Object Tracker User Guide

Version 1.0

Technical Support

For technical support and the latest news of upgrades, documentation, and related products, visit the Vizrt web site at www.vizrt.com.

Created on

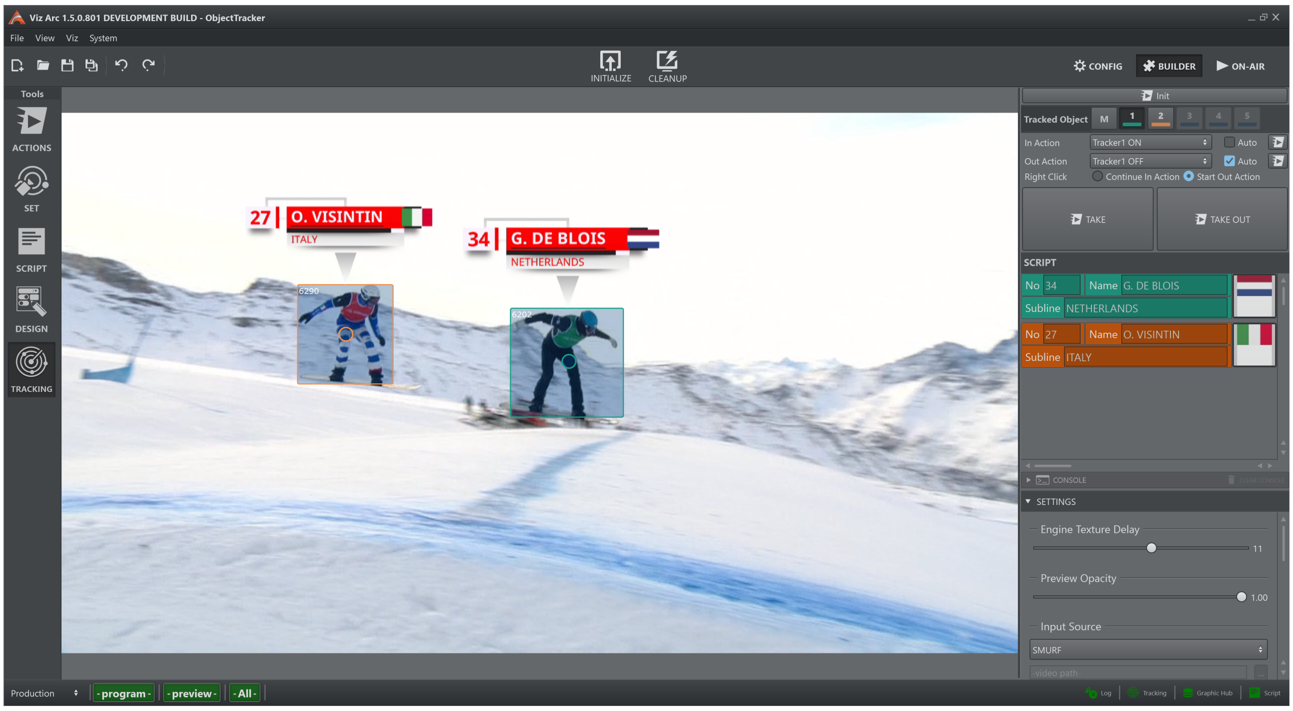
2022/06/09

Contents

1 Introduction

1.1 About Object Tracker

Object Tracker uses Viz AI Technology uses image-based tracking to detect objects on an incoming video stream of a Viz Engine. The Object Tracker is separate software that must run on a machine running the Viz Engine. Viz Arc offers a front end to operate the tracker.



Tracked Object: M 1 2 3 4 5
In Action: Tracker1 ON, Auto
Out Action: Tracker1 OFF, Auto
Right Click: Continue In Action / Start Out Action

SCRIPT
No 34, Name G. DE BLOIS, Subline NETHERLANDS
No 27, Name O. VISINTIN, Subline ITALY

Engine Texture Delay: 11
Preview Opacity: 1.00
Input Source: SMURF
video path

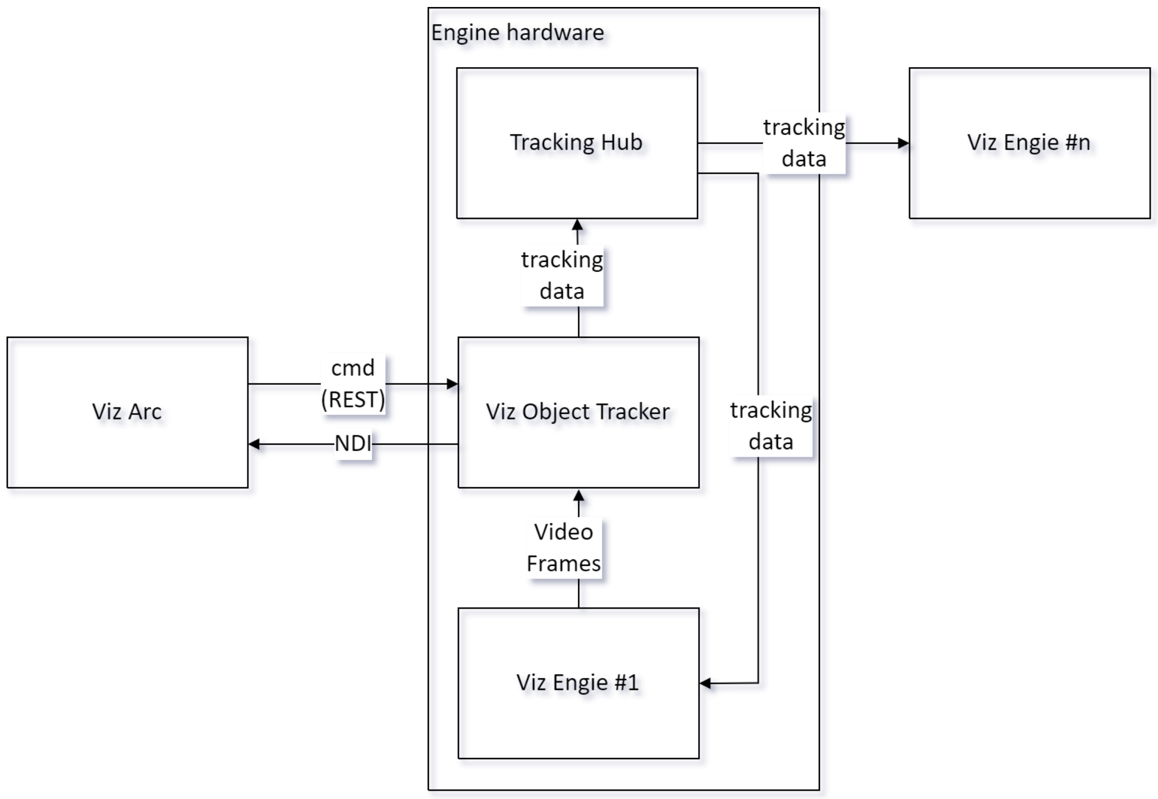
1.2 Related Documents

- Viz Engine Administrator Guide
- Tracking Hub Administrator Guide
- Tracking Hub Command Interface
- Viz Arc User Guide
- Viz Arc Script Guide

1.3 Feedback And Suggestions

We encourage feedback on our products and documentation. Please contact your local Vizrt customer support team at www.vizrt.com.

1.4 System Overview



The overall system is composed of at least two machines, one running the Viz Engine and the Object Tracker and another running Viz Arc as the control application.

- **Viz Engine**: Any signal connected to the Viz Engine (for example, live SDI input, clip channel input) can be exposed to the Object Tracker where it gets processed.
- **Tracking Hub**: Tracking Hub receives the tracking data and distributes it to the local Viz Engine or to any other additional Viz Engine.
- **Object Tracker**: The Object Tracker detects objects on the image stream from the Viz Engine and outputs an NDI stream of the stream and attaches the results as metadata information on each NDI image frame.
- **Viz Arc**: Viz Arc is the front end to the Object Tracker where the NDI is displayed with the respective metadata containing the detected objects. A user can select an object to be tracked. A REST command is sent to the Object Tracker to start the tracking on the selected object resulting in the generation of tracking data through Tracking Hub.

1.5 System Requirements

1.5.1 General

- Windows 10 (64-bit only)
- Windows Server 2012 R2
- Microsoft .NET Core Framework 3.1
- CodeMeter version 6.80 and later

1.5.2 Vizrt Software

- Viz Engine 4.3.0 and newer
- Viz Arc 1.5.0 and newer
- Tracking Hub 1.5.0 and newer

1.5.3 Minimum Hardware Requirements

720p, 1080i, 1080p

Object Tracker

- 1x HP Z8 or Dell R7920
- 2x RTX 4000 or 2X P6000
- 1x Matrox DSXLE4 or better

Viz Arc

- HP Zbook Mobile Workstation

OR

- 1x HP Z4
- 1x RTX 2000

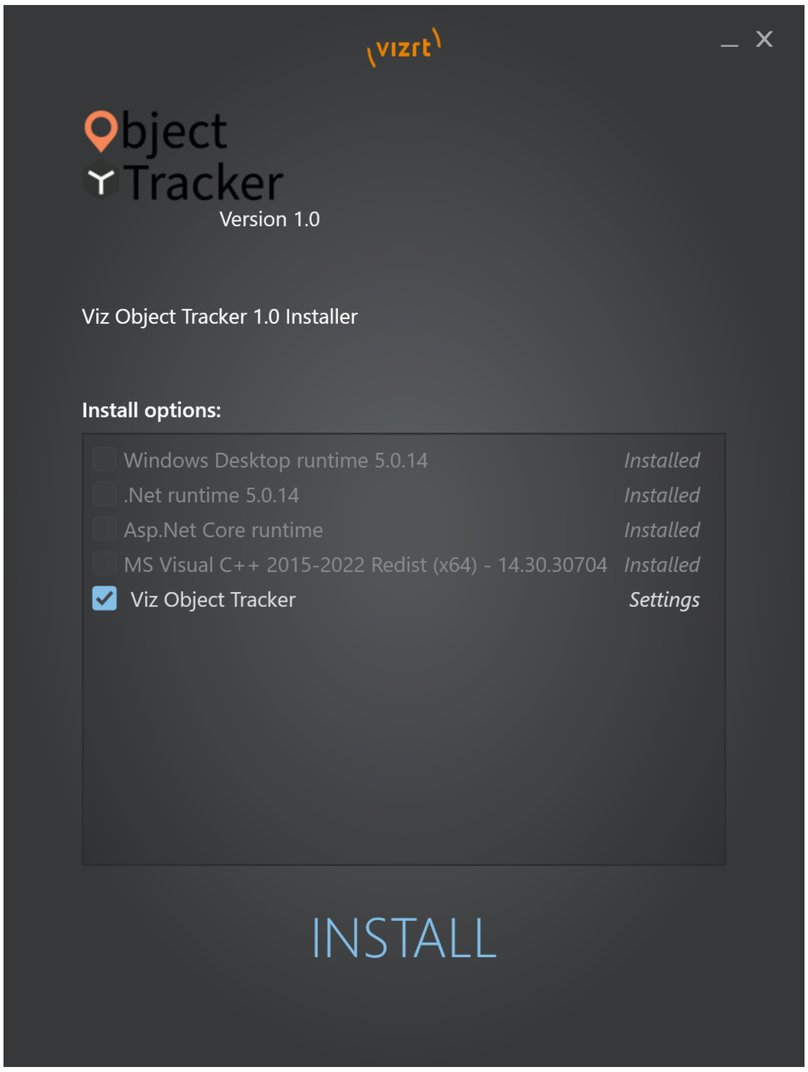
Information: Viz Arc must run on a separate Hardware than the Object Tracker.

