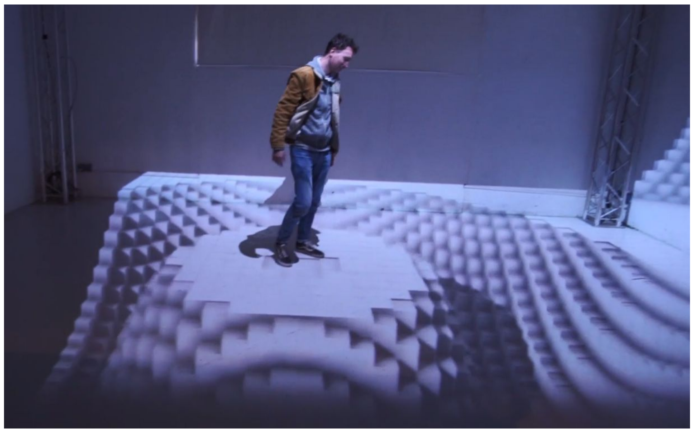
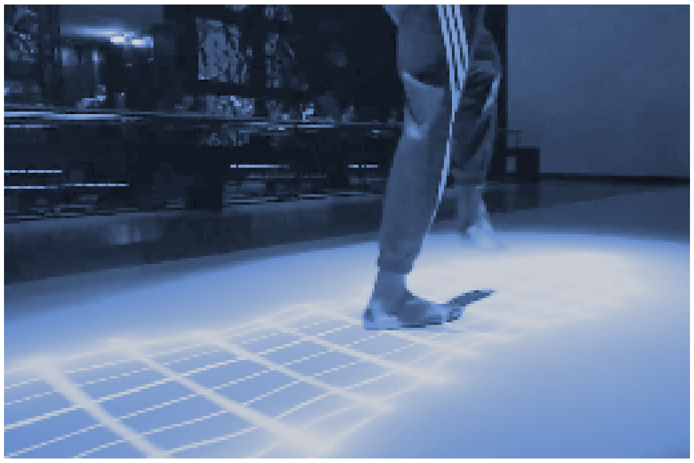
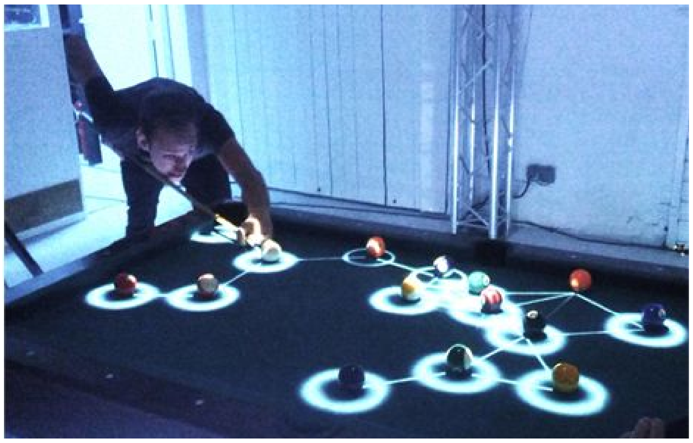
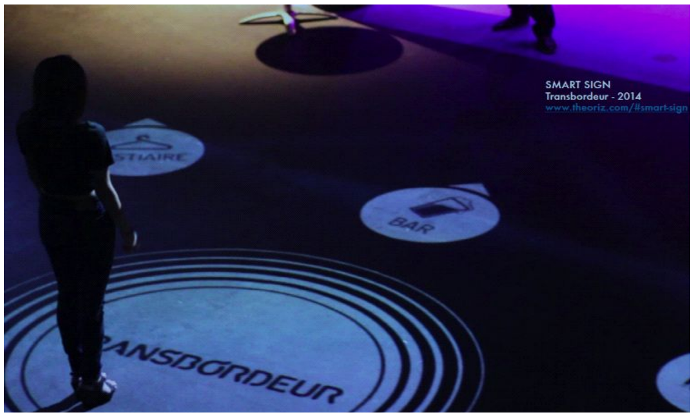
AUGMENTA

A tracking technology for creative people

User Guide

Last update : 08/10/2018

THÉORIZ

Table of Contents

1. DESCRIPTION

1.1. Presentation

Augmenta is a non-invasive distributed tracking technology empowering creators. It allows to track people and objects without them wearing any sensors, in areas of arbitrary size. The tracking data is sent through an open protocol compatible with all video, audio, or light creation softwares.

The Augmenta technology offers a wide range of applications for various locations and settings such as:

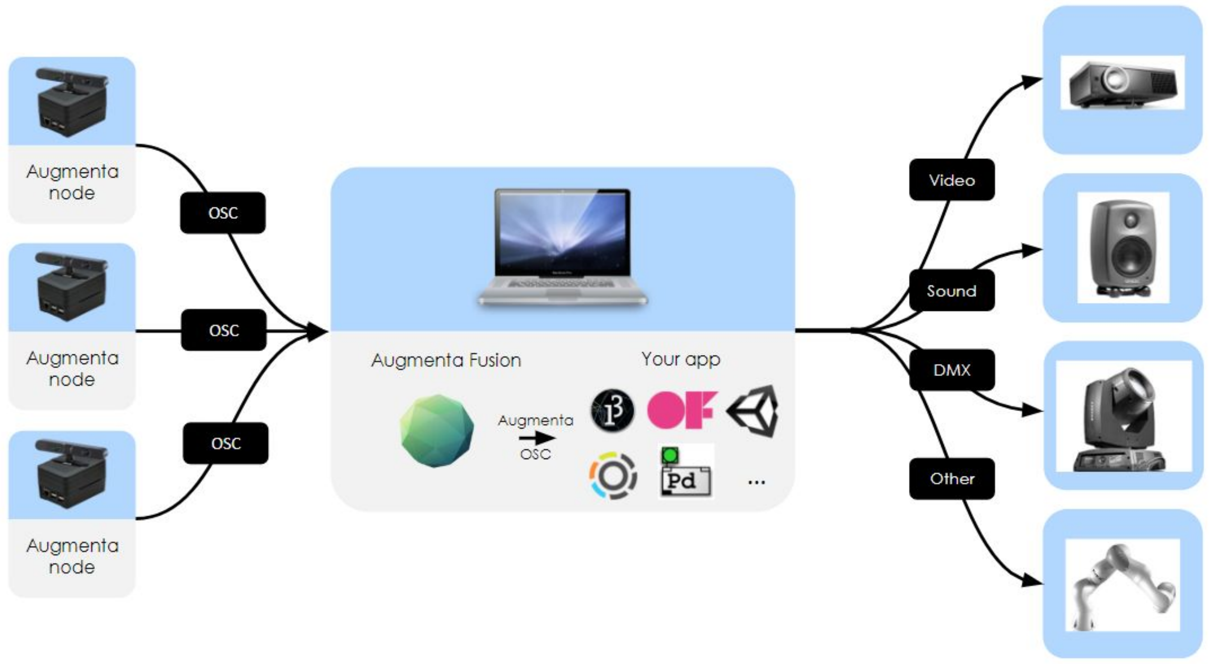
- Theatres and stages
- Events, shows and exhibitions
- Art installations and museums
- Entertainment and amusement parks

Augmenta is developed by THÉORIZ, a creative studio specialized in tracking technologies and designing beautiful and unconventional experiences with cutting-edge technology.

1.2. System Overview

Augmenta is a combination of hardware (multiple sensors, dedicated computers and a server computer) and software (Augmenta, AugmentaFusion).

The data flow diagram of Augmenta is shown below.



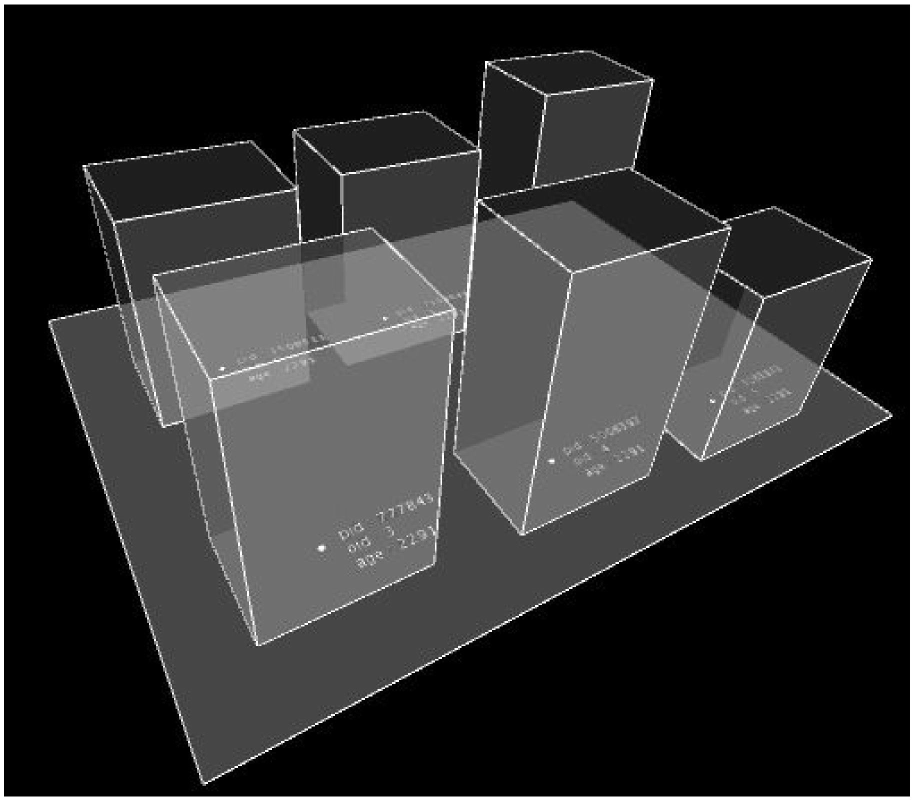
Augmenta data flow overview

The **Augmenta nodes** are a combination of a sensor (3D camera, LiDAR) and a computer that runs Augmenta to process the sensor data and format it to the Augmenta protocol. The output of a node is an OSC stream of the Augmenta data computed by the node.

The nodes stream are sent to a server where the software **AugmentaFusion** is used to calibrates the nodes and fuse their data together. The output of AugmentaFusion is also an OSC stream of Augmenta data. This OSC data following the Augmenta protocol can then be used in any application capable of receiving OSC. These applications can then be used to control any output (video, sound, light, etc…) according to the Augmenta data.

