



# Object Tracker User Guide

Version 1.3

# bject Tracker



**Copyright** ©2024 Vizrt. All rights reserved.

No part of this software, documentation or publication may be reproduced, transcribed, stored in a retrieval system, translated into any language, computer language, or transmitted in any form or by any means, electronically, mechanically, magnetically, optically, chemically, photocopied, manually, or otherwise, without prior written permission from Vizrt.

Vizrt specifically retains title to all Vizrt software. This software is supplied under a license agreement and may only be installed, used or copied in accordance to that agreement.

### **Disclaimer**

Vizrt provides this publication “as is” without warranty of any kind, either expressed or implied. This publication may contain technical inaccuracies or typographical errors. While every precaution has been taken in the preparation of this document to ensure that it contains accurate and up-to-date information, the publisher and author assume no responsibility for errors or omissions. Nor is any liability assumed for damages resulting from the use of the information contained in this document. Vizrt’s policy is one of continual development, so the content of this document is periodically subject to be modified without notice. These changes will be incorporated in new editions of the publication. Vizrt may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time.

Vizrt may have patents or pending patent applications covering subject matters in this document. The furnishing of this document does not give you any license to these patents.

### **Antivirus**

Vizrt does not recommend or test antivirus systems in combination with Vizrt products, as the use of such systems can potentially lead to performance losses. The decision for the use of antivirus software and thus the risk of impairments of the system is solely at the customer's own risk.

There are general best-practice solutions, these include setting the antivirus software to not scan the systems during operating hours and that the Vizrt components, as well as drives on which clips and data are stored, are excluded from their scans (as previously stated, these measures cannot be guaranteed).

### **Technical Support**

For technical support and the latest news of upgrades, documentation, and related products, visit the Vizrt web site at [www.vizrt.com](http://www.vizrt.com).

### **Created on**

2024/04/10

# Contents

<b>1</b>	<b>Introduction.....</b>	<b>5</b>
1.1	About Object Tracker.....	5
1.2	Related Documents.....	5
1.3	Feedback and Suggestions.....	5
1.4	System Overview.....	6
1.5	System Requirements .....	8
1.5.1	General.....	8
1.5.2	Vizrt Software .....	8
1.5.3	Minimum Hardware Requirements .....	8
1.5.4	Antivirus.....	9
<b>2</b>	<b>Installation.....</b>	<b>10</b>
2.1	Requirements .....	10
2.1.1	CUDA v12.1 .....	10
2.1.2	TensorRT v8.6.1.6 .....	11
2.1.3	CUDNN v8.9.0 .....	12
2.2	Installer .....	12
2.2.1	Icon Tray .....	15
2.3	First Power up.....	16
<b>3</b>	<b>Prerequisites.....</b>	<b>17</b>
<b>4</b>	<b>Viz Scene Design .....</b>	<b>19</b>
4.1	Live Video Input.....	19
4.2	Script.....	20
4.2.1	Tracking Script .....	20
4.2.2	Data Structures .....	29
<b>5</b>	<b>Preparing Operations .....</b>	<b>32</b>
5.1	Tracking Panel .....	33
5.1.1	In/out Actions.....	33
5.1.2	Take and Take Out .....	34
5.1.3	Tracked Object .....	34
5.2	Scripting Panel.....	35
<b>6</b>	<b>Configuration Panel .....</b>	<b>36</b>
6.1	Presets.....	37
6.2	Tracker Settings.....	37

6.2.1	Analysis .....	37
6.2.2	Delay Offset .....	43
6.3	Global Settings .....	43
6.3.1	Preview Opacity .....	43
6.3.2	Cut Detecion .....	44
6.3.3	Pointers Offset .....	44
6.3.4	Input Source .....	44
7	<b>Operating the Object Tracker .....</b>	<b>45</b>
7.1	Detection and Tracking, Face Tracking and 2D Pose Tracking .....	45
7.2	Simple Tracking .....	47
7.3	Manual Tracking .....	48
7.4	Offsets .....	48