SDI Preview

If there is a requirement for an SDI Preview in addition to the SDI PGM OUT, an additional Object tracker box is required.

2 Installation

Run the installer. Depending on the target system, you might need to install the required prerequisites.

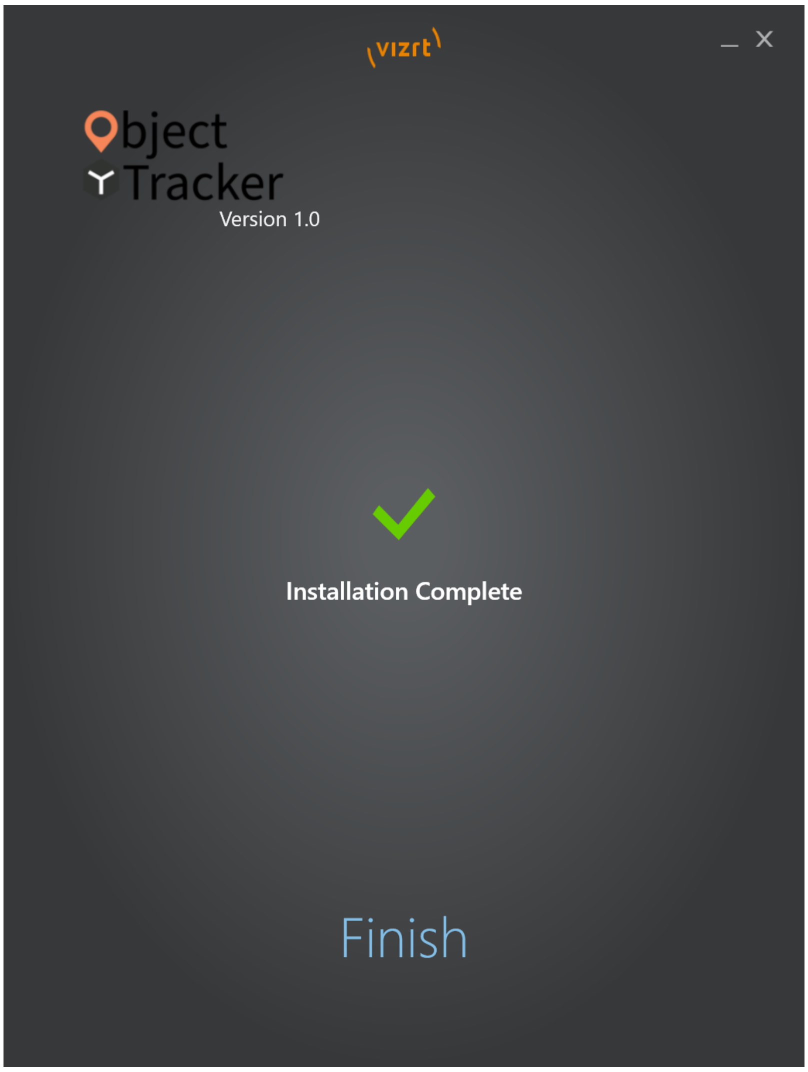


Viz Object Tracker 1.0 Installer

Install options:

☐ Windows Desktop runtime 5.0.14	*Installed*
☐ .Net runtime 5.0.14	*Installed*
☐ Asp.Net Core runtime	*Installed*
☐ MS Visual C++ 2015-2022 Redist (x64) - 14.30.30704	*Installed*
✓ Viz Object Tracker	*Settings*

Select the **Object Tracker** for installation and all required prerequisites if not already installed.



Press **Finish** to complete installation.

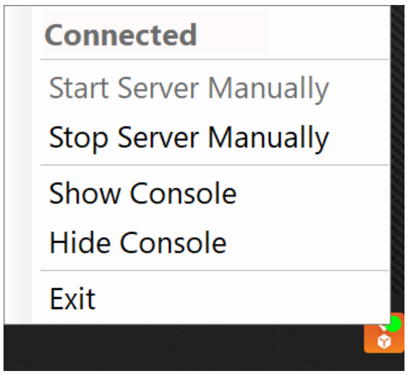
Once the Installation has finished the Object Tracker is started automatically and shows up with a green light in the icon tray .

VizObjectTrackerGUI
VizObjectTrackerGUI
On
Not measured

The installer adds Object Tracker into the **Startup Apps** so it starts up automatically after logging in.

2.1 Icon Tray

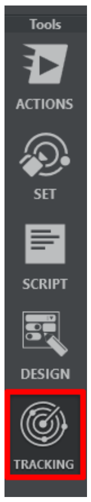
Right click the icon tray to get a context menu.

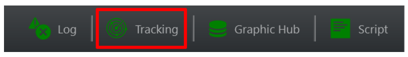


- **Connected/Disconnected:** Indicates whether the Object Tracer is running or not.
- **Start Server Manually:** Starts the service in case it has stopped.
- **Stop Server Manually**: Stops the service.
- **Show Console**: Visualizes the console with debugging information.
- **Hide Console**:Hides the console.
- **Exit**: Stops the service and closes the icon tray.

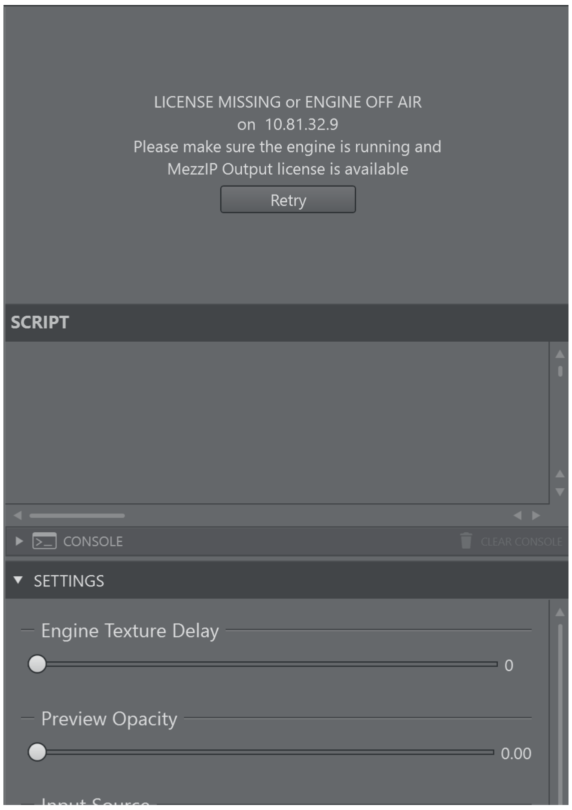
3 Prerequisites

After installing **Object Tracker** and after enabling and configuring the server in Viz Arc, you should see a green light in the status bar and an additional **TRACKING** button on the left hand tool bar in the Viz Arc UI.





- Load an appropriate scene on the renderer, a sample archive containing a scene can be found in the Viz Arc installation folder (*C:/Program Files/vizrt/VizArc 1.5/Resources/ ObjectTracker).*
- Viz Engine needs to run on on Matrox based hardware.
- Make sure you have the **MezzIP Output license** on the program Engine where the **Object Tracker** is running.



LICENSE MISSING or ENGINE OFF AIR
on 10.81.32.9
Please make sure the engine is running and
MezzIP Output license is available

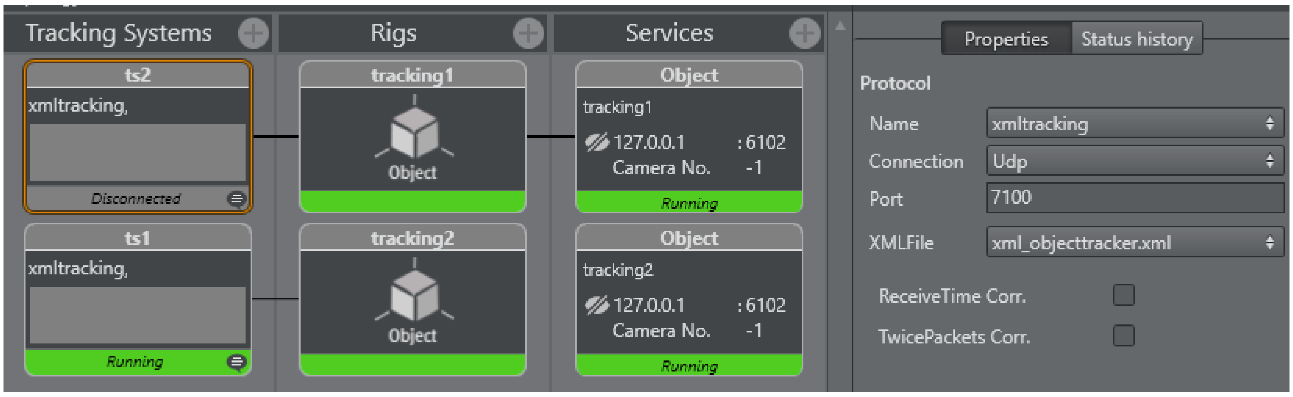
In case the license is missing or Viz Engine is off line, an error message appears as shown above.

4 Tracking Hub Configuration

4.1 General

You need at least Tracking Hub Version 1.5 installed. Copy the file *xml_objectTracker.xml* into the Tracking Hub **XMLTracker** directory (typically in *C:\ProgramData\vizrt\VizTH\XMLTracker*) After this operation restart the Tracking Hub.