A working Augmenta setup contains at least one node that sends an Augmenta data stream. This stream can be used directly in your application, or go through AugmentaFusion first. Using AugmentaFusion allows to calibrate the node and merge different node Augmenta data, however you can use the stream from the nodes directly if you use another calibration and merging method.

The Augmenta data contains information about the whole scene such as its size and the number of object tracked. It also contains information about each detected object such as its bounding box size, its position and its velocity.


Visualization of Augmenta data

1.2.1. IP Address

The augmenta nodes IP address can be given automatically through DHCP or be made static if the address of each node need to remain the same (during an exploitation for example).

1.3. Hardware

All the Augmenta nodes are composed of a sensor (camera or LiDAR) and a computer. The computer are usually Intel Nuc mini-computers. In the sections below, most pictures are only showing the sensor since the computer can be the same for all sensors.

1.3.1. Computer

Material	Black aluminium (dustproof design)
Noise	0db (Fanless design)
Work temp	-10° / +50°C
Work Humidity	0% / 95% non-condensing
Dimension	155 * 126.5 * 52.5mm
Hanging	VESA bracket or light hook
Remote boot	Electricity and WOL enabled
Remote access	Web interface
CPU / RAM / Disk life expectancy	~10 years (industrial grade)

1.3.2 Sensors

1.3.2.1 Astra

Camera FOV	60° horiz. x 49.5° vert.
Camera depth resolution	640x480
Max detection distance	~6m
Number of person tracked	infinite
Approx. weight	1.3 kg
Throw Ratio	0.87:1
Area covered at 5m	~ 5.8m x 4.6m[1]
Data frequency	30Hz
Wavelength	Infrared (~827-850nm)

The **Orbbec Astra** camera is a 3D camera using infrared to provide a depth map. It can be used to detect people or object at up to 6 meters. Note that strong infrared beams reflected in the detection space may interfere with the Astra camera.

1.3.2.2 Xtion

[1] https://www.desmos.com/calculator/qiirwypgvi

Camera FOV	58° horiz. x 45° vert.
Camera depth resolution	640x480
Max detection distance	~6m
Number of person tracked	infinite
Approx. weight	1.3 kg
Throw Ratio	0.9:1
Area covered at 5m	~5.5m x 4.1m
Data frequency	30Hz
Wavelength	Infrared (~827-850nm)

The **Asus Xtion** is a 3D camera detecting up to 6m of depth. Note that strong infrared beams reflected in the detection space may interfere with the Xtion camera.

1.3.2.3 Kinect

Camera FOV	57° horiz. x 43° vert
Camera depth resolution	320 x 240
Max detection distance	~5m
Number of person tracked	infinite
Approx. weight	1.3 kg
Throw Ratio	0.92:1
Area covered at 5m	~5.4m x 3.9m
Data frequency	30Hz
Wavelength	Infrared (~827-850nm)

The **Microsoft Kinect** is a 3D camera detecting up to 5m of depth. Note that strong infrared beams reflected in the detection space may interfere with the Kinect camera.

1.3.2.4 Kinect2 (Deprecated)

Camera FOV	70° horiz. x 60 vert.
Depth camera resolution	512 x 424
Max detection distance	~7m
Number of person tracked	infinite
Approx. weight	1.5 kg
Throw Ratio	0.72:1
Area covered at 5m	~7.0m x 5.8m
Data frequency	30Hz
Wavelength	Infrared (~827-850nm)

The **Microsoft Kinect2** is a 3D camera detecting up to 7m of depth. Note that strong infrared beams reflected in the detection space may interfere with the Kinect2 camera. Be aware that the Kinect2 has less data on the corners of its field of vision and therefore should have more overlap to compensate.

1.3.2.5 LiDAR10

Detection FOV	270°
Maximum angular resolution	0.125°
Max detection distance	10m
Number of person tracked	infinite
Approx. weight	1 kg
Data frequency	40Hz
Wavelength	Infrared

The LiDAR10 node uses an **Hokuyo UST-10LX-H01** LiDAR sensor. It scans a plane in front of it with a 270° field of view. It is useful to detect objects in front of a surface with high precision, like touches on a wall or a table for example.

1.3.2.6 LiDAR20

Detection FOV	270°
Maximum angular resolution	0.25°
Max detection distance	20m
Number of person tracked	infinite
Approx. weight	1 kg
Data frequency	40Hz
Wavelength	Infrared

The LiDAR20 node uses an **Hokuyo UST-20LX** LiDAR sensor. It scans a plane in front of it with a 270° field of view. It is useful to detect objects in front of a surface with high precision, like touches on a wall for example.

1.4. Software

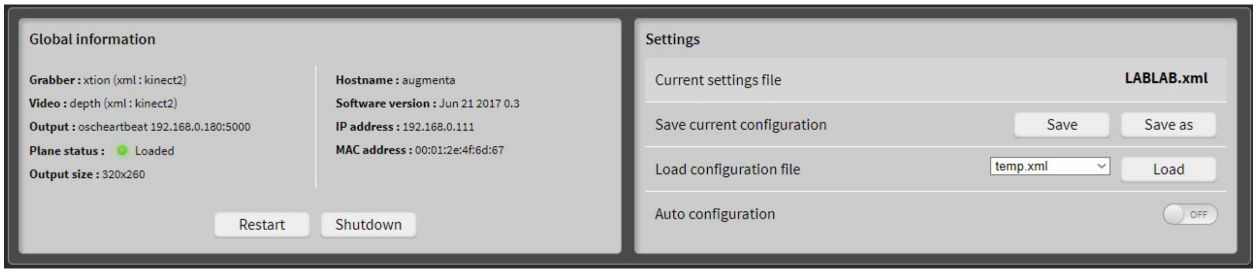
1.4.1. Augmenta

1.4.1.1. Presentation

Each Augmenta node is running the Augmenta software whose role is to grab the data stream from the connected sensor and apply the corresponding image processing chain to extract Augmenta points from the sensor data. Those Augmenta points are then sent through OSC via the Augmenta protocol.

1.4.1.2. Interface

The Augmenta software is accessible through a web Interface for each Augmenta node. This interface can be accessed through the ip address of the node. It contains four top panels and a bottom panel. Those panels are the following:



Global information

Grabber : xtion (xml : kinect2)
Video : depth (xml : kinect2)
Output : oscheartbeat 192.168.0.180:5000
Plane status : ● Loaded
Output size : 320x260

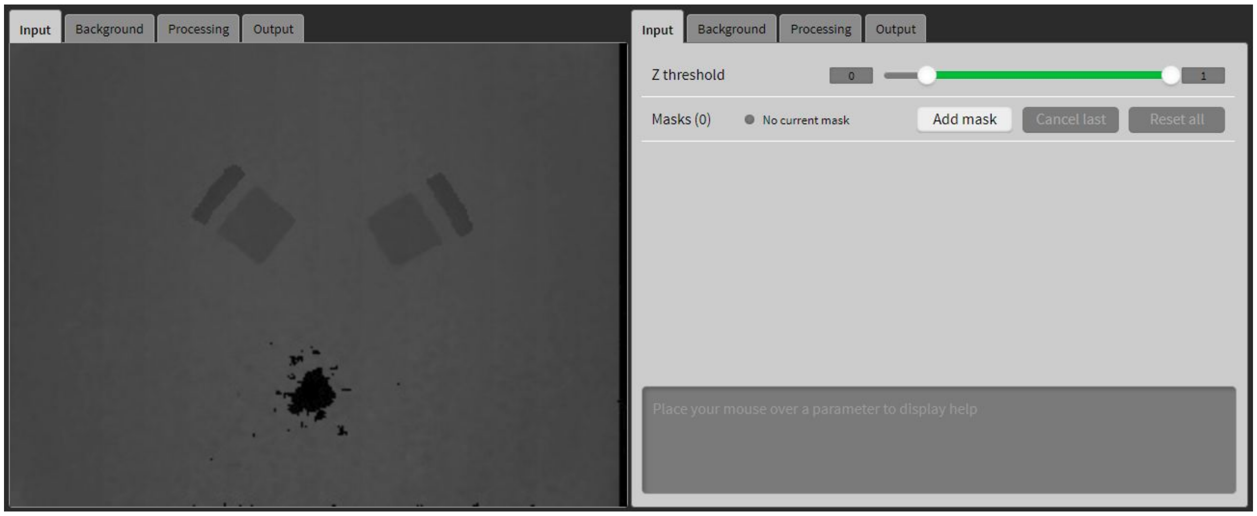
Hostname : augmenta
Software version : Jun 21 2017 0.3
IP address : 192.168.0.111
MAC address : 00:01:2e:4f:6d:67

Settings

Current settings file	**LABLAB.xml**
Save current configuration	
Load configuration file	temp.xml
Auto configuration	OFF

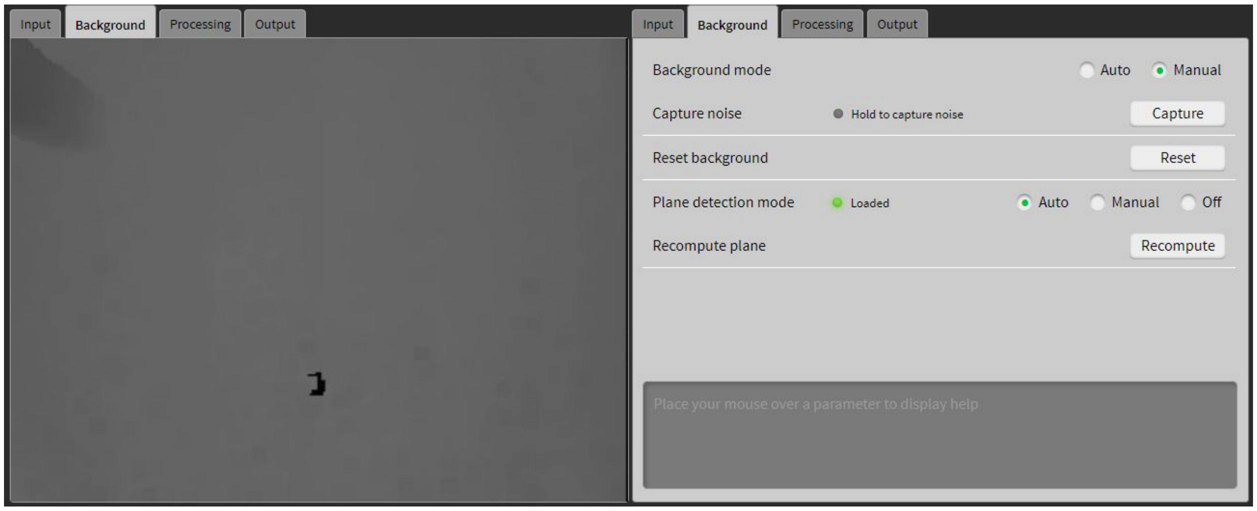
The **Global information & Settings** panel corresponds to the node information and configuration:

- On the left global information of the node are displayed, along with buttons to restart or shutdown the node.
- On the right the settings of the node, allowing to save the current configuration or load a new one. The Auto Configuration mode automatically choose the settings, disabling all manual settings.



