a left item is removed from the scene
- the item detected is standing still for a long amount of time. After this time the item became part of the depth and rgb background. When the item became part of the background model we can't detect its presence doing $current\_frame - bg\_model$ so we need a way to retain the information previously discovered. If a Bounding box is present at the frame t-1 but not in the frame t, we check if pixels in the area, defined by his bounding box, are still the same (i.e. the luggage is still there): this check is performed by using the normalized correlation between the pixel in the t-1 and t frames. If the similarity is above a certain threshold (i.e. 0.9) we keep drawing the old bounding box.

2.1.4 Acknowledgments

This work was supported by the Media Integration and Communication Center (MICC), Alberto Del Bimbo, Lorenzo Seidenari and Lamberto Ballan

2.2 Developer Documentation

Here you can find the developer documentation

2.2.1 Depth Processing

The depth module contains classes that hide some of all the repeated code associated with processing depth data. The main component is the *DepthProcessing* class, which is used to process continuously retrieved data from the kinect. The background model used in this class is obtained through running average via the method *DepthProcessing.update_background_model()*

Usage Example

If code already exists to retrieve the data extracting the bounding boxes proposals can be reduced to as little as the following:

```
# DepthProcessing instance
depth = DepthProcessing(IMAGE_SHAPE)

while True:
    # retrieve the depth information
    depth.current_frame = cam.get_depth_matrix()
    if first_run:
        # in first run moving average start from first frame
        depth.background_model = depth.current_frame.astype(depth.background_model.dtype)
        first_run = False

    # get depth background
    depth.update_background_model(depth.current_frame)

    # get depth foreground
    depth.extract_foreground_mask_from_run_avg(depth.current_frame)

    # apply opening to remove noise
    depth.foreground_mask = bg_models.apply_opening(depth.foreground_mask,
                                                    BG_OPEN_KSIZE,
```

```
                                                    cv2.MORPH_ELLIPSE)

    depth_proposal_bbox = depth.extract_proposal_bbox(depth.ACCUMULATOR)
```

DepthProcessing class

class depth_processing.**DepthProcessing**(*image_shape=(640, 480)*)

Depth Processing Class

extract_foreground_mask_from_run_avg(*current_frame*)

Extract depth foreground mask from running average computed substracting current_frame from background model where the difference is above BG_MASK_THRESHOLD

Parameters **current_frame** – current frame from which extract foreground

Returns binary mask with 1 for foreground and 0 for background

Return type np.int64

extract_proposal_bbox(*method=0*)

Compute bounding boxes for connected components from foreground masks that remain constant for AGG_DEPTH_MAX_E frames.

To keep track of the bounding boxes over time the function uses an aggregator depending on the method specified

Parameters **method** – method used to keep track of the bounding boxes history. Methods available are:

- *ACCUMULATOR*: to use an image accumulator for each pixel (fastest method). The bounding boxes are extracted from the pixels accumulated AGG_DEPTH_MAX_E times.
- *RECT_MATCHING/RECT_MATCHING2*: to keep track of the number of times a particular bounding box occurs over time (slower method but more accurate). Two bounding boxes in different frames are considered the same if their placement and dimension remain within a tolerance threshold.

Returns list of bounding boxes in the form of (x,y, width, height) where (x,y) is the top left corner

Return type List

Raise *NotImplementedError*: if a method different from ACCUMULATOR or RECT_MATCHING or RECT_MATCHING2 is specified

update_background_model(*current_frame*, *holes_frame=<Mock object>*)

Update depth background by running average

Parameters **current_frame** – current frame whereby update bg model

Returns background model

Return type np.float32

depth_processing.**update_depth_detection_aggregator**(*aggregator*, *foreground_current*)

Update aggregator with the provided foreground. Each pixel of the image has a value that keeps the number of times it has been seen as foreground.

Parameters

- **aggregator** – an image of uint8
- **foreground_current** – mask of the current foreground

Returns updated accumulator

2.2.2 Intensity Processing

The intensity module contains classes that hide some of all the repeated code associated with processing intensity data. The main component is the *IntensityProcessing* class, which is used to process continuously retrieved data from the kinect. The background model used in this class is obtained through Zivkovic method: Adaptive Gaussian Mixture Model. The extraction of the foreground pixels proposals is obtained via the *IntensityProcessing.compute_foreground_masks()*. The bounding boxes proposals are extracted using *IntensityProcessing.extract_proposal_bbox()* from an aggregator build using Porikli's method via *IntensityProcessing.update_detection_aggregator()*.