4.2 Create A Configuration From Scratch



ts2: xmltracking, Disconnected
ts1: xmltracking, Running
tracking1: Object
tracking2: Object
Object: tracking1, 127.0.0.1 :6102, Camera No. -1, Running
Object: tracking2, 127.0.0.1 :6102, Camera No. -1, Running

Protocol

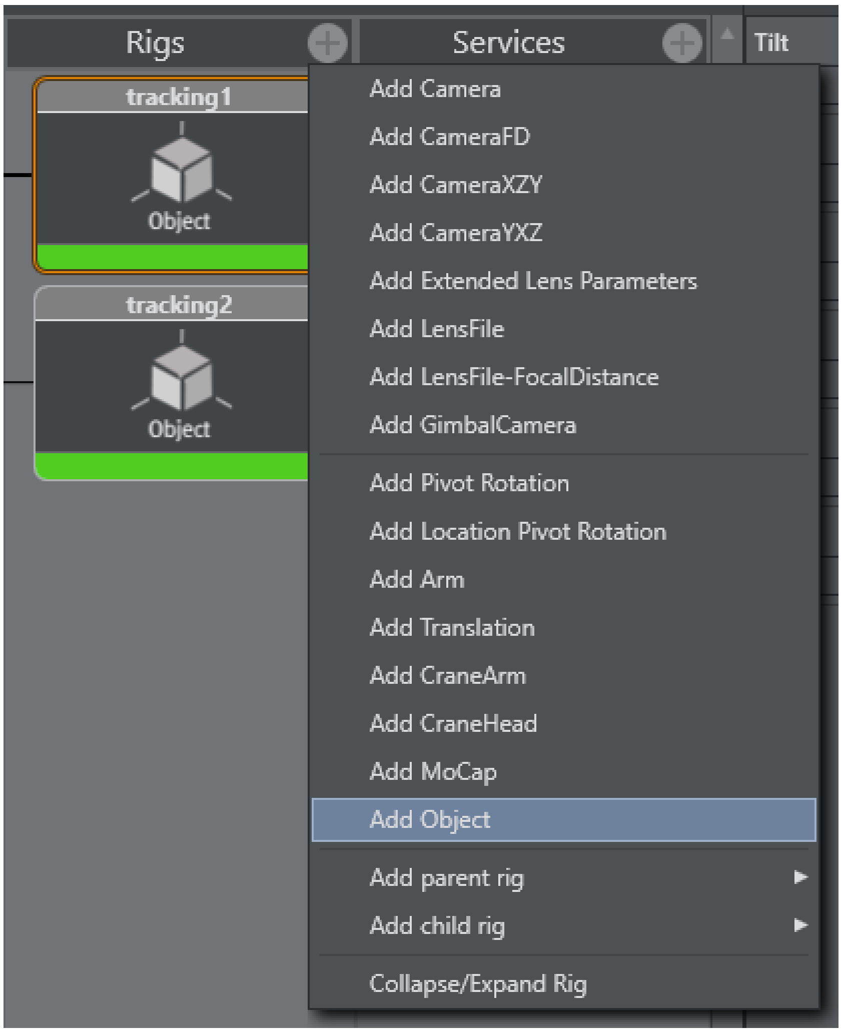
Name	xmltracking
Connection	Udp
Port	7100
XMLFile	xml_objecttracker.xml
ReceiveTime Corr.	☐
TwicePackets Corr.	☐

Create a new Tracking System and configure it as seen above.

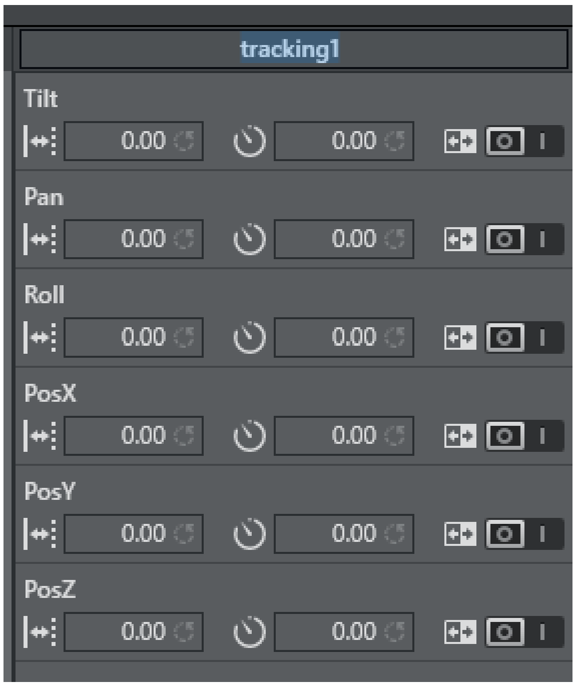
- Set the **Protocol** to *xmltracking.*
- Set the **Connection** to *UDP.*
- Set the **Port** to `7100` (representing the first tracking object), use port `7101` for the second tracker and so forth).
- Set **XMLFile** to *xml_objecttracker.xml* in the drop down menu.

4.3 Services Rig

Under **Services** select **Add Object** to create a new *Object Service*.



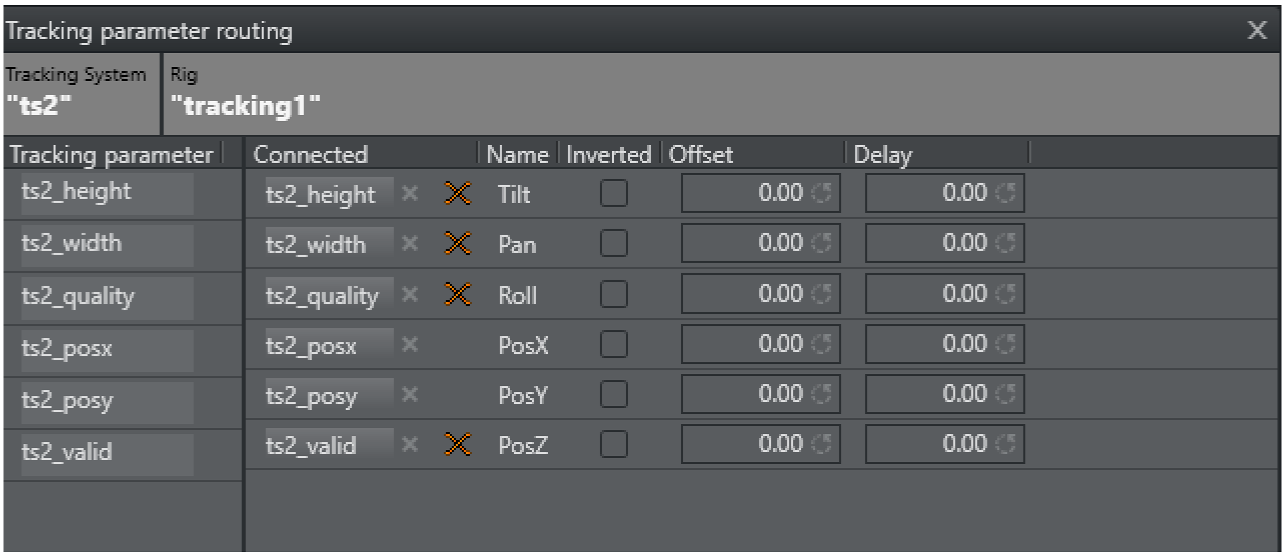
Rename the service `tracking1` (for the first tracker, use *tracker2* and so forth for any additional trackers). This is the name that is used in the Shared Memory on Viz Engine and in the scene to reference the tracker.



tracking1

Tilt	0.00	0.00
Pan	0.00	0.00
Roll	0.00	0.00
PosX	0.00	0.00
PosY	0.00	0.00
PosZ	0.00	0.00

Connect the tracking service to the newly created Object Service like this:



Tracking parameter routing

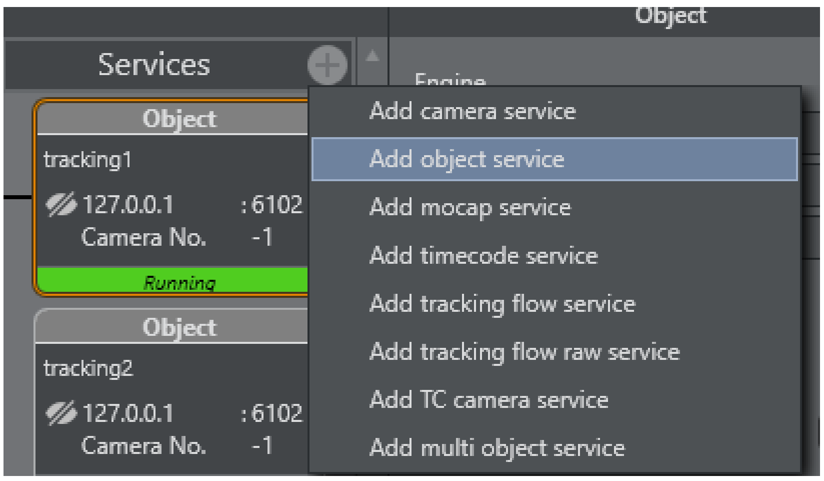
Tracking System "ts2" Rig "tracking1"

Tracking parameter	Connected	Name	Inverted	Offset	Delay
ts2_height	ts2_height	Tilt	☐	0.00	0.00
ts2_width	ts2_width	Pan	☐	0.00	0.00
ts2_quality	ts2_quality	Roll	☐	0.00	0.00
ts2_posx	ts2_posx	PosX	☐	0.00	0.00
ts2_posy	ts2_posy	PosY	☐	0.00	0.00
ts2_valid	ts2_valid	PosZ	☐	0.00	0.00

Map the tracking parameters as follows:

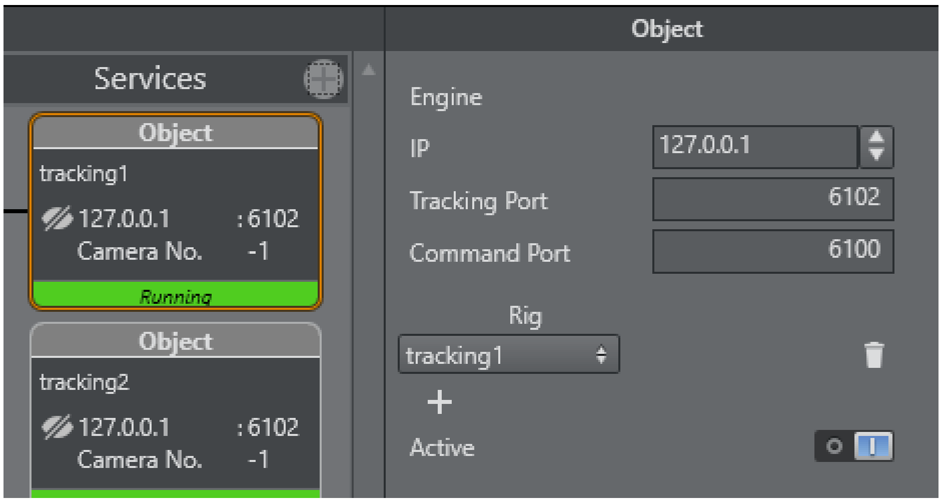
- tsX_height: **Tilt**
- tsX_width: **Pan**
- tsX_quality: **Roll**
- tsX_posx: **PosX**
- tsX_posy: **PosY**
- tsX_valid: **PosZ**

In the *Services* column, add an **Object Service.**



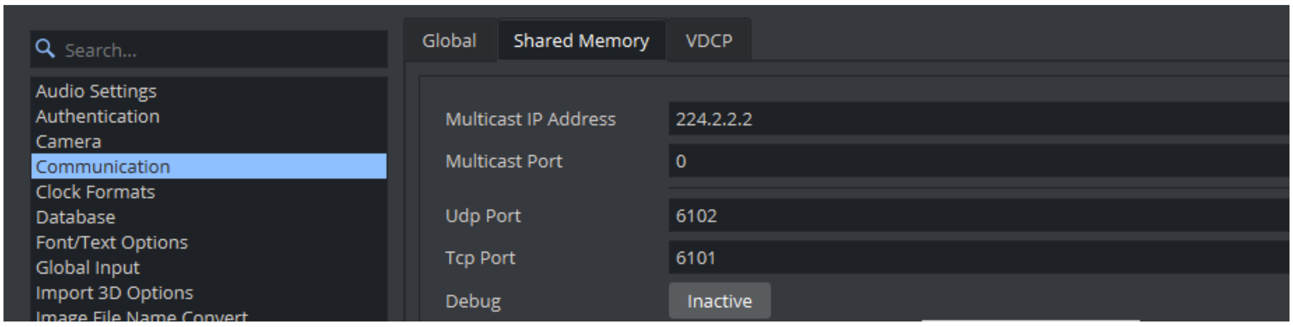
4.4 Configure The Object Service

Click the **+** symbol under the *Rig* label and select the rig from the middle column (for example, *tracking1*).