The **Input** panel corresponds to the input stream from the camera and contains:

- On the left a preview of the camera stream (here the depth from a Xtion camera looking at two chairs from above).
- On the right, parameters to define the detection area are available, with their descriptions written in the grey box below. The **Z threshold** parameter allows to limit the depth range where objects are detected. The **Masks** allows to define areas where nothing will be detected.



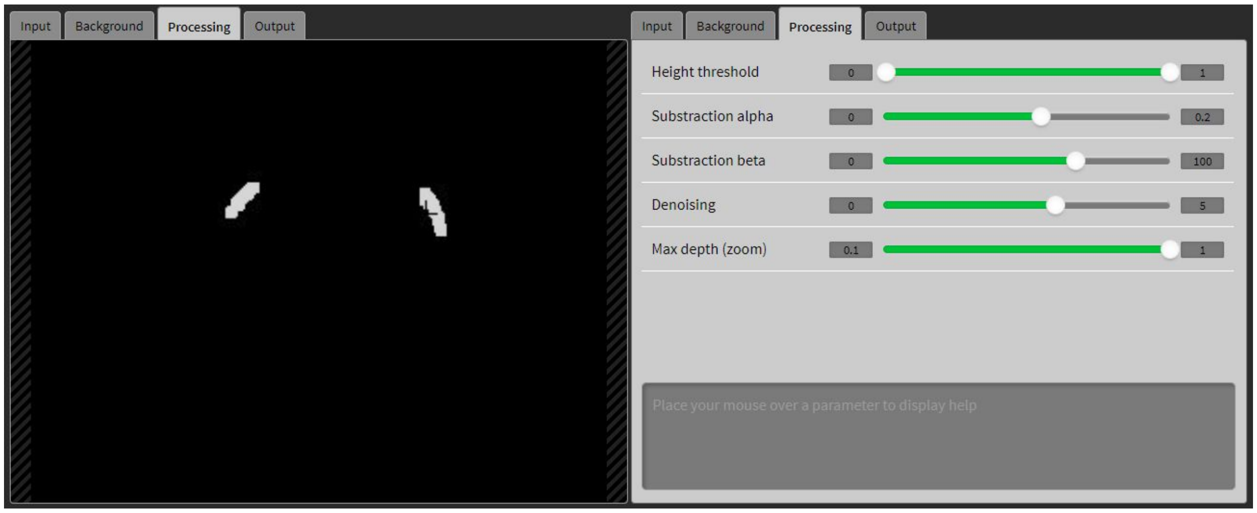
The **Background** panel corresponds to the background detection and contains:

- On the left a preview of the computed background.

- On the right, parameters for the background detection. The background can be detected automatically or manually. **Capture noise** allows to choose the amount of noise to remove in manual mode. The background plane can be detected automatically, manually, or not at all. **Recompute plane** is useful to update the background plane after the camera is moved.

When the **Background mode** is set to **Auto**, the node will continuously update its background with the farthest distance seen at each pixel. This allows to have occluders moving in front of the background and still get an accurate background after each pixel as seen the real background at least once. This mode is useful when people or moving objects might be occluding the background during calibration. However, the background mode should be set to **Manual** when objects may be seen behind the desired background. For example if the background has a door that can be opened, the background would be updated automatically with what is seen behind the door as soon as the door is opened in Auto mode.

When the **Plane detection mode** is set to **Auto**, a RANSAC algorithm is used to automatically compute the background plane equation. This should be used when a flat plane is visible (wall or floor for example). In **Manual**, the plane detection is done by manually choosing points in the preview that should belong to the background plane. This should be used when no flat plane is easily visible (complex scenery in the background for example). When set to **Off**, no plane is computed. This should be used when the background is not visible by the camera (too far away).

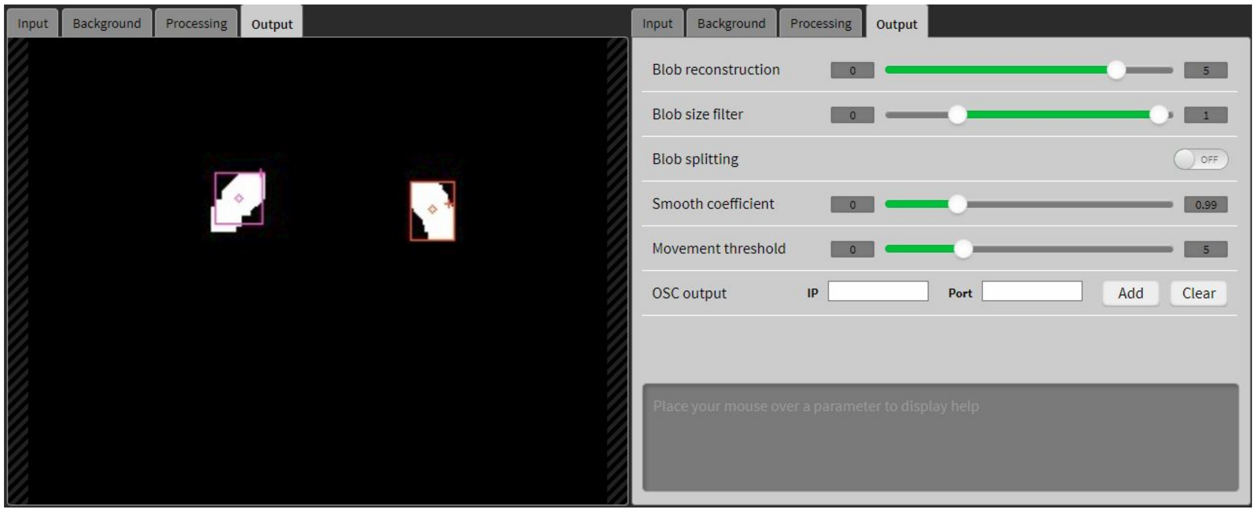


Height threshold 0 1
Substraction alpha 0 0.2
Substraction beta 0 100
Denoising 0 5
Max depth (zoom) 0.1 1

Place your mouse over a parameter to display help

The **Processing** panel corresponds to the object detection and contains:

- On the left a preview of the detected objects.
- On the right, parameters to refine the points detection. The **Height threshold** defines the height where the objects will be detected, parallel to the background plane. The **Subtraction** parameters allows to refine the object detection, here they are used to keep only the back of the chairs. The **Denoising** parameter allows for additional denoising at this step. The **Max depth** parameter allows to define the maximum depth of the objects.



Blob reconstruction 0 5
Blob size filter 0 1
Blob splitting OFF
Smooth coefficient 0 0.99
Movement threshold 0 5
OSC output IP Port
Place your mouse over a parameter to display help

The **Output** panel corresponds to the blobs (or persons) detection from the detected objects and contains:

- On the left a preview of the detected blobs.
- On the right, parameters to refine the blob detection. The **Blob reconstruction** parameter defines how much detected objects are enlarged in order to better create a blob. The **Blob size filter** defines the minimum and maximum sizes of the desired blobs. When **Blob splitting** is activated, blobs bigger than the maximum size will be splitted into smaller blobs instead of removed. The **Smooth coefficient** will smooth the movements of the blobs for smoother velocity, at the cost of increased delay. The **Movement threshold** defines how much a blob has to actually move in order to be moved in the detection. **OSC output** allows to define a new OSC output IP address and port.

Note that **when using AugmentaFusion**, the OSC connections with Augmenta nodes are done automatically. This means that **no OSC output should be defined** on the nodes as AugmentaFusion will set them up automatically.