Usage Example

If code already exists to retrieve the data extracting the bounding boxes proposals can be reduced to as little as the following:

```
# get next video frame
rgb.current_frame = cam.get_image()
while True:
    # get rgb dual background (long and short sensitivity)
    # N.B. background is black (0) and foreground white (1)
    rgb.compute_foreground_masks(rgb.current_frame)

    # update rgb aggregator
    rgb.update_detection_aggregator()

    # extract bounding box proposals
    rgb_proposal_bbox = rgb.extract_proposal_bbox()
```

IntensityProcessing class

This module contains class for intensity processing. This class handles the rgb camera status and its methods ensure proper updates to the background models and the bounding boxes extraction.

class intensity_processing.**IntensityProcessing**(*image_shape=(640, 480)*)

compute_foreground_masks(*frame*)
Compute foreground masks for term background and short term background following Porikli's method

Parameters **frame** (*np.uint8*) – frame (3 channels) from which extract foregrounds masks

Returns foreground masks for long term and short term backgrounds

Return type np.int8

extract_proposal_bbox()
Extract RGB proposal as the bounding boxes of the areas of the accumulator that have reached a value of AGG_RGB_MAX_E

Returns list of bounding boxes

update_detection_aggregator()
Update aggregator with the provided foregrounds. If a pixel is in foreground_long but not in foreground_short increment its accumulator otherwise decrement it. If a particular area has already been detected as proposal don't decrement if the above condition is not verified.

Returns updated accumulator

Return type np.int8

2.2.3 Background Modules

bg_models.**apply_dilation**(*image*, *kernel_size*, *kernel_type*)
Apply dilation to image with the specified kernel type and image

Parameters

- **image** – image to which apply opening
- **kernel_size** – size of the structuring element
- **kernel_type** – structuring element

Returns image with opening applied

Return type np.uint8

bg_models.**apply_opening**(*image*, *kernel_size*, *kernel_type*)
Apply opening to image with the specified kernel type and image

Parameters

- **image** – image to which apply opening
- **kernel_size** – size of the structuring element
- **kernel_type** – structuring element

Returns image with opening applied

Return type np.uint8

bg_models.**compute_background_running_average**(*frame*, *average*, *alpha*, *holes_frame*)
Calculate background using running average technique new background is equal to:

$bg_{new} = (1 - alpha) * bg_{old} + alpha * frame$

Parameters

- **frame** (*np.uint16*) – current frame for background update
- **average** (*np.float32*) – background model to update
- **alpha** (*float*) – update learning rate
- **frame_holes_mask** (*np mask*) –

Returns updated background model

Return type np.float32

bg_models.**compute_foreground_mask_from_func**(*f_bg*, *current_frame*, *alpha*)
Extract binary foreground mask (1 foreground, 0 background) from f_bg background modeling function and update background model.

Parameters

- **f_bg** – background modeling function
- **current_frame** – current frame from which extract foreground
- **alpha** – update learning rate

Returns foreground mask

Return type np.uint8

bg_models.**compute_holes_mask_in_frame**(*frame*)

bg_models.**cut_foreground**(*image*, *mask*)
Cut the foreground from the image using the mask supplied

Parameters

- **image** – image from which cut foreground
- **mask** – mask of the foreground

Returns image with only the foreground

Raise *IndexError* error if the size of the image is wrong

bg_models.**get_bounding_boxes**(*image*)
Return Bounding Boxes in the format x,y,w,h where (x,y) is the top left corner

Parameters **image** – image from which retrieve the bounding boxes

Returns bounding boxes list

Return type list

bg_models.**get_bounding_boxes2**(*image*)
Return Bounding Boxes in the format x,y,w,h where (x,y) is the top left corner

Parameters **image** – image from which retrieve the bounding boxes

Returns bounding boxes array, where each element has the form (x, y, w, h, counter) with counter = 1

Return type np.array

2.2.4 Kinect Connector

class kinectconnector.**KinectConnector**(*device_number=0*)
Wrapper for the Freenect python libraries you can get_image() and get_depth() for separate channel images

get_depth()
Get the next available depth frame from the kinect, as a numpy array. Low bits in this depth are stripped so it fits in an 8-bit image channel

Returns A numpy array, shape:(640, 480)

Return type np.uint8

get_depth_matrix()
Get the next available depth frame from the kinect, as a numpy array.