IP	127.0.0.1
Tracking Port	6102
Command Port	6100
Rig	tracking1

Select as IP the Viz Engine destination, tracking port should be the **shared memory UDP communication port** as it is configured on the **Viz Engine**, command port of the engine (typically `6100`).

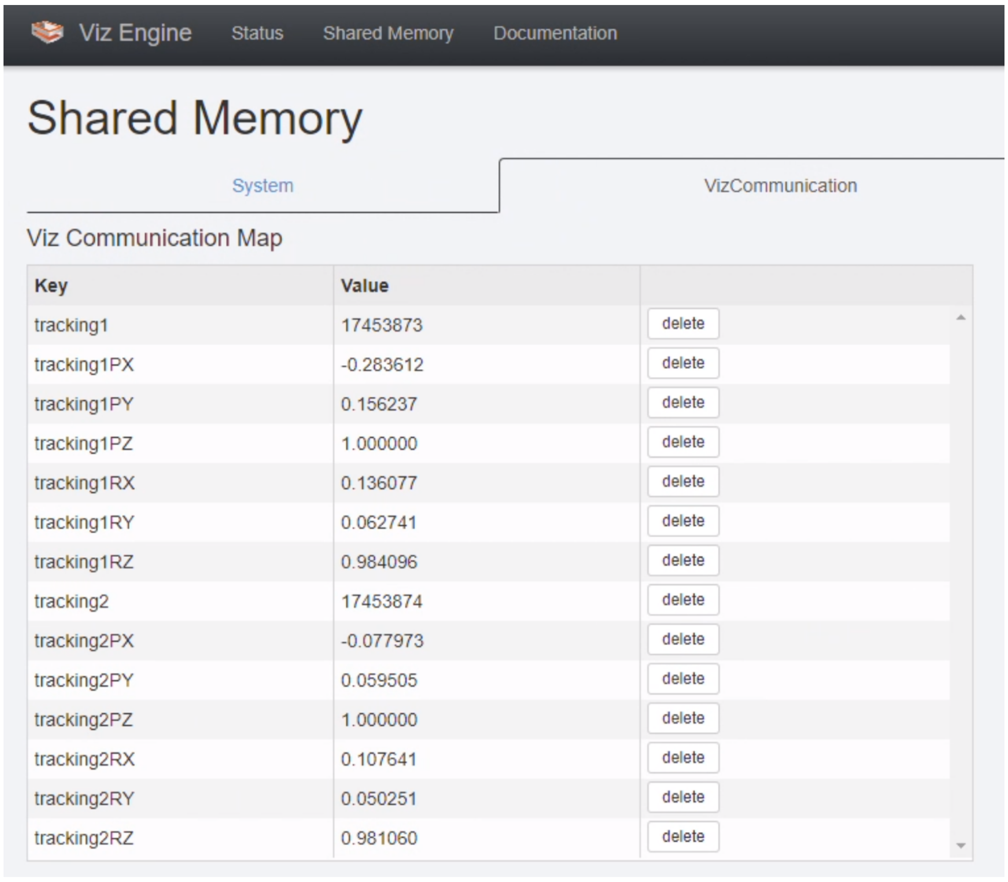


Multicast IP Address	224.2.2.2
Multicast Port	0
Udp Port	6102
Tcp Port	6101
Debug	Inactive

Viz Engine's Shared Memory configuration, pick any free port number (e.g. `6102`).

4.5 Tracking Data

The tracking data arrives from the **Viz Tracking Hub** through the *VizCommunication* shared memory map.



Key	Value	
tracking1	17453873	delete
tracking1PX	-0.283612	delete
tracking1PY	0.156237	delete
tracking1PZ	1.000000	delete
tracking1RX	0.136077	delete
tracking1RY	0.062741	delete
tracking1RZ	0.984096	delete
tracking2	17453874	delete
tracking2PX	-0.077973	delete
tracking2PY	0.059505	delete
tracking2PZ	1.000000	delete
tracking2RX	0.107641	delete
tracking2RY	0.050251	delete
tracking2RZ	0.981060	delete

The prefixes are given by the rig names in Tracking Hub, in this sample *tracking1* and *tracking2*.

- **PX** and **PY** are normalized coordinates in the range [-0.5, 0.5], where 0.0 represents the center of the screen.
- **PZ** is 0 when there is no tracking and 1 when tracking is valid.
- **RX** is the height of the bounding box and can be used to position the graphics. This value a normalized percent value of the image width (a value of 0.1 means 10% of the screen width).
- **RY** is the width of the bounding box and can be used to position the graphics. This value a normalized percent value of the image height (a value of 0.1 means 10% of the screen height).

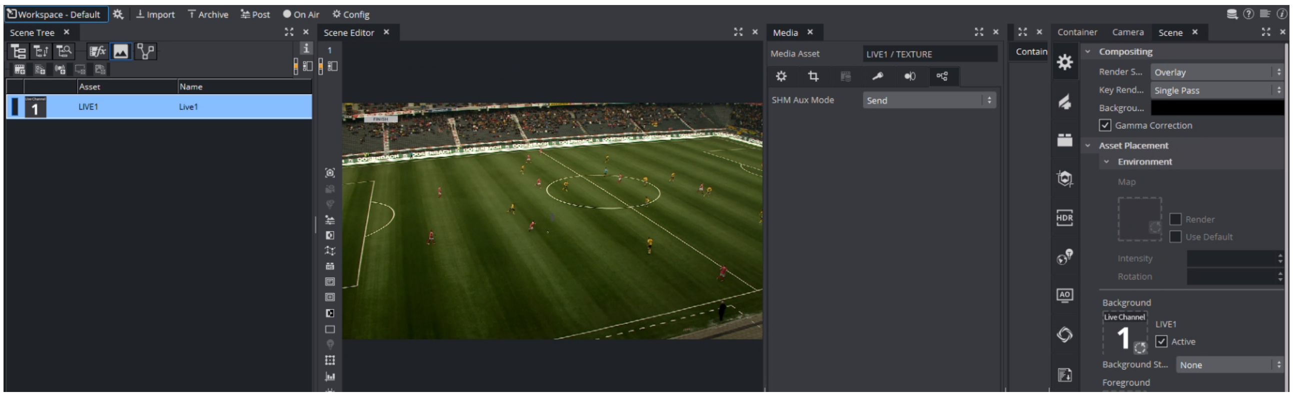
  Note: The bounding box values **RX** and **RY** are more heavily filtered and might lag a little bit. The values might be 0 in some cases, for example when manual or simple tracking is active.

- **RZ** gives an indication over tracking quality, 1.0 is the best quality. When the quality drops below 0.5 or less the quality begins to become critical and tracking might be lost soon.

5 Viz Scene Design

This section describes how to design a scene to be used with the object tracker.

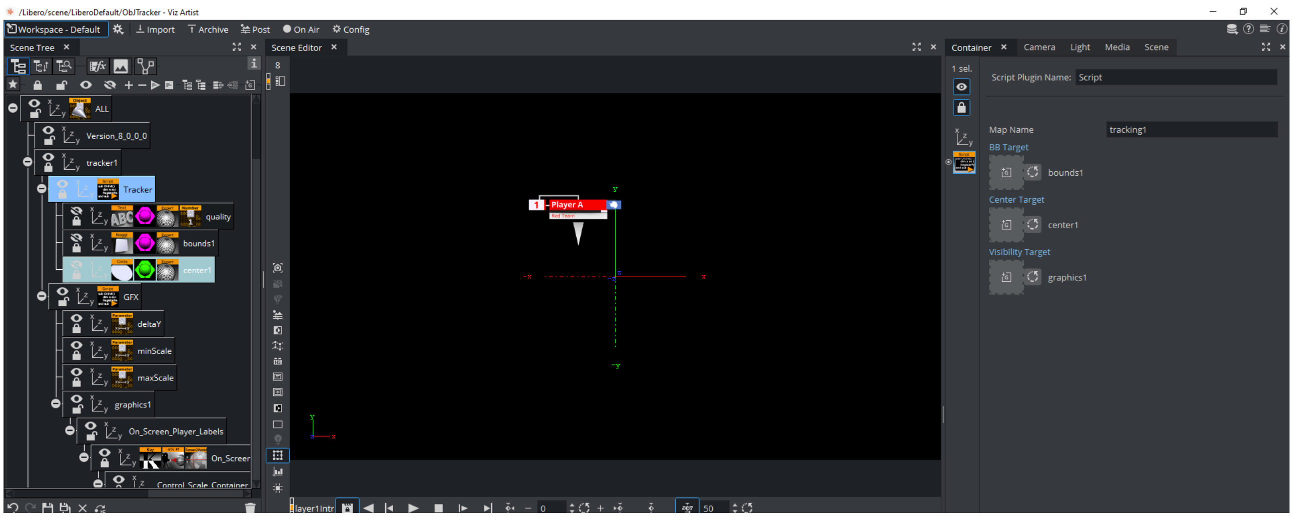
5.1 Live Video Input



The most important setting of a scene that can be used with the Object Tracker is the live input. The **SHM Aux Mode** must be set to *Send*. This allows the Object Tracker to read the input surface from the Engine. The Live Input asset needs to be present in the scene but does not necessarily need to be visible, so the result can be still composed on an external mixer.

It is possible to use other texture source such as clip channels. You need to make sure that the **SHM Aux Mode** is set to *Send* and that the configured **Input Key** (see **Configuration > Tracking** in the Viz Arc User Guide) matches the **SHMAuxKey** configuration of the Object Tracker's Viz Engine (for example, *viz_clip1_aux*, *viz_live1_aux*, etc.).