1.4.2. AugmentaFusion

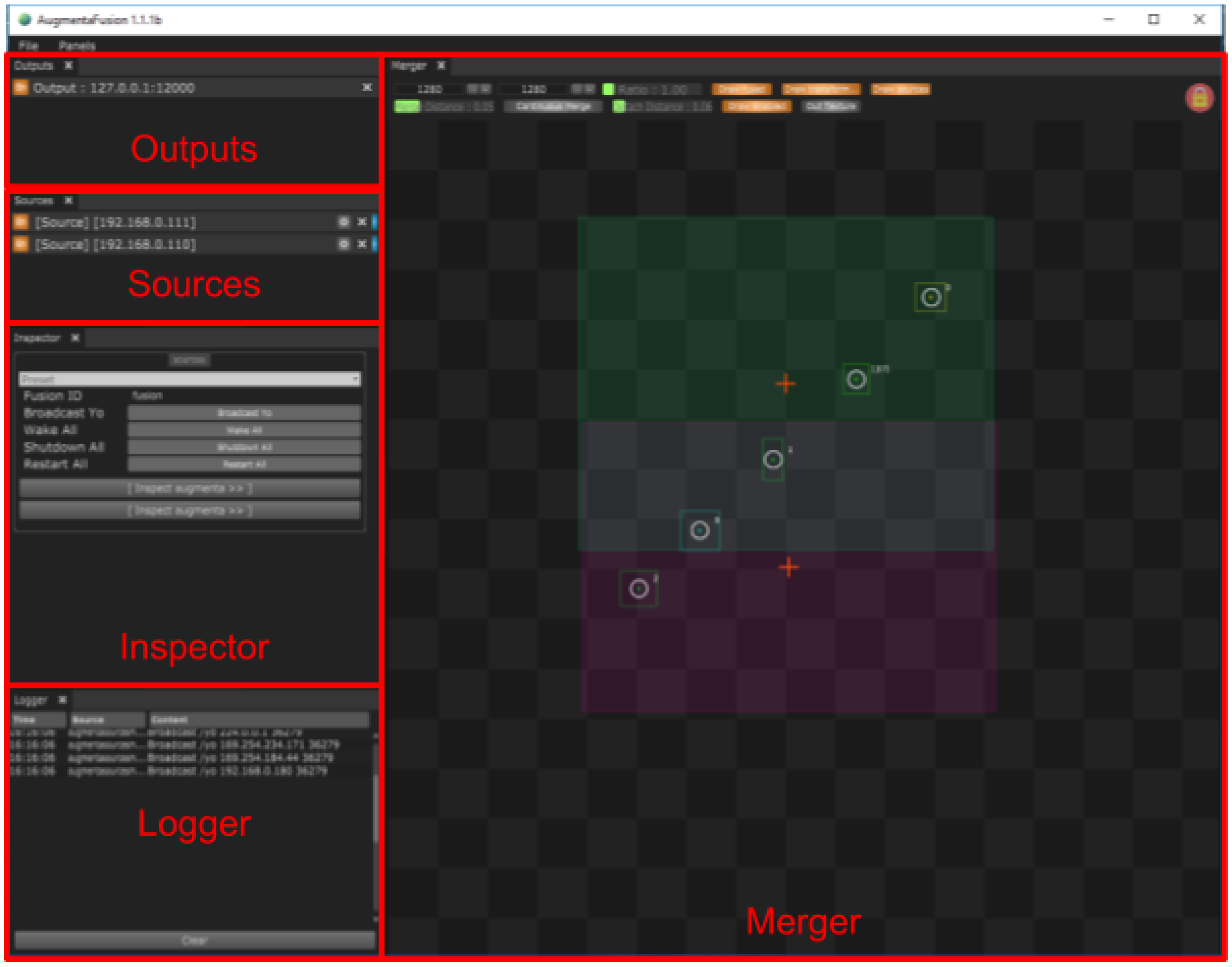
1.4.2.1. Presentation

AugmentaFusion is a software taking one or several Augmenta stream as input (from Augmenta nodes) and displaying them all together with their respective resolution on an arbitrary output resolution. Its two main roles are: 1. to calibrate the nodes relatively to each other by translating and rotating their output, and 2. to merge the overlapping blobs from different nodes.

The output of AugmentaFusion is also an Augmenta OSC data stream, using the same Augmenta protocol. This makes it possible to use several instances of AugmentaFusion in the Augmenta chain if needed. It also means that Augmenta compatible application can receive and use data from an Augmenta node or AugmentaFusion in the exact same way.

1.4.2.2. Interface

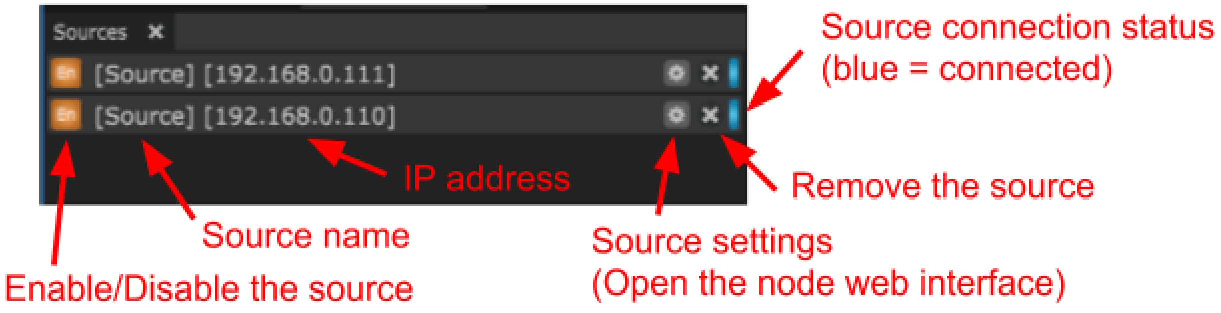
The interface of AugmentaFusion contains five panels described below. Note that hovering the cursor over any part of AugmentaFusion UI will display a tooltip with information about the hovered element.



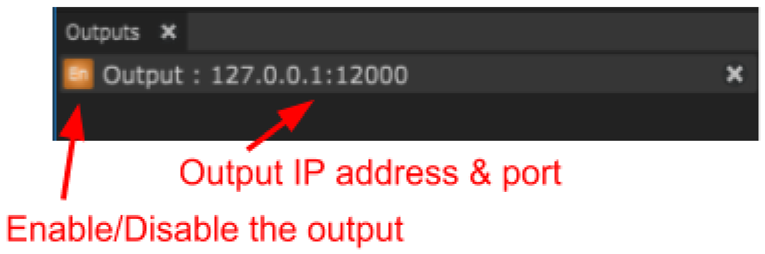
```
16:16:06  augmentasourcem...Broadcast /yo 224.0.0.1 36279
16:16:06  augmentasourcem...Broadcast /yo 169.254.234.171 36279
16:16:06  augmentasourcem...Broadcast /yo 169.254.184.44 36279
16:16:06  augmentasourcem...Broadcast /yo 192.168.0.180 36279
```

AugmentaFusion interface

The **Sources** panel contains the names and IP addresses of every incoming Augmenta OSC stream. Right clicking on the empty area of the panel allows to add a new source.



The **Outputs** panel contains the output IP address and ports for the outgoing Augmenta OSC stream.



The **Inspector** panel contains the parameters and options for the selected panel or object.

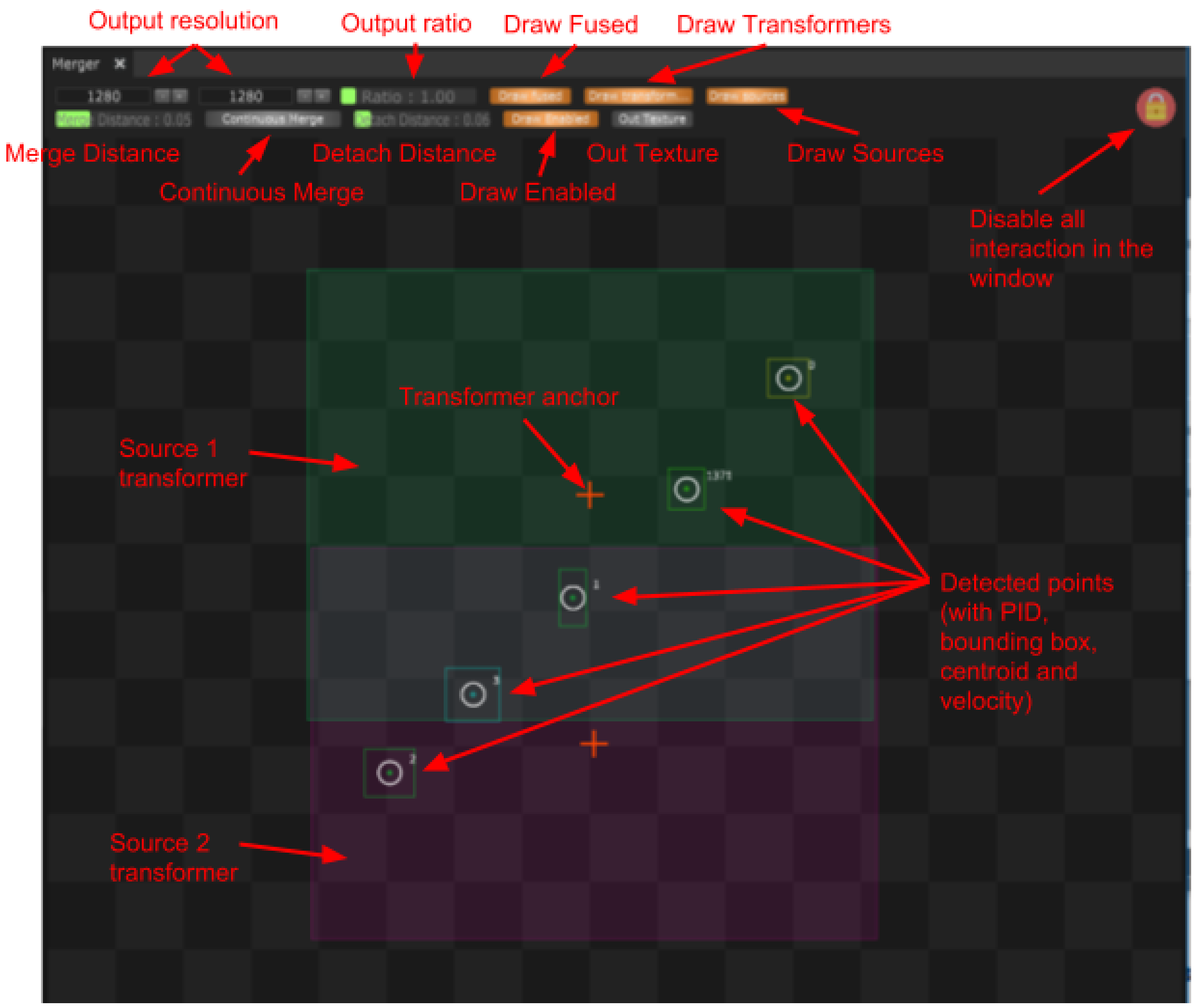
The **Logger** panel contains messages from the application.

The **Merger** panel contains the preview of the output resolution with each input source and their merged blobs. The following interaction are available in the Merger window:

- **Holding Alt + left click** inside a transformer will move this transformer anchor.
- **Holding left click** on a transformer while translate it.
- **Holding Maj + left click** inside a transformer will rotate and scale it around its anchor.

Draw Enabled enable the drawings in the Merger window and should be disabled to save resources once everything is calibrated and AugmentaFusion is running in the background.

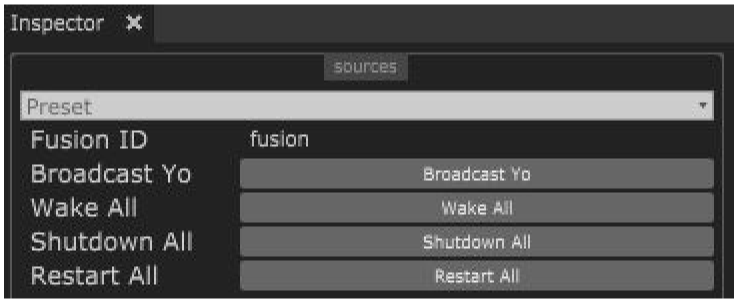
Out Texture enable a spout output called AugmentaFusion of the Merger preview window and can be used to calibrate Augmenta with the projection area.



1.4.2.3. Nodes Control

Global Controls

Global controls for all sources appear in the inspector panel when clicking on an empty area in the Sources panel.



```
Inspector
sources
Preset
Fusion ID        fusion
Broadcast Yo     [Broadcast Yo]
Wake All         [Wake All]
Shutdown All     [Shutdown All]
Restart All      [Restart All]
```

Broadcast Yo: Broadcast Yo to the whole network. Augmenta nodes will respond to this broadcast with their settings and AugmentaFusion will then establish an OSC connection will every node that answered.