Returns A numpy array, shape:(640, 480)

Return type np.uint16

get_image()
Get the next available rgb frame from the kinect, as a numpy array.

Returns A numpy array, shape:(640, 480, 3)

Return type np.uint8

2.2.5 Kinect Calibration

These are some functions to help work with kinect camera calibration and projective geometry. Tasks:

- Convert the kinect depth image to a metric 3D point cloud
- Convert the 3D point cloud to texture coordinates in the RGB image

Notes about the coordinate systems. There are three coordinate systems to worry about.

1. Kinect depth image: (u, v, depth) u and v are image coordinates,

   (0,0) is the top left corner of the image (640,480) is the bottom right corner of the image.

   Depth is the raw 11-bit image from the kinect, where 0 is infinitely far away and larger numbers are closer to the camera (2047 indicates an error pixel)

2. Kinect rgb image: (u, v) u and v are image coordinates

   (0,0) is the top left corner (640,480) is the bottom right corner

3. XYZ world coordinates: (x, y, z) The 3D world coordinates, in meters, relative to the depth camera.

   (0,0,0) is the camera center. Negative Z values are in front of the camera, and the positive Z direction points towards the camera. The X axis points to the right, and the Y axis points up. This is the standard right-handed coordinate system used by OpenGL.

calibkinect.**depth2xyzuv**(*depth*, *u=None*, *v=None*)
Return a point cloud, an Nx3 array, made by projecting the kinect depth map through intrinsic / extrinsic calibration matrices

You can provide only a portion of the depth image, or a downsampled version of the depth image if you want; just make sure to provide the correct coordinates in the u,v arguments.

Example: # This downsamples the depth image by 2 and then projects to metric point cloud u,v = mgrid[:480:2,:640:2] xyz,uv = depth2xyzuv(freenect.sync_get_depth()[::2,::2], u, v)

This projects only a small region of interest in the upper corner of the depth image u,v = mgrid[10:120,50:80] xyz,uv = depth2xyzuv(freenect.sync_get_depth()[v,u], u, v)

Parameters

- **depth** – comes directly from the kinect
- **u** – image coordinates, same size as depth (default is the original image)
- **v** – image coordinates, same size as depth (default is the original image)

Returns xyz - 3D world coordinates in meters (Nx3) uv - image coordinates for the RGB image (Nx3)

calibkinect.**uv_matrix**()
Returns a matrix you can use to project XYZ coordinates (in meters) into U,V coordinates in the kinect RGB image

Returns matrix

calibkinect.**xyz_matrix**()
Returns a matrix you can use to project U,V coordinates (in meters) into XYZ coordinates in the kinect RGB image

Returns matrix

2.2.6 Constant

const.**DEPTH_HOLE_VALUE = 2047**
Depth holes in openfreenect have maximum value in 11 bit

const.**BG_OPEN_KSIZE = 7**
Structuring element size used to apply opening

const.**BG_RUN_AVG_LRATE = 0.001**
Learning rate for running average in depth processing

const.**BG_MASK_THRESHOLD = 3**
Minimum threshold to consider a pixel foreground in running average e.g. $|pixel - average(pixel)| \geq BG\_MASK\_THRESHOLD$

const.**BG_ZIV_LONG_LRATE = 0.0005**
Background learning rate in Zivkovich method for long background model

const.**BG_ZIV_SHORT_LRATE = 0.02**
Background learning rate in Zivkovich method for short background model

const.**BG_ZIV_HIST = 1**
History for Zivkovick background method

const.**BG_ZIV_LONG_THRESH = 900**
Threshold for Zivkovich method for long background model

const.**BG_ZIV_SHORT_THRESH = 200**
Threshold for Zivkovich method for short background model

const.**AGG_RGB_MAX_E = 15**
number of frames after which a pixel is considered an left item in rgb domain

const.**AGG_RGB_PENALTY = 7**
penalty in the accumulator for a pixel not in current foreground in rgb domain

const.**AGG_DEPTH_MAX_E = 30**
number of frames after which a pixel is considered an left item in depth domain

const.**AGG_DEPTH_PENALTY = 20**
penalty in the accumulator for a pixel not in current foreground in depth domain