5.2 Container Scripts

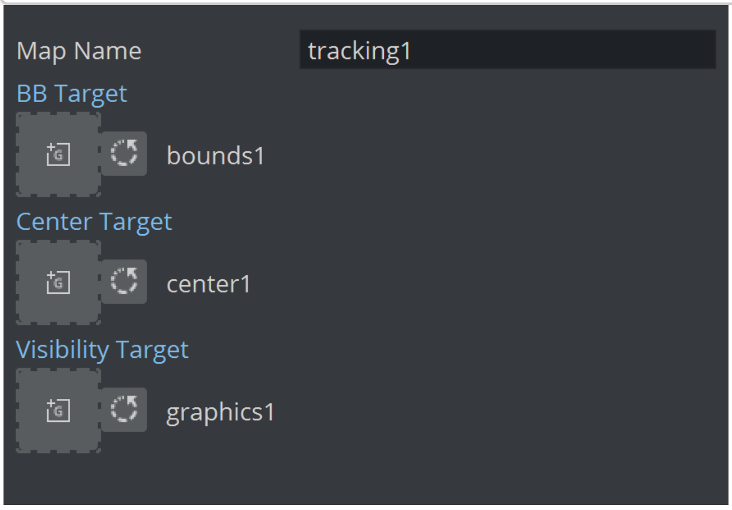


5.2.1 Main Tracking Script

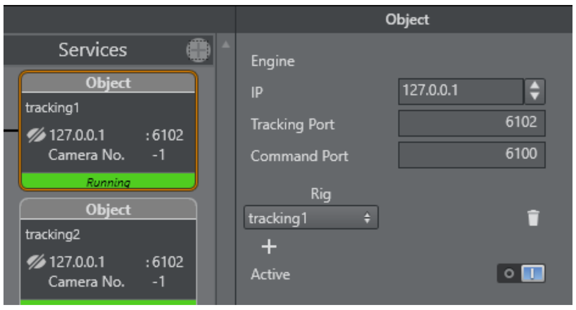
As the tracking data comes into Viz Engine through shared memory, these values need to be read and translated into screen coordinates. This can happen through a script:

```
dim MapName as String
dim screenWidth as Double
dim screenHeight as Double
dim mpi =  3.14159265359
dim degToRad = mpi/ 180.0

sub OnInit()
    MapName = GetParameterString( "mapName" )

    ' do  some trigonometry to get the screen width/height on the  0  plane.
    ' this  calculation works in orto and perspective mode but the camera
needs
    'to look straight down the z axis (no rotation). Works well with the
    ' default  camera settings

    screenWidth = tan(scene.CurrentCamera.Fovx* 0.5 *degToRad)*scene.Current
Camera.Position.z* 2
    screenHeight = tan(scene.CurrentCamera.Fovy* 0.5 *degToRad)*scene.Curren
tCamera.Position.z* 2
end sub

sub OnExecPerField()
    Dim bbTarget = GetParameterContainer( "bbTarget" )
    Dim centerTarget = GetParameterContainer( "centerTarget" )
    Dim visibilityTarget = GetParameterContainer( "visibilityTarget" )
    dim pos as Vertex
    dim bbHeight as Double
    dim bbWidth as Double
    dim quality as Double
    dim zActive as Boolean

    pos.x = VizCommunication.Map[MapName& "PX" ]
    pos.y = VizCommunication.Map[MapName& "PY" ]
    pos.z = VizCommunication.Map[MapName& "PZ" ]
    bbWidth = VizCommunication.Map[MapName& "RY" ]
    bbHeight = VizCommunication.Map[MapName& "RX" ]
    quality = VizCommunication.Map[MapName& "RZ" ]
    zActive = cBool(pos.z ==  1 )

    'println  zActive &  " "  & pos.x
     if ( visibilityTarget.Active ==  false  AND zActive ==  true ) Then
      visibilityTarget.Active =  true
       this .Active =  true
      Println  "ACTIVATE GFX ++++++++++++++++"
    end  if
     if ( visibilityTarget.Active ==  true  AND zActive ==  false ) Then
```

```
        visibilityTarget.Active =  false
         this .Active =  false
        Println  "DEACTIVATE GFX ++++++++++++++++"
    end  if

     if  visibilityTarget.Active ==  true  then
        centerTarget.position.x =  pos.x*screenWidth
        centerTarget.position.y = -pos.y*screenHeight

        findSubContainer( "quality" ).GetFunctionPluginInstance( "ControlNum" )
.SetParameterString( "input" , cStr(quality* 100.0 ))

        bbTarget.position.x =  pos.x*screenWidth
        bbTarget.position.y = -pos.y*screenHeight
        bbTarget.Scaling.x  = bbWidth*screenWidth/ 100
        bbTarget.Scaling.y  = bbHeight*screenHeight/ 100
    end  if
end sub

sub OnInitParameters()
    RegisterParameterString( "mapName" ,  "Map Name" ,  "tracking1" ,  20 , 2
00 ,  "" )
    RegisterParameterContainer( "bbTarget" ,  "BB Target" )
    RegisterParameterContainer( "centerTarget" ,  "Center Target" )
    RegisterParameterContainer( "visibilityTarget" ,  "Visibility Target" )
end sub

sub OnParameterChanged(parameterName As String)
    MapName = GetParameterString( "mapName" )
end sub
```


Map Name tracking1
BB Target
bounds1
Center Target
center1
Visibility Target
graphics1

In a first step, the script calculates in the **OnInit** method the available width and height of the viewport on the zero Z plane. In the **OnExecPerField** method, the tracking data is read out using a given shared memory key (set through the *Map Name* parameter, default is *tracking1*). The name of the key must match the name of the Rig selected in the Tracking Hub's Object Service.



The script then switches on and off the target container defined in the **Visibility Target** parameter when tracking begins and tracking ends or is lost. This can be also replaced with a stage command for example. The parameter container *Center Target* is updated on every field to the actual tracking position. This is the reference point of the tracker.

Parent Transformations: The script above does not consider any parent transformations of the target container. Make sure the target container contains no additional parent transformations

The **BB Target** container parameter (which might contain a Noggi or Rectangle plug-in) gets resized arroding to the atracked width and height of the object.

Bounding Box Sizes: The bounding box sizes might become zero. This is always the case for simple and manual tracking.

5.2.2 Follow Bounding Box Script

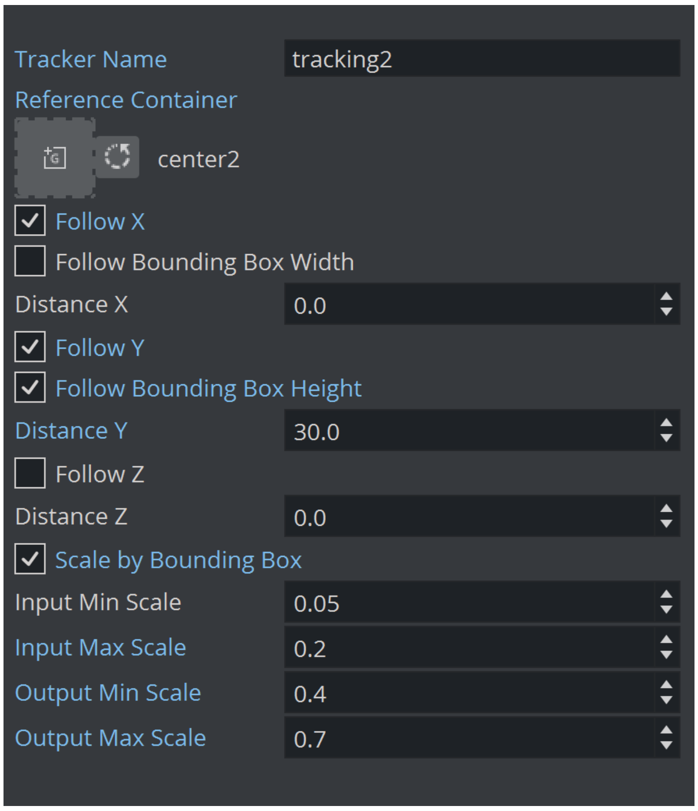
Another container script can be used to position graphics in relation to the bounding box of the detected object:

Poistioning Script

```
Dim  screenWidth as  Double
Dim  screenHeight as  Double
dim mpi = 3.14159265359
dim degToRad = mpi/180.0

sub OnInit()
    screenWidth =
tan(scene.CurrentCamera.Fovx*0.5*degToRad)*scene.CurrentCamera.Position.z*2
    screenHeight =
tan(scene.CurrentCamera.Fovy*0.5*degToRad)*scene.CurrentCamera.Position.z*2
end sub

sub OnInitParameters()
    RegisterParameterString( "mapName" , "Tracker Name" , "tracking1" ,
20,200, "" )
```

```
    RegisterParameterContainer( "ref" ,  "Reference Container" )
    RegisterParameterBool( "followX" ,  "Follow X" ,  False )
    RegisterParameterBool( "followBBX" ,  "Follow Bounding Box Width" ,
 False )
    RegisterParameterDouble( "deltaX" ,  "Distance X" , 0.0, -1000000,
1000000)
    RegisterParameterBool( "followY" ,  "Follow Y" ,  False )
    RegisterParameterBool( "followBBY" ,  "Follow Bounding Box Height" ,
 False )
    RegisterParameterDouble( "deltaY" ,  "Distance Y" , 0.0, -1000000,
1000000)
    RegisterParameterBool( "followZ" ,  "Follow Z" ,  False )
    RegisterParameterDouble( "deltaZ" ,  "Distance Z" , 0.0, -1000000,
1000000)

    RegisterParameterBool( "doScale" ,  "Scale by Bounding Box" ,  False )

    RegisterParameterDouble( "inMinScale" ,  "Input Min Scale" , 0.0, 0,
1.0)
    RegisterParameterDouble( "inMaxScale" ,  "Input Max Scale" , 1.0, 0,
1.0)

    RegisterParameterDouble( "minScale" ,  "Output Min Scale" , 0.0, 0,
1000000)
    RegisterParameterDouble( "maxScale" ,  "Output Max Scale" , 1.0, 0,
1000000)
 end sub


 sub OnExecPerField()
     Dim  bbHeight as  Double
     Dim  bbWidth as  Double
     Dim  trackerName as  String

    trackerName = GetParameterString( "mapName" )

    bbHeight = VizCommunication.Map[trackerName& "RX" ]
    bbWidth = VizCommunication.Map[trackerName& "RY" ]

    if GetParameterBool( "followX" ) == true  Then
        if(GetParameterBool( "followBBX" ) == true) then
            position.X = GetParameterContainer( "ref" ).Position.X + bbWidth
* screenWidth*0.5 + GetParameterDouble( "deltaX" )
        else
            position.X = GetParameterContainer( "ref" ).Position.X +
GetParameterDouble( "deltaX" )
        end if
    end  If

    if GetParameterBool( "followY" ) == true  Then
        if(GetParameterBool( "followBBY" ) == true) then
```

```
            position.Y = GetParameterContainer( "ref" ).Position.Y +
bbHeight * screenHeight*0.5 + GetParameterDouble( "deltaY" )
        else
            position.Y = GetParameterContainer( "ref" ).Position.Y +
GetParameterDouble( "deltaY" )
        end if
    end  If

    if GetParameterBool( "followZ" ) == true  Then
        position.Z = GetParameterContainer( "ref" ).Position.Z +
GetParameterDouble( "deltaZ" )
    end  If

    if GetParameterBool( "doScale" ) == true  Then
         Dim  inMin as  Double
         Dim  inMax as  Double
         Dim  outMin as  Double
         Dim  outMax as  Double
         Dim  scale = bbHeight

        inMin = GetParameterDouble( "inMinScale" )
        inMax = GetParameterDouble( "inMaxScale" )

        outMin = GetParameterDouble( "minScale" )
        outMax = GetParameterDouble( "maxScale" )

        scale = Max(inMin, scale)
        scale = Min(inMax, scale)

        scale = (scale-inMin)/(inMax-inMin)
        scale = outMin + scale*(outMax-outMin)

        scaling.x = scale
        scaling.y = scale
        scaling.z = scale
    end  If
end sub
```



Tracker Name	tracking2
Reference Container	center2
✓ Follow X	
Follow Bounding Box Width	
Distance X	0.0
✓ Follow Y	
✓ Follow Bounding Box Height	
Distance Y	30.0
Follow Z	
Distance Z	0.0
✓ Scale by Bounding Box	
Input Min Scale	0.05
Input Max Scale	0.2
Output Min Scale	0.4
Output Max Scale	0.7

The script uses the **Reference Container** (that represents the tracked point), it is used very much like the Autofollow plug-in. Additionally, to follow the position of the reference container, it uses a few more parameters that determine an additional offset relative to the tracked object's bounding box.