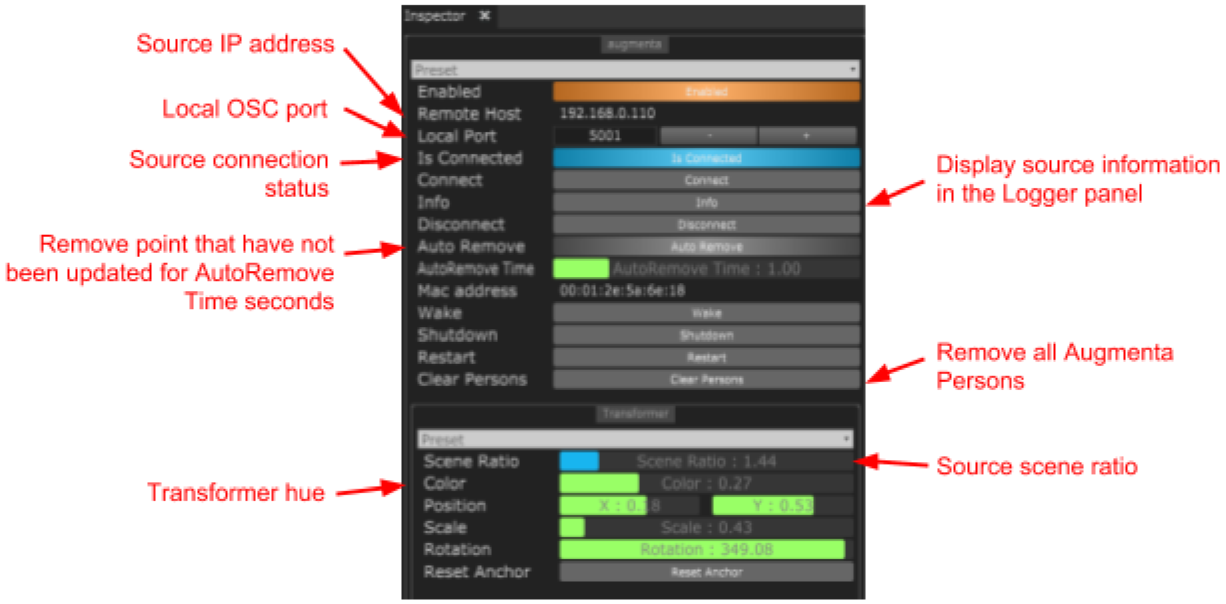
Wake All: Wake all sources **and** all sources in AugmentaFusion cache. Every time a source is connected to AugmentaFusion, its mac address is added to AugmentaFusion cache.

Shutdown All: Shutdown all sources.

Restart All: Reboot all sources.

<u>Source Controls</u>

The source controls can be seen in the Inspector panel when clicking on a source.


Source IP address
Local OSC port
Source connection status
Remove point that have not been updated for AutoRemove Time seconds
Transformer hue
Display source information in the Logger panel
Remove all Augmenta Persons
Source scene ratio

```
Inspector
augmenta
Preset
Enabled          [Enabled]
Remote Host      192.168.0.110
Local Port       5001  [-] [+]
Is Connected     [Is Connected]
Connect          [Connect]
Info             [Info]
Disconnect       [Disconnect]
Auto Remove      [Auto Remove]
AutoRemove Time  AutoRemove Time : 1.00
Mac address      00:01:2e:5a:6e:18
Wake             [Wake]
Shutdown         [Shutdown]
Restart          [Restart]
Clear Persons    [Clear Persons]
Transformer
Preset
Scene Ratio      Scene Ratio : 1.44
Color            Color : 0.27
Position         X : 0.18   Y : 0.53
Scale            Scale : 0.43
Rotation         Rotation : 349.08
Reset Anchor     [Reset Anchor]
```

The same **Wake**, **Shutdown** and **Restart** controls are available for each source to monitor sources independently.

Note that the nodes automatically wake up when plugged-in.

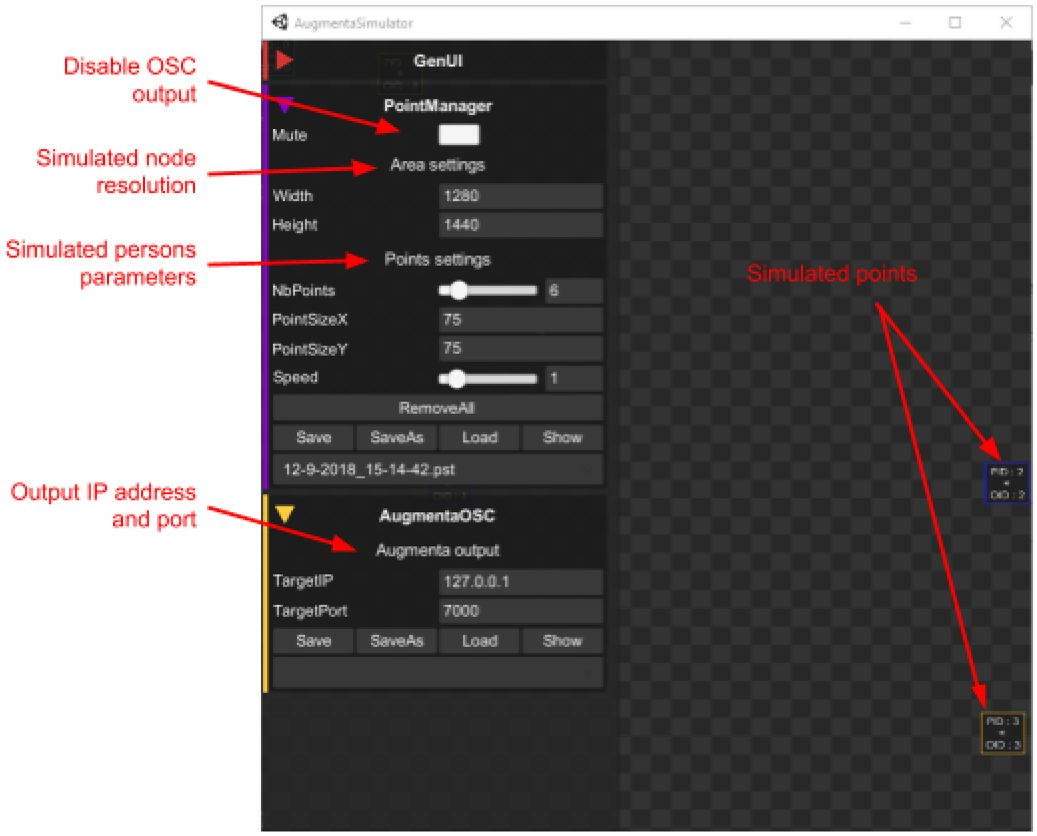
1.4.3. Augmenta Simulator

1.4.3.1. Presentation

The AugmentaSimulator is an application simulating an Augmenta node and sending fake Augmenta persons via OSC. It is useful to test applications using Augmenta without a functional Augmenta node or people/objects in the interaction space.

You can download it at https://github.com/Theoriz/Augmenta-Simulator.

1.4.3.2. Interface

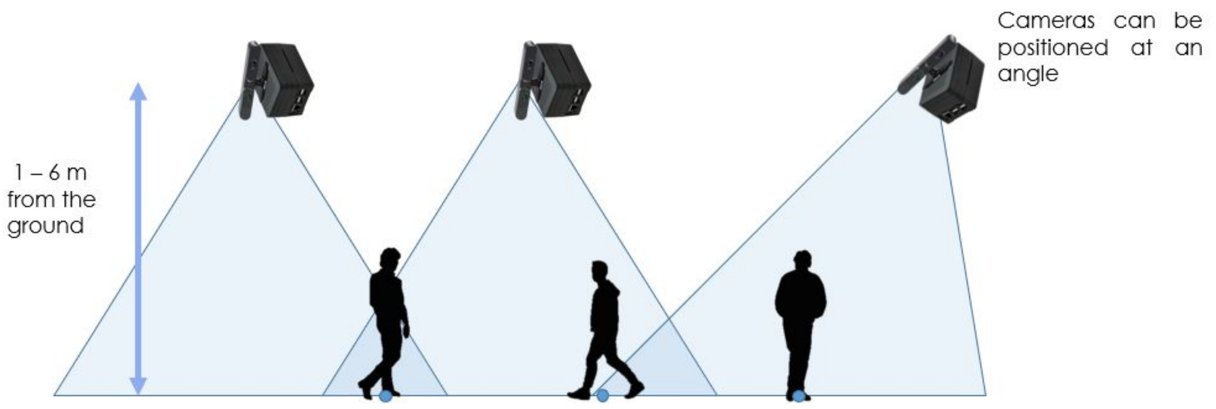


Mute
Width 1280
Height 1440
NbPoints 6
PointSizeX 75
PointSizeY 75
Speed 1
12-9-2018_15-14-42.pst
TargetIP 127.0.0.1
TargetPort 7000

2. SETUP GUIDELINES

2.1. Installation

The Augmenta nodes should be placed in order to effectively cover the entire desired area. Where the nodes should be placed in space depends on the nodes specifications. Specifications for the existing nodes are detailed in Section 4, Nodes Specifications. An example of setup using Astra nodes is given below.



Example of Augmenta setup with Astra nodes

In order to use several nodes with a server running AugmentaFusion, the nodes and the server should be on the same network in order to communicate via OSC.

Some **general installation guidelines** for camera nodes are given below :

- Cameras **height should not be below 1m or above 6m**.
- The **ideal angle for the cameras is 0°** (facing the ground directly). The camera angle should not go above 45°.
- **The main plane seen by the camera must be the ground**. If this is not the case, the reprojection of the data must be disabled in Augmenta or the data will make no sense.
- Ideally **the ground should be light and mat**. The darker and reflective the ground is, the closer the camera should be to the ground. A perfect reflective or transparent surface will not be seen by the cameras.
- If the ground cannot be seen, the **background can be learned using another temporary**

material on the ground surface (paper, sheet, cardboard, etc...). Once the background is computed, the background plane can be recomputed.

- If the ground cannot be seen by any mean, Augmenta will work without reprojection and the cameras angles should be 0°.
- When using Kinect or Xtion cameras, a good practice is too orient all cameras in the same direction in order to void some loss of information on slightly reflective surfaces.
- **Cameras overlap should be between 0.5 and 2 meters** depending on the application. A high overlap will improve the tracking but increase the number of cameras required for the same area.

2.2. Calibration

2.2.1. Augmenta Setup

Setting up Augmenta differs slightly depending on the desired use case (video, sound, light). For video projection, the setup is detailed below. For the other use cases, the calibration steps and prerequisites may vary slightly.