const.**AGG_DEPTH_BBOX = 5**
accumulator threshold for RECT_MATCHING/RECT_MATCHING2 in depth detection

const.**BBOX_MIN_AREA = 70**
minimum area in pixel to create a bounding box

const.**DISPLAY_TYPE = 'PYGAME'**
Default display type: can be PYGAME or SIMPLECV

const.**IMAGE_SHAPE = (640, 480)**
Default image size retrieved from kinect

const.**PYGAME_LAYOUT = 4**
number of images to show in the output can be 2 or 4

const.**SHOW_FPS = True**
shows fps

const.**ENABLE_PROFILING = False**
get profiling info for the first 100 frames

2.2.7 Utility

utils.**boxes_intersect**(*bbox1*, *bbox2*)
Return if two rect overlap

utils.**boxes_intersect2**(*bbox1*, *bbox2*)
Return if two rect overlap

utils.**draw_bounding_box**(*image*, *bbox*)
Draw all bounding box inside image as red rectangle

Parameters

- **image** – image where draw the bounding box
- **bbox** – array of bounding boxes as (x,y,w,h) where x,y is the topleft corner of the rectangle

Returns image with bbox drawn

utils.**get_center_area_from_rect**(*rect*)
coordinates rect center

utils.**norm_correlate**(*a*, *v*)

utils.**rect_similarity**(*rect1*, *rect2*)
Check whatever two rect are similar with a tolerance of 10px in center distance and 0.1 in area ratio

utils.**rect_similarity2**(*r1*, *r2*)
Return if r1 and r2 satisfy overlapping criterion

utils.**rgb2gray**(*rgb*)

utils.**similarity_measure_rect**(*bbox_test*, *bbox_target*)
Return similarity measure between two bounding box

Parameters

- **bbox_test** –
- **bbox_target** –

Returns

utils.**to_rgb**(*im*)

2.2.8 Display

class video_type.**VideoDisplay**(*v_type*, *n_views*)
Video Display class. Depending on the method choosed (PYGAME or SIMPLECV) Initialize a screen type to show the output.

quit()
Quit the video stream

show(*frame_upper_left*, *frame_upper_right*, *frame_bottom_left=None*, *frame_bottom_right=None*)
Display the four frames in a 1280x960 display

Parameters

- **frame_upper_left** –
- **frame_upper_right** –
- **frame_bottom_left** –

- **frame_bottom_right** –

Returns True if the drawing succeed or False if the user choose to exit

Return type boolean

Raises SystemExit

2.3 Usage

To use our application you can simply launch the main application via:

```
python left-luggage-detection.py
```

Note: make sure your kinect is connected to your pc **and** the power supply otherwise you'll only be able to control the motor and not the video stream

2.3.1 Offline Usage

You can also test this application using registered video via Fakenect library which is included inside Openkinect.

To record a video:

```
mkdir directory_record
record directory_record
```

To use a recorded video you need to specify two environment variables `LD_PRELOAD` to point to the fakenet lib instead of the freenect one and `FAKENECT_PATH` that point to the video folder:

```
LD_PRELOAD="/usr/local/lib64/fakenect/libfreenect.so" FAKENECT_PATH="video/path" python left-luggage-
```

2.4 Project Dependencies

You need to have the following libs/programs installed:

2.4.1 1. OpenKinect

Home page: http://openkinect.org/wiki/Main_Page

Source Code: https://github.com/OpenKinect/libfreenect

Dependencies

Manual Build on Linux: for compiling, you need to have the following libs/programs installed:

- cmake
- libusb-1.0-0
- libusb-1.0-0-dev
- pkg-config

- libglut3
- libglut3-dev

APT users: (Works on Ubuntu 10.10)

```
sudo apt-get install cmake libglut3-dev pkg-config build-essential libxmu-dev libxi-dev libusb-1.0-0-
```

For Ubuntu 13.04, use this instead (replaced libglut3 with freeglut3):

```
sudo apt-get install cmake freeglut3-dev pkg-config build-essential libxmu-dev libxi-dev libusb-1.0-0
```

The python wrapper also need:

```
sudo apt-get install cython python-dev python-numpy
```

Manual Build

Download last libfreenect version from github and compile with CMAKE:

```
git clone git://github.com/OpenKinect/libfreenect.git
cd libfreenect
mkdir build
cd build
cmake ..
make
sudo make install
sudo ldconfig /usr/local/lib64/
```