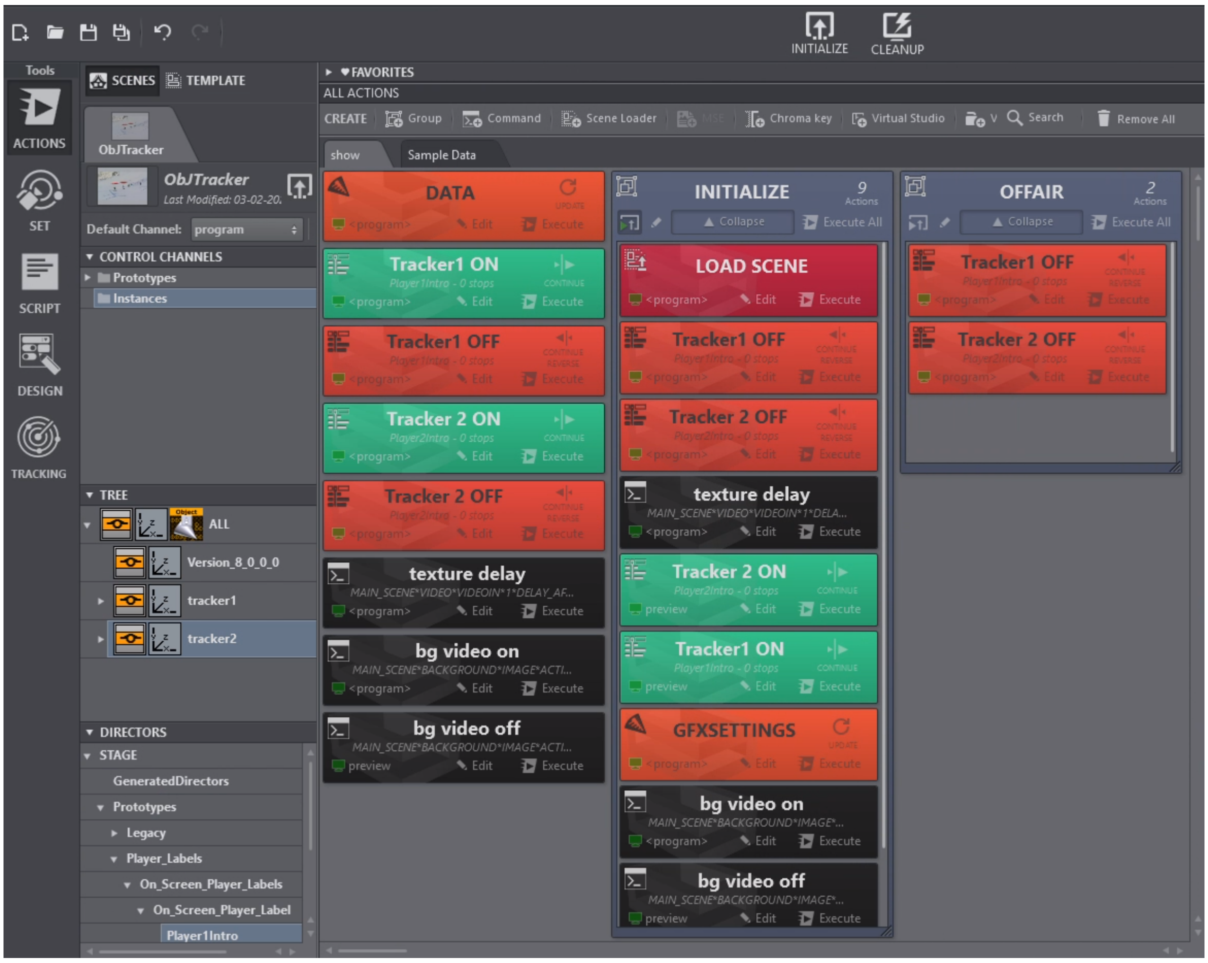
- **Follow X**: Allows the graphics follow the horizontal movement of the tracked object.
- **Follow Bounding Box Width**: Positions the graphics on the right hand side of the bounding box.
- **Distance X**: Adds constant horizontal offset.
- **Follow Y**: Allows the graphics follow the vertical movement of the tracked object.
- **Follow Bounding Box Height**: Aligns the graphics on top of the bounding box.
- **Distance Y**: Adds constant vertical offset.
- **Follow Z**: Unused.
- **Distance Z**: Adds constant depth offset.
- **Scale by Bounding Box**: Uses Bounding Box size to scale graphics.
- **Input Min Scale**: Sets minimum input scale of the Bounding Box height. The input height is normalized between 0 and 1 (where 1 is the full screen height, 0.1 is 10% of the screen height etc.).
- **Input Max Scale**: Sets maximum input scale of the Bounding Box height. The input height is normalized between 0 and 1 (where 1 is the full screen height, 0.1 is 10% of the screen height etc.).
- **Output Min Scale**: Maps minimum output scale of the graphics.
- **Output Max Scale**: Maps maximum output scale of the graphics.

Using the above sample values, a bounding box with height of 0.05 or smaller (thus 5% of the screen height or smaller) results in a scaling of 0.4 (Output Min Scale). A bounding box of height 0.2 or larger (20% or larger of the screen height) is scaled to 0.7.

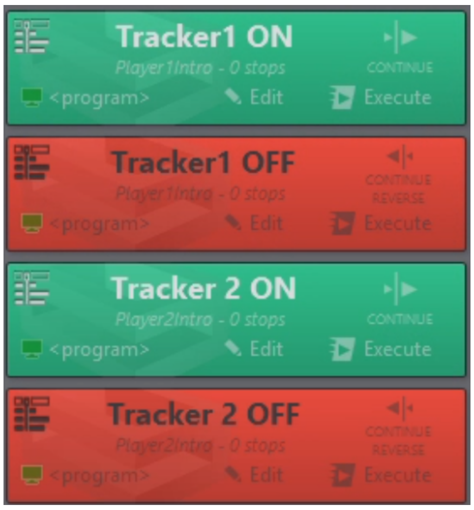
All values in between 0.05 and 0.2 are interpolated linearly between 0.4 and 0.7. Adjust those values to suit your graphics. The sample in this script considers only the height of the bounding box, but it could be easily changed to consider the surface of the bounding box or the width only.

6 Preparing Operations

Before being able to put graphics On Air, you need to define actions or templates that can bring those graphics in and out. Those actions can vary from simply switching on and off scene containers, to animating in and out directors or more complex templates.



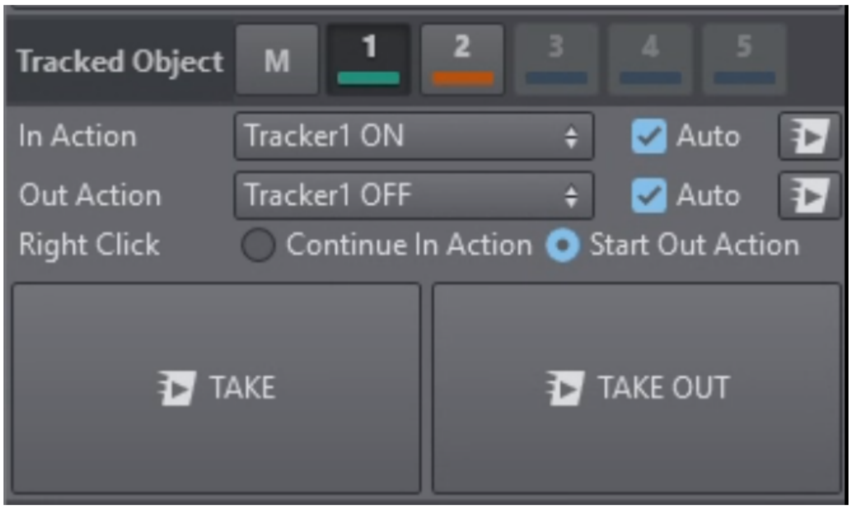
For example, each tracker has a director action animating the graphics in and out respectively.



Those actions can be used to trigger an in animation when starting to track object and an out animation when the tracking is lost or the operator takes it Off Air.

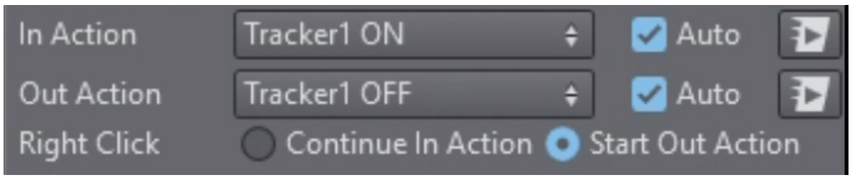
6.1 Tracking Panel

This panel allows you to define which actions to execute for in and out and it allows you to operate the graphics.



In Action: Tracker1 ON ✓ Auto
Out Action: Tracker1 OFF ✓ Auto
Right Click: Continue In Action [● Start Out Action]

6.1.1 In/out Actions



In Action: Tracker1 ON ✓ Auto
Out Action: Tracker1 OFF ✓ Auto
Right Click: Continue In Action [● Start Out Action]

Select an **In Action** from the dropdown that should be triggered when an object is selected for tracking.

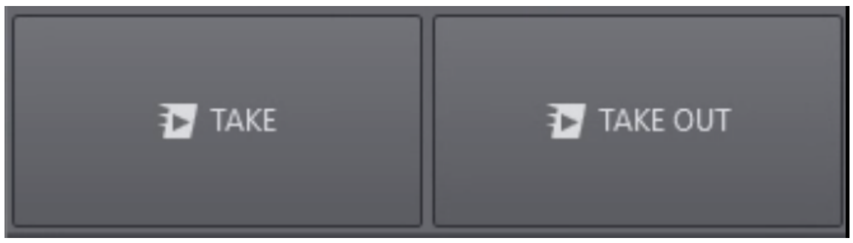
Select an **Out Action** from the dropdown that should be triggered when an object is de-selected for tracking or when tracking is lost.