<u>**Prerequisites**</u>

Video projection: A video-mapped projection surface with a video software able to use a Spout or Syphon video input. You also need to know the resolution expected by the media server (or video projector) as this is the resolution you need to put in AugmentaFusion.

LED or TV screen: You need to know the resolution expected by the media server (or the screen) as this is the resolution you need to put in AugmentaFusion.

Others: Check the Use Case Section.

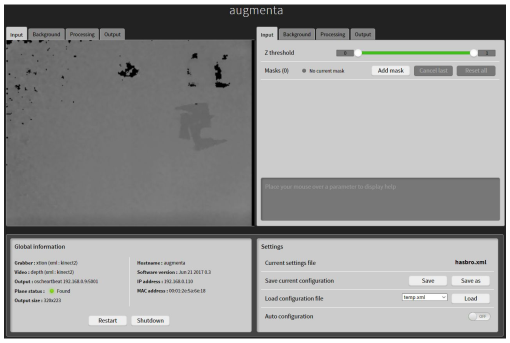
For a complete Augmenta setup, follow these steps:

1. **Setup your nodes** in the physical space, according to the node specifications and desired area coverage.
2. **Ensure that your nodes overlap** correctly to cover the whole area. This can be computed using the nodes specifications in Section 1.3, or visualized through each node interface

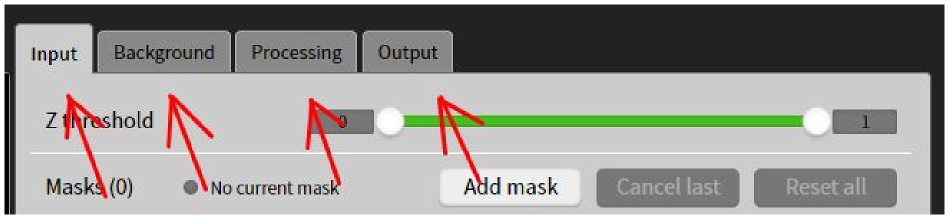
in the next step.

3. **Configure each node** with the following steps :

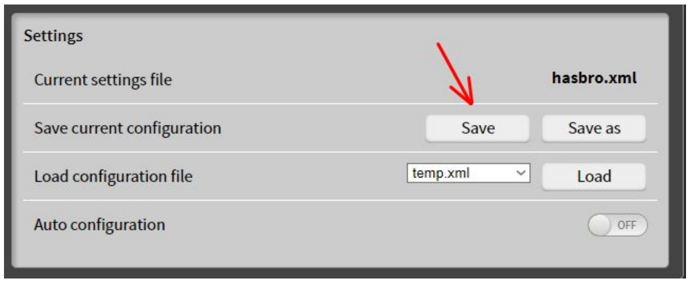
    a. **Connect to the node's** web interface with a laptop, tablet or smartphone.



    b. **Configure the Augmenta parameters** of the node in each panels successively (Input -> Background -> Processing -> Output) according to your scene.



    c. **Save the node configuration**. Note that Augmenta has an auto-save and will regularly save the last configuration.



Settings

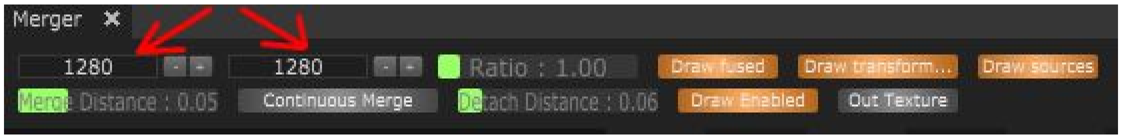
Current settings file	hasbro.xml
Save current configuration	Save / Save as
Load configuration file	temp.xml / Load
Auto configuration	OFF

4. **Open AugmentaFusion** on your server.

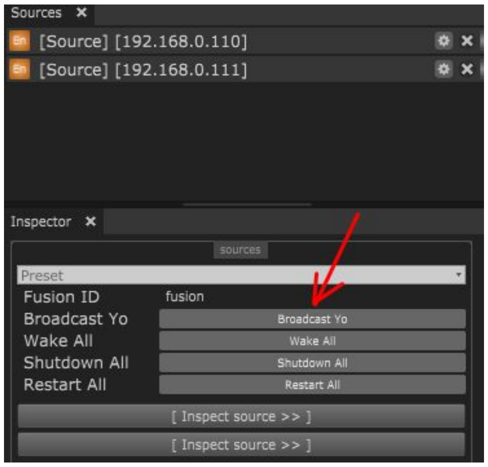
5. **Enter the whole area size** in meters.

6. **Enter your desired output resolution** in the Merger panel. The desired output resolution is the projection resolution expected by the display device. This is necessary for video calibration and/or if your display device is a video projector, a screen or a LED panel.

This is not needed if you do not have a display device (lights or sound).



7. **Add your nodes** in AugmentaFusion sources (either manually or using Broadcast Yo, cf Section 1.4.2).



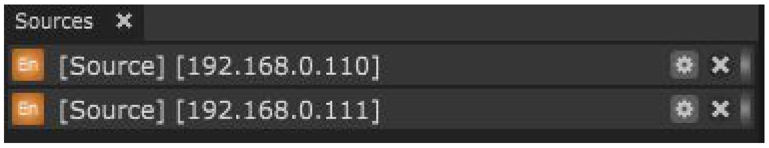
8. **Calibrate each node** in AugmentaFusion (cf Section 2.2.2.)

2.2.2. AugmentaFusion Calibration

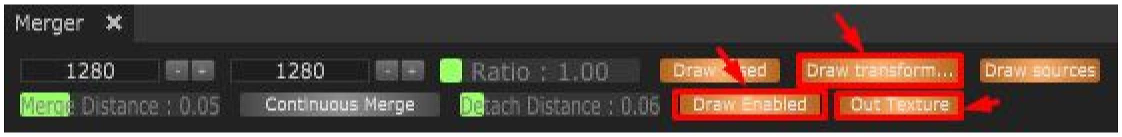
Add Screenshot

The calibration of each Augmenta node in AugmentaFusion is done with a two-point calibration. This is done by following these steps:

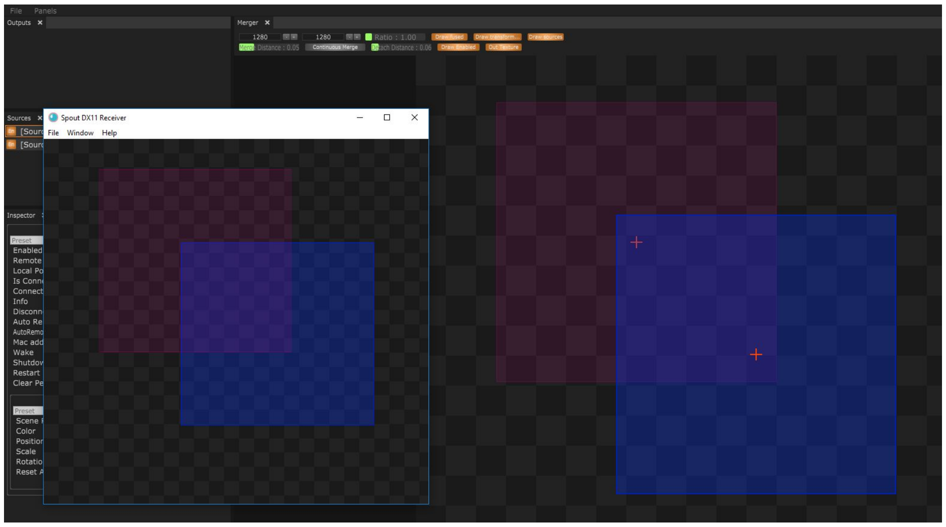
1. **Connect the Augmenta node(s)** to have it(them) as a source(s) in AugmentaFusion, enable the source(s) if needed.



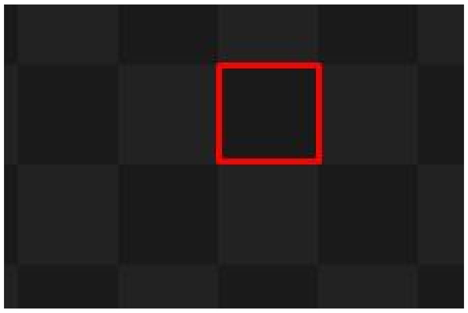
2. In the Merger panel, **enable Draw transformers, Draw Enabled and Out Textur**e in order to have a spout output with the Merger window.



3. **Display the spout output** called AugmentaFusion on your projection display in order to see the Merger window on your projection area.



4. **Check that the background squares are actual squares** on the projection surface. If this is not the case, either the output resolution in AugmentaFusion is not correct, or the video-mapping is not correct.



5. **Calibrate each Augmenta node** (or source in AugmentaFusion) with the following steps:

    a. **Put a single object in your interaction area**. You should see a single point in AugmentaFusion.

   b. **Translate the source transformer** by dragging and holding left click in the Merger window until the detected point match position of the object in the projection area.
   c. **Move the transformer anchor to the center of the detected object** by holding Alt and dragging with the left click.
   d. **Move the object to the opposite side of the interaction area**. The detected point should also move in AugmentaFusion.
   e. **Rotate and scale the transformer** around its anchor until the detected point match the new object position in the projection area.

6. **Disable Draw Enabled and Out Texture** in AugmentaFusion to improve its performances.
7. **Save** your AugmentaFusion configuration. You are done!

Note that since Spout is used by AugmentaFusion to calibrate with video projection, AugmentaFusion should be on the machine that generate the video projection for the video calibration. Once the calibration is done, Spout is no longer used and AugmentaFusion can then be deported on another machine. Once AugmentaFusion supports an NDI output, it will no longer need to be on the same machine for calibration.

2.3. Calibration without video

Add Screenshot

You do not need a video output to calibrate Augmenta. In that case, you can manually match position of the calibrations points in the area to their position in AugmentaFusion by placing flags in AugmentaFusion manually.

The calibration of an Augmenta node can then be done by following those steps:

1. **Get your whole area size in meters**. Enter it in AugmentaFusion.
2. **Put an object in your area and measure its coordinates** in meters from the top left of the

area.

3. **Put a flag in AugmentaFusion** at the coordinates of your object.
4. **Translate the transformer of the source** to match the detected point and the flag in AugmentaFusion. Place the anchor of the transformer at that point.
5. **Move the object** to another place in your area and measure its new coordinates.
6. **Put a flag in AugmentaFusion** at the new coordinates of your object.
7. **Rotate and scale the transformer** of the source to match the detected point and the new flag.