To test if the library is correctly installed use:

```
sudo glview
```

To install the Python wrapper

```
cd libfreenect/wrappers/python
sudo python setup.py install
```

To use Kinect as a non-root user do the following:

```
sudo adduser $USER video
```

2.4.2 2. OpenCV

To install OpenCV you can use the following script

```
wget https://raw.github.com/jayrambhia/Install-OpenCV/master/Ubuntu/opencv_latest.sh
chmod +x opencv_latest.sh
./opencv_latest.sh
```

Note: If you want cuda support add WITH_CUDA=ON in the cmake section in the above script

2.4.3 3. Pygame

To display the video stream we use pygame so you'll need:

```
sudo apt-get install python-pygame
```

2.4.4 4. Optional

SimpleCV

If you decide to use SimpleCV class to display the video stream install SimpleCV from: http://simplecv.org/download

cProfile

To run memory and speed benchmark of the application

```
sudo apt-get install pythontracer
```

2.5 Dataset

Video4 https://mega.co.nz/#!2dwj0SLD!P5qPJTFT5II2egI3-5ZLgXhQYH-OQIIEZL0IkdtAefY

Video5 https://mega.co.nz/#!bI4h0T7Q!Q8BzVH_WK8rhRqHFlFuyPiUOyz2shkwf6Tp-UWdnEcU

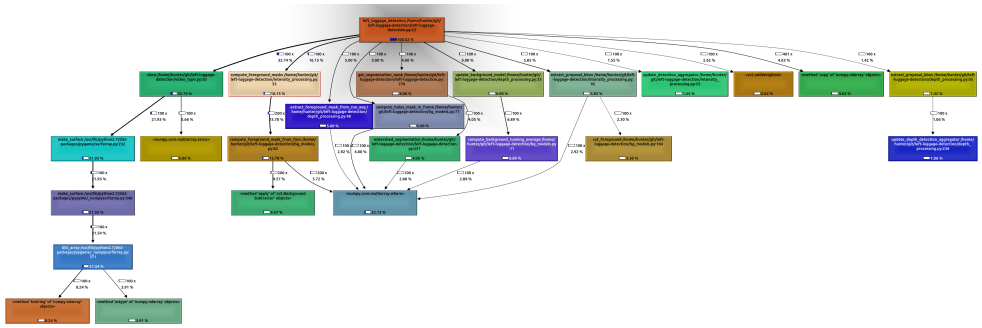
Video12 https://mega.co.nz/#!mBpDQYbJ!NBMqY2sgXgtC-9GkTEtdVwaLegos_hHYhS_by2-vMcg

Video13 https://mega.co.nz/#!jUIyAJxb!CN3aYrluSl4AnP18BPYN6jrVYgPKd6KOuhXpYf88UGA

Video14 https://mega.co.nz/#!iF5kwRYT!8IDAO76Yx4Ytl9Fj0fxGPtF3YIMkY2Jd9IA9x71FnGA

Video16 https://mega.co.nz/#!qUJVSJbY!9m3DMLIKzqRZa1MokpQlWpmhnISSqnhILXeuHlkgvEQ

2.6 Performance



The approach used in this application is demanding in term of performance. The proposed framework runs at 5 fps on a modern PC, just like the one proposed in the paper from which the authors were inspired.

2.7 GNU GENERAL PUBLIC LICENSE

Version 2, June 1991

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software–to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Lesser General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

GNU GENERAL PUBLIC LICENSE TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

> a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.
>
> b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.
>
> c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

> a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
>
> b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
>
> c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED

IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

> <one line to give the program's name and a brief idea of what it does.> Copyright (C) <year> <name of author>
>
> This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.
>
> This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.
>
> You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

Gnomovision version 69, Copyright (C) year name of author Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type ‘show w’. This is free software, and you are welcome to redistribute it under certain conditions; type ‘show c’ for details.

The hypothetical commands ‘show w’ and ‘show c’ should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than ‘show w’ and ‘show c’; they could even be mouse-clicks or menu items–whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the program ‘Gnomovision’ (which makes passes at compilers) written by James Hacker.

> <signature of Ty Coon>, 1 April 1989 Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License.

2.8 Indices and tables

- genindex
- modindex
- search

b

c

d

i

k

u

v

Q

R

S

T

U

V

X