Check **Auto** checkboxes when those actions should be executed automatically on selection of a tracked object and when the object is lost respectively.

The execute push buttons can be used to execute the actions manually. Clicking using the right mouse button on the execute action buttons executes the respective actions on the preview channel.

Under the option Right Click select what should happen when the user clicks the right mouse button on a tracked object or when tracking is lost. Either Start Out Action that means to execute what is selected as Out Action or select Continue In Action that execute a continue on the selected In Action. The latter option might be useful when the selected In Action is a template that might execute its continue as the Out Action.

6.1.2 Take and Take Out



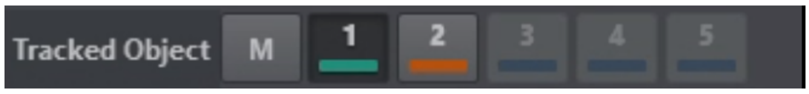
When clicking the **Take** button, all currently actively tracked trackers are executed their predefined **In Action**. The keyboard shortcut **SPACEBAR** might be used instead.

When clicking the **Take Out** button, all trackers execute their predefined **Out Action**. The keyboard shortcut **BACKSPACE** might be used instead.

Clicking using the right mouse button on the buttons executes the respective actions on the preview channel.

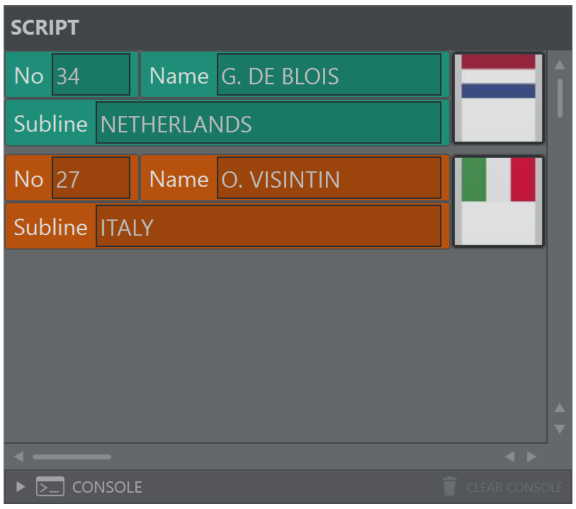
6.1.3 Tracked Object

Select the active tracker from the available trackers. Select the **M** toggle button to switch into manual mode.



The **keyboard shortcut Tab** allows you to switch from one active tracker to the next. The **M** toggle toggles the trackers between manual mode and normal mode. Right click the toggle buttons of the trackers to change the assigned color.

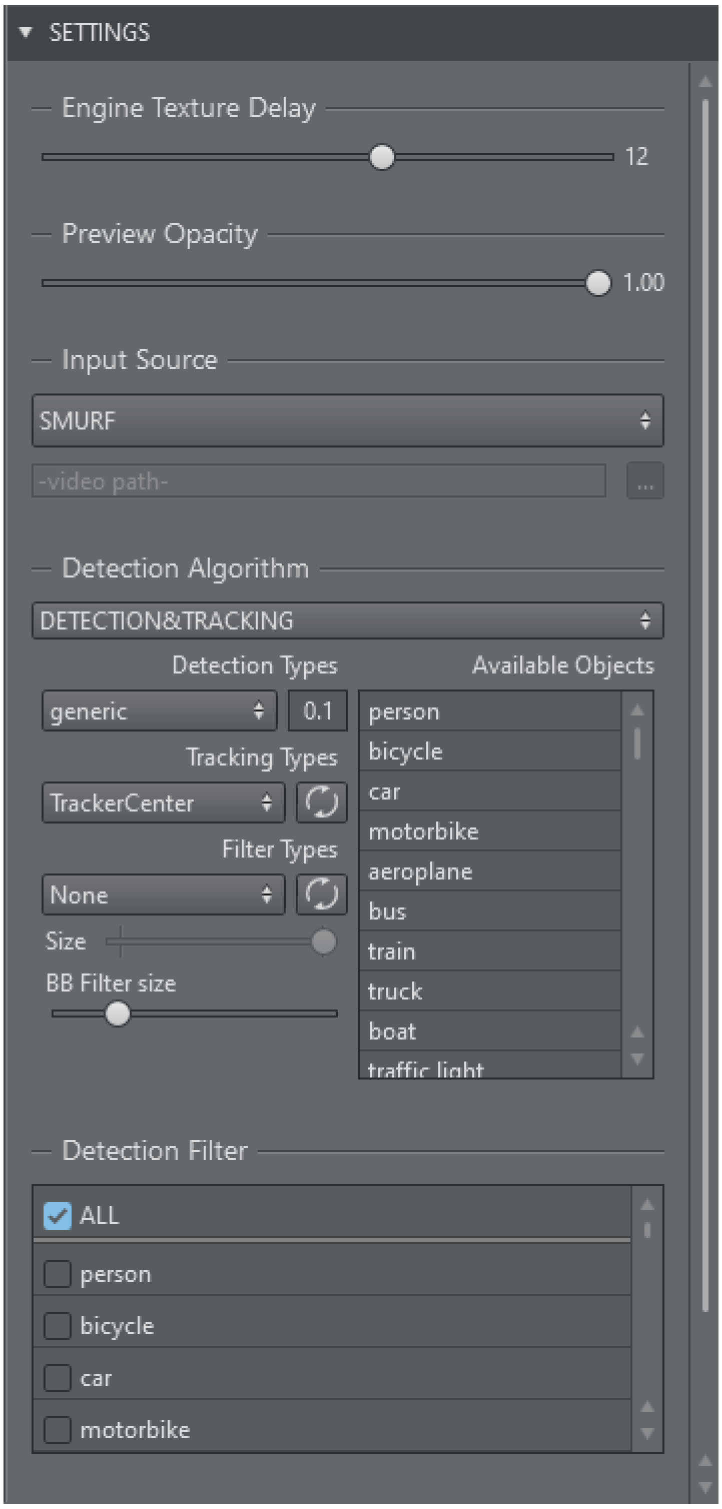
6.2 Scripting Panel



The scripting panel can be used for custom scripts that might be tailored for specific use cases. It allows easy and quick data entry and features the full power Viz Arc's scripting capabilities. Tracking related callbacks and functions can be used within the scripting for maximum flexibility.

7 Settings Panel

In the settings panel, you can select which tracking algorithm to use and which neural network to work with for the AI tracker. Each tracker and algorithm has specific settings which are exposed in the panel.



SETTINGS

Engine Texture Delay: 12

Preview Opacity: 1.00

Input Source: SMURF

-video path-

Detection Algorithm: DETECTION&TRACKING

Detection Types: generic 0.1

Tracking Types: TrackerCenter

Filter Types: None

Size

BB Filter size

Available Objects: person, bicycle, car, motorbike, aeroplane, bus, train, truck, boat, traffic light

Detection Filter:

- [x] ALL
- [ ] person
- [ ] bicycle
- [ ] car
- [ ] motorbike

Apply Settings: Please note that most settings that can be changed in the settings panel do not get applied as you change them in the Object Tracker. You need to press the **Init** button to apply your changes on the server.

7.1 Detection Algorithm

7.1.1 Manual

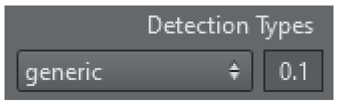
This mode allows you to track manually using the mouse pointer, touch screen or pen input.

7.1.2 Detection and Tracking

This option selects the AI tracker based on a neural network.

Detection Types

The network to be used can be selected under **Detection Types**.

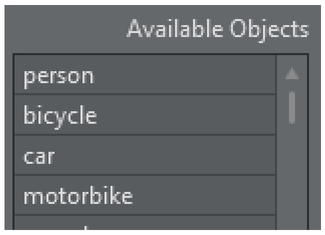


Detection Types
generic
0.1

The **generic** network is always available and offers a couple of standard objects that it recognizes (for example, people, cars, etc.).

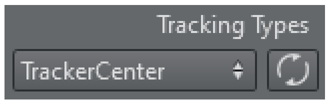
The number besides the dropdown represents a threshold in the range of 0.0 to 1.0. It represents a factor of confidence of how well an object has been detected. The higher the value the higher the probability the detection is correct. Object with a low confidence score can be ignored by using this threshold value. Depending on the quality of the images and the neural network this value can be adjusted to improve the overall tracking.

The list of available objects is shown under *Available Objects*. This is the list of objects that are recognized by the neural network.



Available Objects
person
bicycle
car
motorbike

Tracking Types



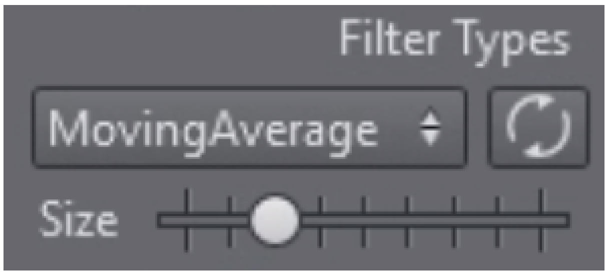
Tracking Types
TrackerCenter

The Tracking Types dropdown lets you choose what kind refined tracking to use inside of the detected bounding box. The button next to the drop down sends the new settings to the Object Tracker without having to use the global initialize button.

- **TrackerCenter**: Extracts features around the center of the bounding box. A gravity force re-centers the tracking point toward the center of the bounding box over time to avoid drifting.
- **TrackerTop**: Extracts features towards the top area of the bounding box. This is typically used when tracking persons as the arms and legs might move too much for a stable tracking, while the area of the chest and head are more stable and yield better tracking results. As for the tracker above, a gravity force re-centers the tracking point toward the center of the bounding box over time to avoid drifting.
- **FeaturesCenter**: Same as TrackerCenter without the gravity. The tracked point might drift away over time.
- **Simple**: Uses the center of the detected bounding box as tracked reference.

An example of a bounding box of a detected *person* object.

Filter Types



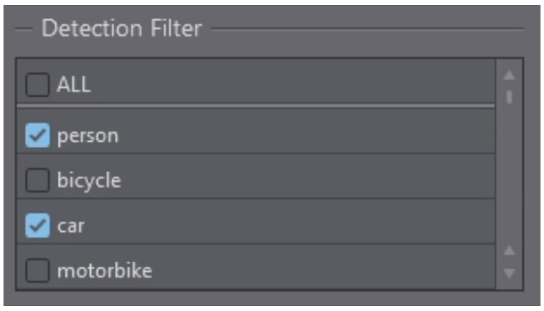
The filter types allow you to filter the tracked point over time. Several of them are available. The button next to the drop down sends the new settings to the Object Tracker without having to use the global initialize button.

- **MovingAverage**: Calculates the average of the last points.
- **ExponentialMovingAverage**: Gives more weight to the most recent points. Similar to MovingAverage.

- **KalmanSimple:** Simple Kalman filter.
- **KalmanLinear:** Linear Kalman filter.
- **KalmanUnscented:** Unscented Kalman filter.
- **KalmanAugmentedUnscented:** Augmented Unscented Kalman filter.