2.4. Monitoring Nodes

Using the source controls in AugmentaFusion, you can easily **wake (All)**, **shutdown (All)** or **restart (All)** each (all) node(s) on the network. **Wake All** will start all nodes on the network that have been connected to AugmentaFusion at least once.

To wake up a new node, create a new source in AugmentaFusion, enter the node mac address in the source properties, then Wake it.

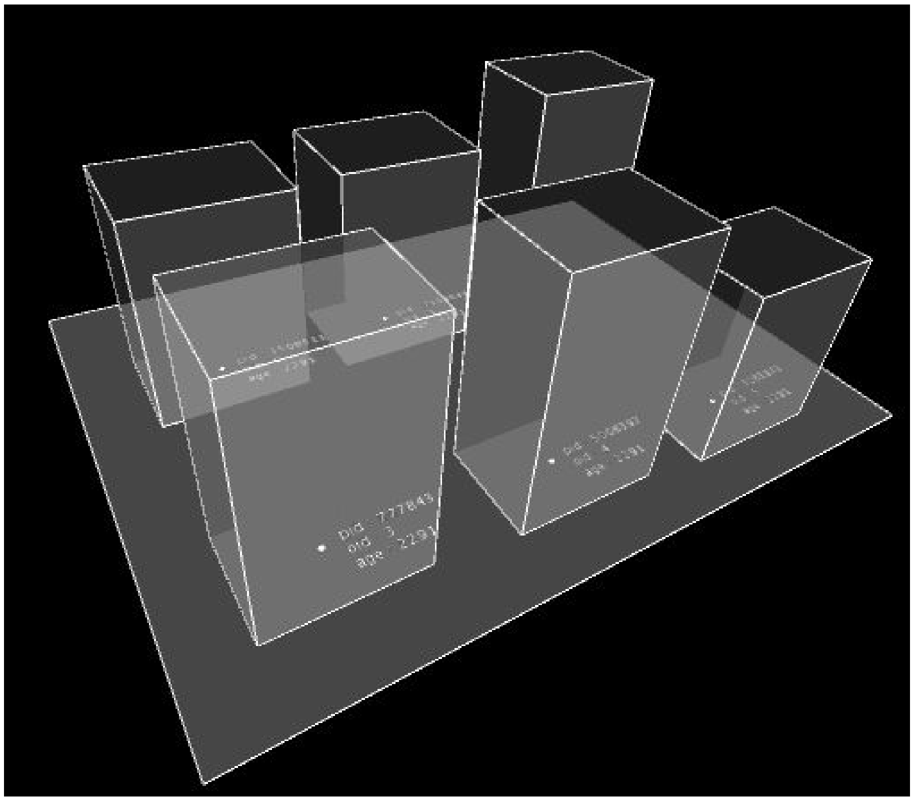
3. CONTENT CREATION

3.1. Augmenta Protocol

The Augmenta Protocol is an OSC protocol that is sent by the Augmenta nodes (or AugmentaFusion and AugmentaSimulator). It can be directly listened to and used by any software capable of receiving OSC messages. To facilitate this, some softwares and libraries have already been developed as described below.

The details of the Augmenta OSC Protocol are the following:

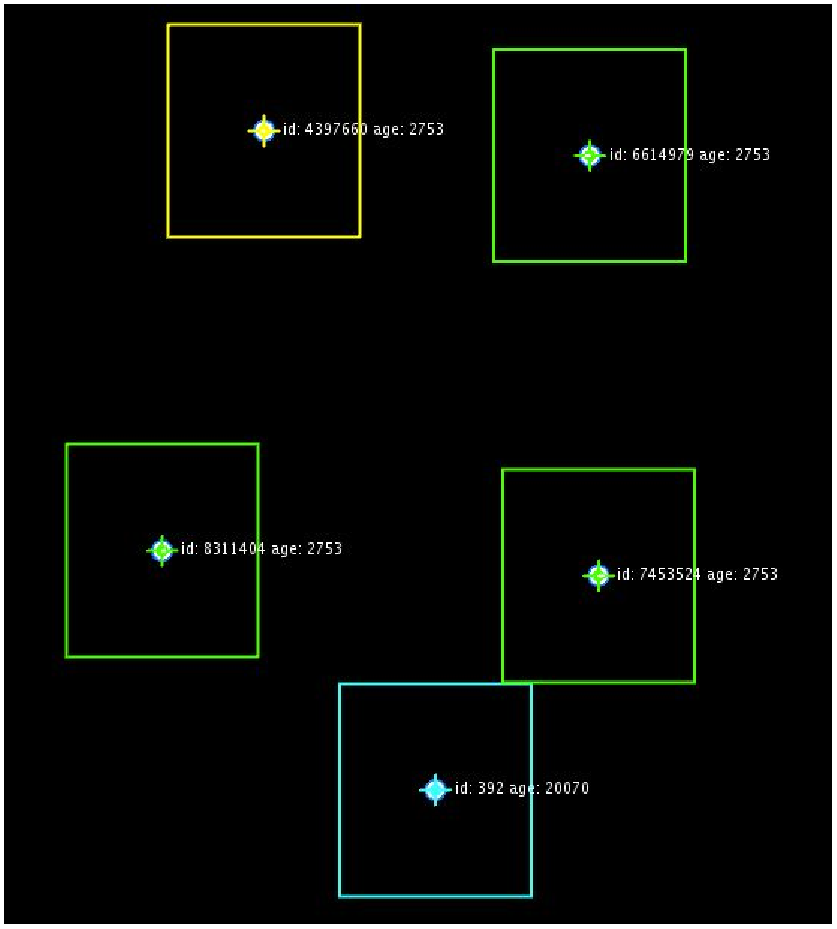
Message sent for each detected person:



```
/au/personEntered args0 arg1 ...
/au/personWillLeave args0 arg1 …
/au/personUpdated args0 arg1 …

Where args are:
```

```
0: pid (int)     // Personal ID ex: 42th person  to enter has pid=41
1: oid (int)     // Ordered ID ex: 3rd person still present has oid=2

2: age (int)     // Time on stage (in frame number)

3: centroid.x (float 0:1)       // Position projected to the ground
4: centroid.y (float 0:1)        // relative to the area size with
(0,0) on the top left

5: velocity.x (float -1:1)      // Speed and direction vector
6: velocity.y (float -1:1)

7: depth (float)            // Distance to sensor (in m) (not impl.)

8: boundingRect.x (float 0:1)    // Top left coord of bounding box
9: boundingRect.y (float 0:1)     // relative to the area size

10: boundingRect.width (float 0:1)    // Bounding box size
11: boundingRect.height (float 0:1)   // relative to area size

12: highest.x (float 0:1)       // Highest point placement
13: highest.y (float 0:1)
14: highest.z (float 0:1)       // Height of the person (not impl.)
```

Message sent each frame to describe the scene:



```
/au/scene args0 arg1 …

Where args are:
0: currentTime (int)             // Time (in frame number)
1: percentCovered (float 0:1)    // Percent covered
2: numPeople (int)               // Number of person

3: averageMotion.x (float -1:1)  // Average motion
4: averageMotion.y (float -1:1)

5: scene.width (int)             // Scene size (in pixel)
6: scene.height (int)
7: scene.depth (int)             // (not implemented)
```

3.2. Compatibles Media Servers

Any media server capable of receiving OSC can be made compatible with Augmenta. The following media servers are directly compatible with Augmenta:

- **Smode**

The following media servers are directly compatible with OSC and therefore can be made compatible with Augmenta:

- **Modulo**

3.3. Compatibles Softwares

Since Augmenta is based on an OSC protocol, it is naturally compatible with any software capable of receiving OSC.

Augmenta libraries are available for the following softwares:

- **Unity:** https://github.com/Theoriz/AugmentaUnity
- **Processing:** https://github.com/Theoriz/AugmentaP5
- **OpenFrameworks:** https://github.com/Theoriz/ofxAugmenta
- **Action Script 3**: https://github.com/Theoriz/AugmentaAS3
- **MaxMSP:** https://github.com/Theoriz/Augmenta4Max
- **Chataigne:** https://github.com/Theoriz/Augmenta-chataigne-module

3.4. Existing Tools & Community

Several tools have been developed around Augmenta by THEORIZ and the community using Augmenta. Most of them are available for download and open to contributions on THEORIZ Github at https://github.com/Theoriz/.

The following tools are available:

- **AugmentaFusion:** Closed source, it allows to merge different Augmenta data stream and helps calibrate and manage Augmenta nodes, cf Section 1.4.2.
- **AugmentaSimulator:** Open source, it allows to simulate an Augmenta node, cf Section 1.4.3. Available at https://github.com/Theoriz/Augmenta-Simulator
- **Area2OSC**: Open source, Area2OSC allows you to create polygons which will refer to detection zones. Osc messages will then be sent for every person entering or leaving the

zone. Available at https://github.com/Theoriz/Augmenta-Area2Osc

3.5. Library Implementation

séparer en :

# Recommended features to integrate

getNewest concept etc.

# Example of integration (THEORIZ Unity, processing and openframeworks implementation)

Implementing an Augmenta library for your software is easy and should be done with the following points mind.

Naming and conventions. The name of the library should be “Augmenta<Technology>”, following the library naming convention of the technology, like “ofxAugmenta” for OpenFrameworks or “AugmentaP5” for Processing. If there is no convention for naming the libraries, you can simply append the name of the technology, like AugmentaUnity. If needed, you can add the prefix “Augmenta” or “au” to specify that an object is linked to Augmenta. For everything else, use the code conventions of the technology you are implementing for.

Objects. The different objects (or classes) that should be implemented are:

- **AugmentaPerson:** Contains all the Augmenta data of one person sent in the OSC messages (cf Augmenta protocol in Section 3.1). If possible, has a Draw() function that displays all info (centroid, bounding box, pid, etc.).
- **AugmentaScene:** Contains all the Augmenta data of the scene sent in the OSC message sent each frame (cf Augmenta protocol in Section 3.1). The most important info that will be used is the scene width/height sent by Augmenta.
- **Main Object**. The main object should have the name of the lib. The instance of this object in the example code can be called augmentaReceiver or auReceiver. It should contain an AugmentaScene and an array (or equivalent) of AugmentaPerson representing all the people in the scene. It should be created with a given OSC port (or with default 12000) and start listening right away. It

should have tools like isConnected() to check if the port has already been successfully bound by the lib, or reconnect() to connect to a new OSC port on the fly. The main object will handle the OSC messages and should provide helping functions as described below.

Handling the OSC messages. The main object listens to four specific OSC messages: `/au/scene`: sent every frame to update the scene info, `/au/personEntered`: a new person entered the scene, `/au/personUpdated`: a person already present has been updated, `/au/personWillLeave`: a person already present is about to leave the scene. The scene info is directly stored into the AugmentaScene object. Every "person" message leads to either adding, updating or removing an AugmentaPerson in the array containing all the people present in the scene. If an update message is sent for a person that doesn't exist, it should be created. For safety reasons, if an AugmentaPerson has not been updated for a certain number of frames, it is considered missing and should be removed. Set the number of frames with a setTimeOut() function (with default 120 for example).

Helping functions. The main object should expose several functions:

- A function to get all AugmentaPerson, allowing to easily iterate over every person object. This function could be called **getPersonsArray**() for example.
- The two methods: **getNewestPerson**() and **getOldestPerson**(). They both return the AugmentaPerson which has the (respectively) smallest / greatest "age" value, or null if no one is in the scene. The oldest person is usually the most used, because you often want the first person that has entered the scene to keep interacting even if someone else entered after him. It can also prevent noise from disturbing the experience.
- The library should register and call the following callback methods, which should be triggered at every similar OSC event (or when a person is artificially added/removed). It provides the AugmentaPerson as an argument.

```
void personEntered (AugmentaPerson p) {
  print("Person entered: "+ p.pid );
};

void personUpdated (AugmentaPerson p) {
  print("Person updated: "+ p.pid );
};
```

```
void personWillLeave (AugmentaPerson p) {
  print(“Person left: "+ p.pid );
};
```

These functions can then be used by the user to define specific behaviours for each event.

Examples. The library should contain use cases example of the library. It should at least contain a minimalistic example with no external libraries dependency (except an OSC library if needed). This example should simply display a point for each AugmentaPerson and a point of a different color for the oldest/newest person. The draw() function of the AugmentaPerson should be used to draw all data so the user can quickly test the library.

Sharing. When you start working on your library, addon or plugin, or even when it is fully ready, do not hesitate to contact us at contact@theoriz.com so we can add it to the official repositories and documentation and help you share it with the world.

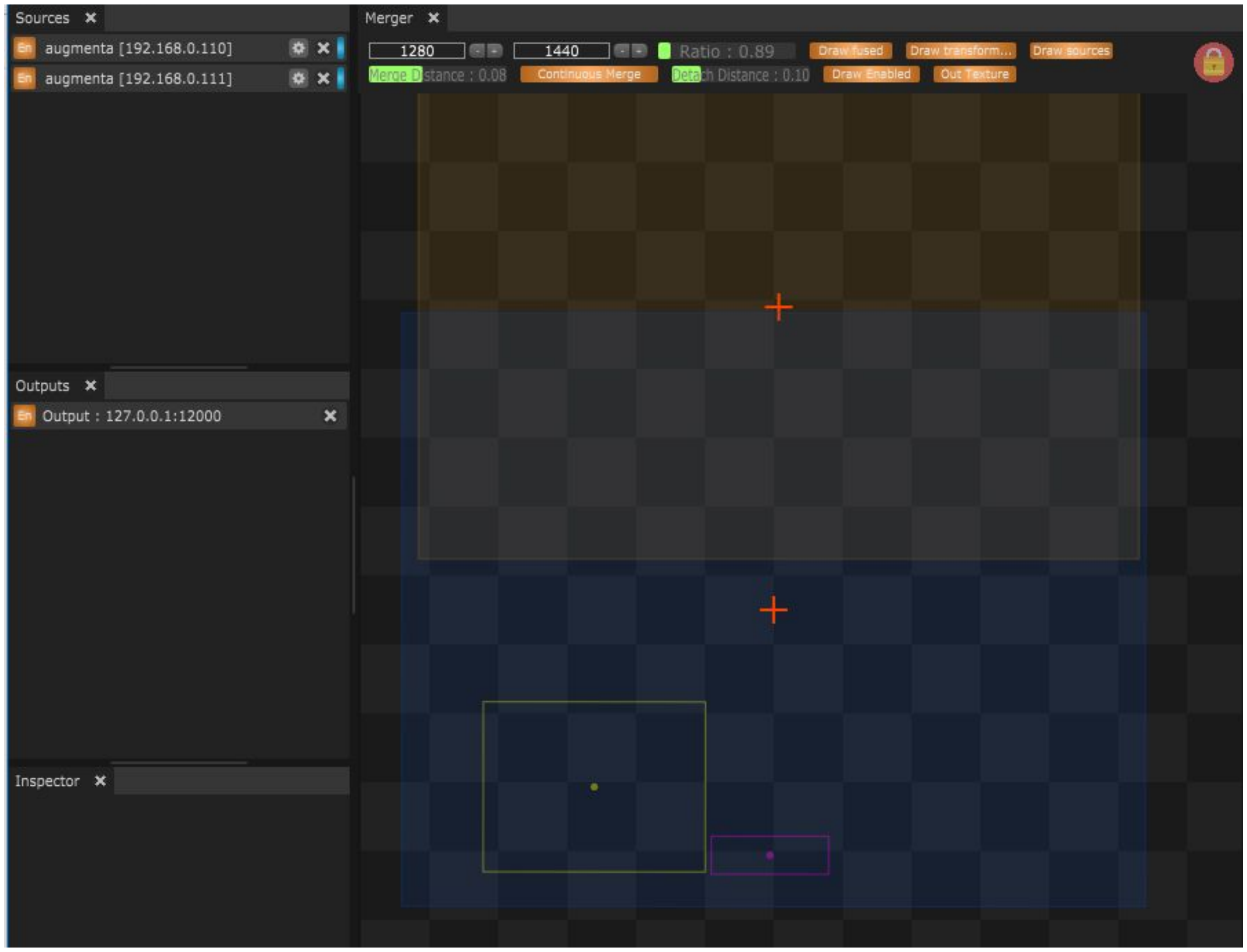
4. EXAMPLE USE CASES

4.1. Detecting people from above with two overlapping cameras

This use case represent a setup with two cameras looking at the floor on which people are walking. The floor is a flat white surface of around 40m² on which video projection is used to display visuals that people can interact with.

Augmenta setup

For this setup, we use two Xtion nodes positioned at X meters from the ground, looking down, with an overlap of X cm. The screenshot below shows how this setup look like in AugmentaFusion.



We can see in AugmentaFusion the two camera in the middle of the space with an overlap.

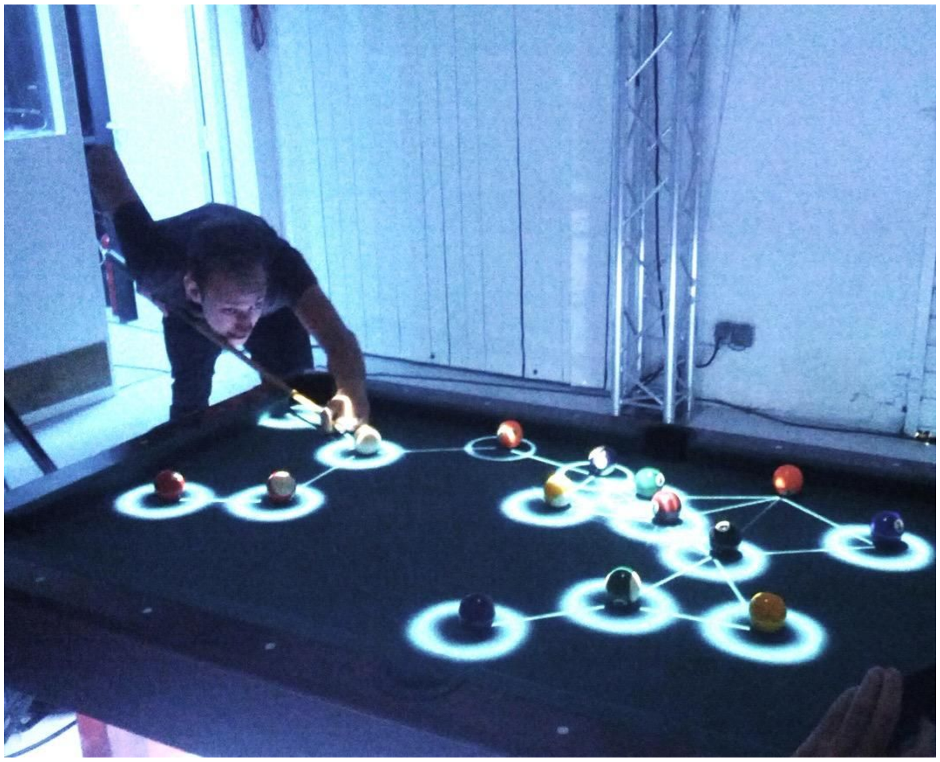
The nodes are configured with the following values:

- **Input: Z threshold** between 0 and 1 to keep the full depth range of the cameras. No masks since the camera only see the ground.
- **Background**: The **background mode** is set to Auto since the floor will always be the background (no hole or trap in the floor). The **plane detection** mode is also set to Auto since nothing on the floor could disturb the plane computation.
- **Processing:** The **height threshold** is set to keep the full height range. The **subtraction alpha and beta** parameters are set to 0.11 and 67 respectively since we are aiming for rather big objects (people). **Denoising** is set to 3 to remove the noise introduced by the floor material. The **max depth** is set to 1 in order to keep the full depth range.
- **Output**: The **blob reconstruction** is set to 4 with the **blob size** filter range of [0.21, 0.91] in order to get goog person blobs. The **smooth coefficient** is set to 0.25 and the

movement threshold to 1.35 in order to follow people walking slowly in the space.

In Augmenta

4.2. Detecting billiard balls



This use case is about using Augmenta to detect billiards balls and projecting visuals on the table, depending on the position of the balls.

Augmenta setup

For this setup

4.3. Using a LiDAR on a wall

4.4. Using two LiDARs on a floor

4.5. Using a manual background

4.6. A scene with no background

4.7. Calibrating without video projection

5. TROUBLESHOOTING

5.1. Contact

Do not hesitate to contact us at contact@augmenta-tech.com or support@augmenta-tech.com if you have feedback, questions or problems.

We are located at THEORIZ Studio, 36 rue Emile Decorps, 69100 Villeurbanne, France and you can keep in touch with our work at www.augmenta-tech.com and www.theoriz.com.

6. Changelog

2018 - 10 - 08 : First version for Smode seminar.