Warning: Be very careful when using filters. In most cases, they lead to undesired lags especially for rapid and sudden changes in direction of the tracked object. The larger the size of the filter, the larger is the lag. However, the filters might be very beneficial on smooth and slow movements of objects.

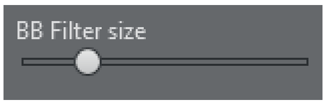
Detection Filter


Detection Filter

- [ ] ALL
- [x] person
- [ ] bicycle
- [x] car
- [ ] motorbike

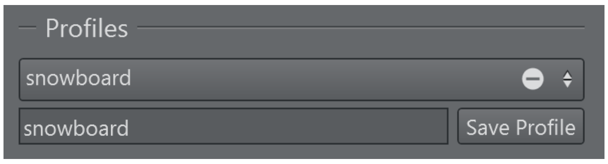
Select which objects of you want to detect. In most cases, you would like to select one or two kind of objects in a video stream.

Bounding Box Filter



The bounding boxes undergo a different filtering than the actual tracking point as it is more noisy and is subject to more extreme and sudden changes over time (for example, a person walking). You can configure a filter size for smoothing the bounding boxes for the tracking output. It is particularly useful when graphics need to be placed aligned to the bounding box.

7.1.3 Profiles

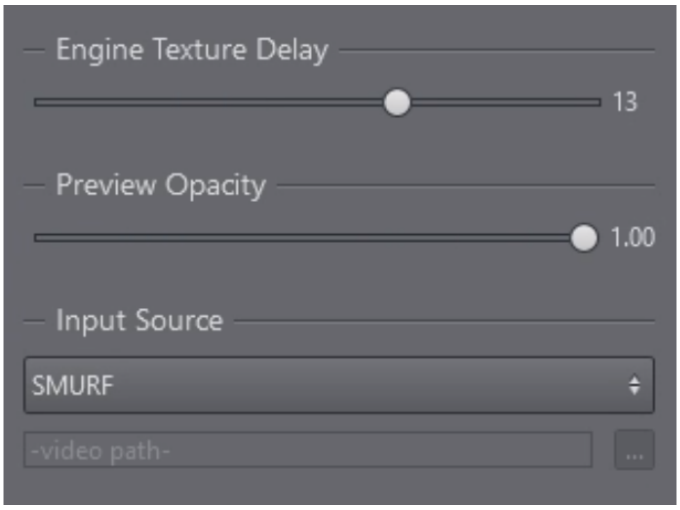

Profiles

snowboard

snowboard | Save Profile

Here you can save all the above settings into a profile which you can recall later on. Profiles are stored locally.

7.2 Global Settings



7.2.1 Engine Texture Delay

The Engine texture delay (in fields) determines the delay between the acquisition of the input texture for the Object Tracker and the actual rendering of the resulting tracking. This value depends heavily on the selected algorithm and underlying hardware. Typically the delay is around 11-13 fields.

Information: The delay is applied on the **program** channel that has been configured in Viz Arc. The delay value is set on the Engine when a object has been selected for tracking. Changing this value might lead to a freeze frame on Viz Engine's output, make sure not to change this value during live operations.

7.2.2 Preview Opacity

The preview output that has been configured in Viz Arc can be rendered on top of the NDI stream coming from the Object Tracker. This allows the operator to preview the graphics on screen. The opacity of this preview can be adjusted here.

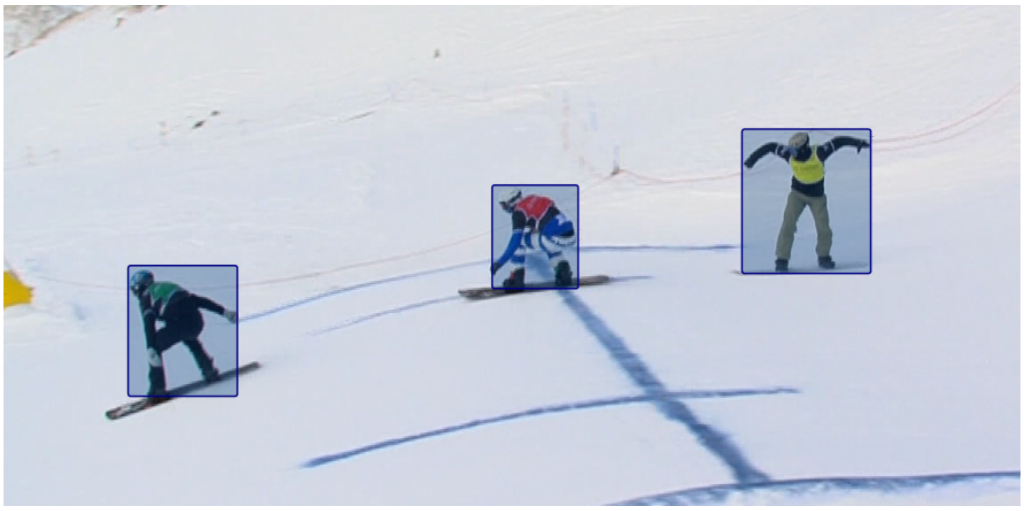
The preview is intended to verify the editorial graphics content and not the accuracy of the tracking.

Information: For performance reasons the composed on-screen preview might not look the same as on the actual program output. Make sure the fill channel of the preview is black where it is supposed to be transparent. The timing between the underlying NDI stream and preview is not guaranteed to be correct. The tracking and the preview graphics might not match.

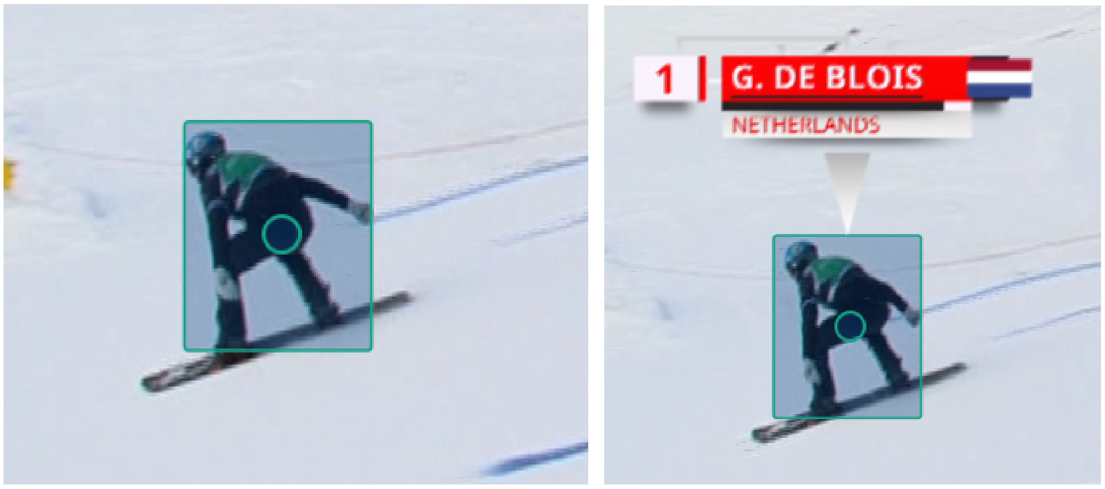
7.2.3 Input Source

Select **SMURF** to select the Engine's input running where the Object Tracker has been installed. When installing Viz Arc locally, you are able to select a file as well. This workflow is intended for testing only as it is not guaranteed to run in real-time.

8 Operating The Object Tracker



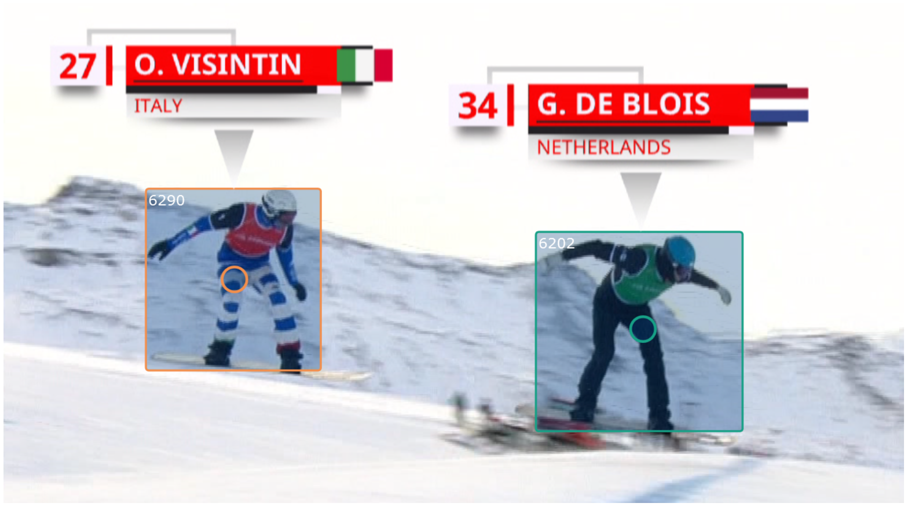
Select an active tracker and use the **left mouse button** and click on any of the blue bounding boxes around the tracked objects. Clicks don't have to be within the bounding box, the bounding box closest to the mouse click is selected.



Once selected, the bounding box changes to the color of the tracker and a circle appears in the center of the bounding box. The center of the circle is the actual point being tracked. Depending on how the **In Action** has been configured, the action is executed automatically or it needs to be triggered manually either through the **SPACEBAR** or **Take** button. It always executes on the preview cannel.

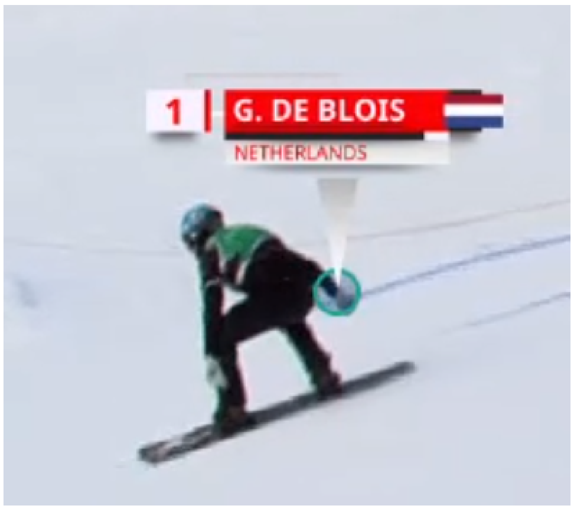
Right click on or close to the bounding box to put the graphics Off Air. Use the **BACKSPACE** or the **Take Out** button to take all graphics Off Air.

When available, select the next available tracker either through the **TAB** shortcut or by selecting the tracker from the top **Tracked Object** bar.



8.1 Manual Mode

It is possible to switch into manual mode at any time. Either through the keyboard shortcut **M** or by toggling the **M toggle** in the **Tracked Object** bar. Once in manual mode, hold down the left mouse button to start tracking and release it to lose tracking.



Manual mode might be used in very difficult visual conditions, on still frames or for graphics testing. It is not recommended to use it on fast moving objects.