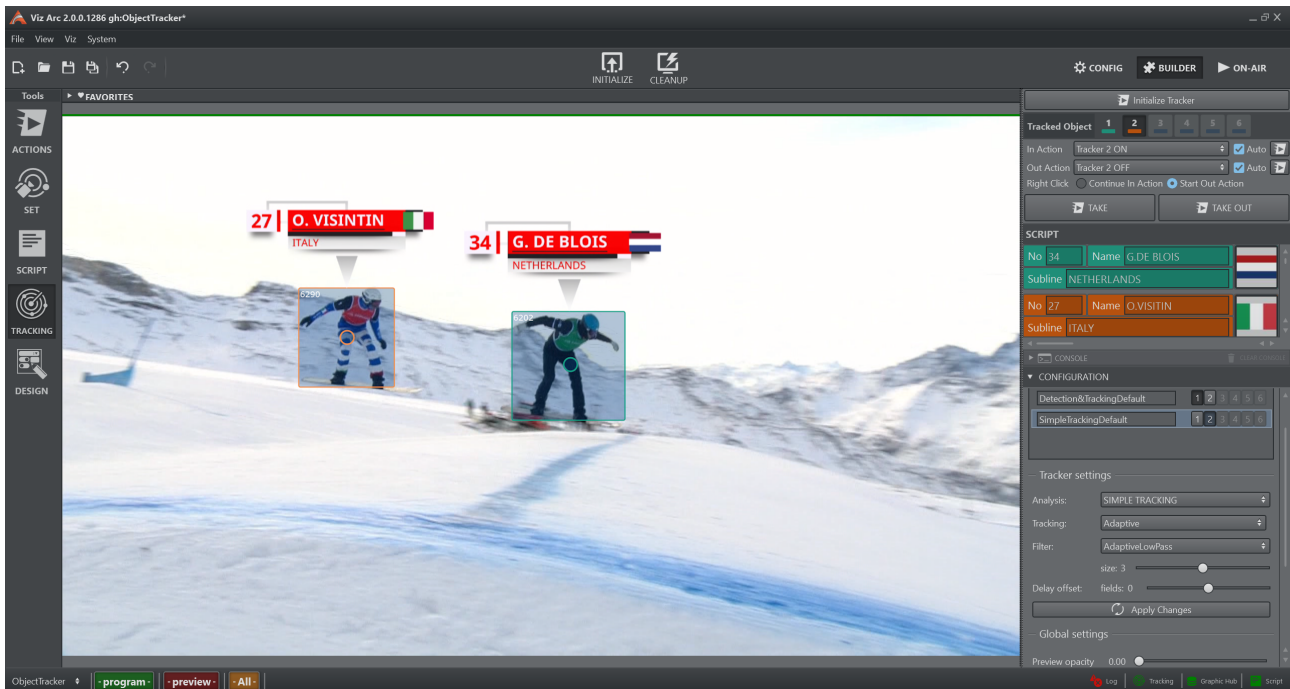
---

# 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.



---

## 1.2 Related Documents

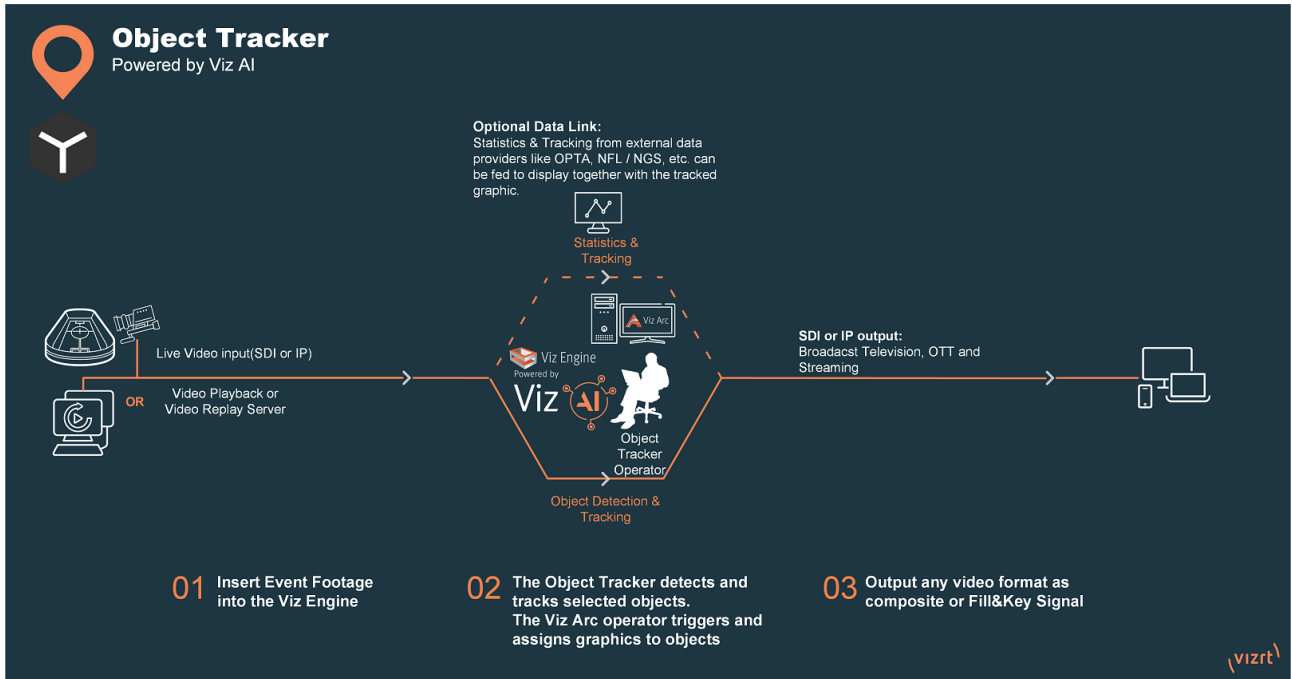
- [Viz Engine Administrator Guide](#)
- [Viz Arc User Guide](#)
- [Viz Arc Script Guide](#)

---

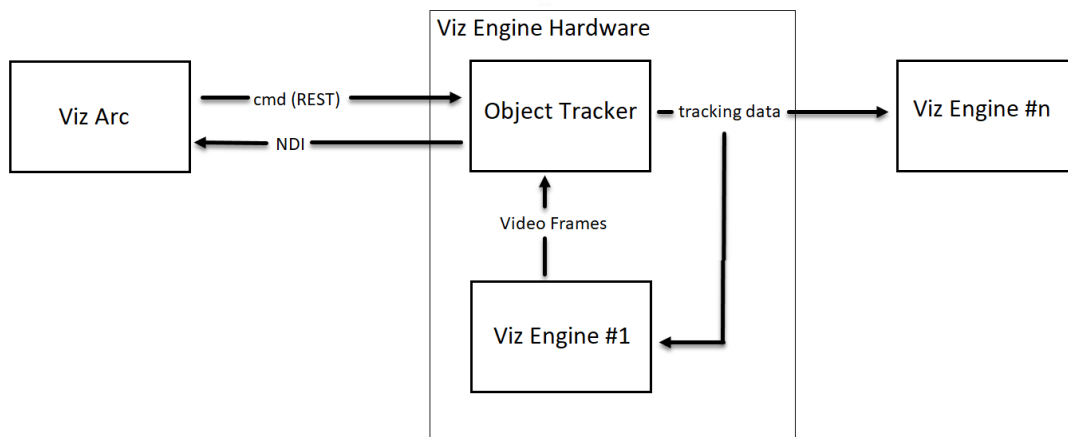
## 1.3 Feedback And Suggestions

We encourage feedback on our products and documentation. Please contact your local Vizrt customer support team at [www.vizrt.com](http://www.vizrt.com).

## 1.4 System Overview



Object Tracker allows tracking individual or multiple objects within a video to focus on the key parts of the action. This fully software-based and only requires video input.

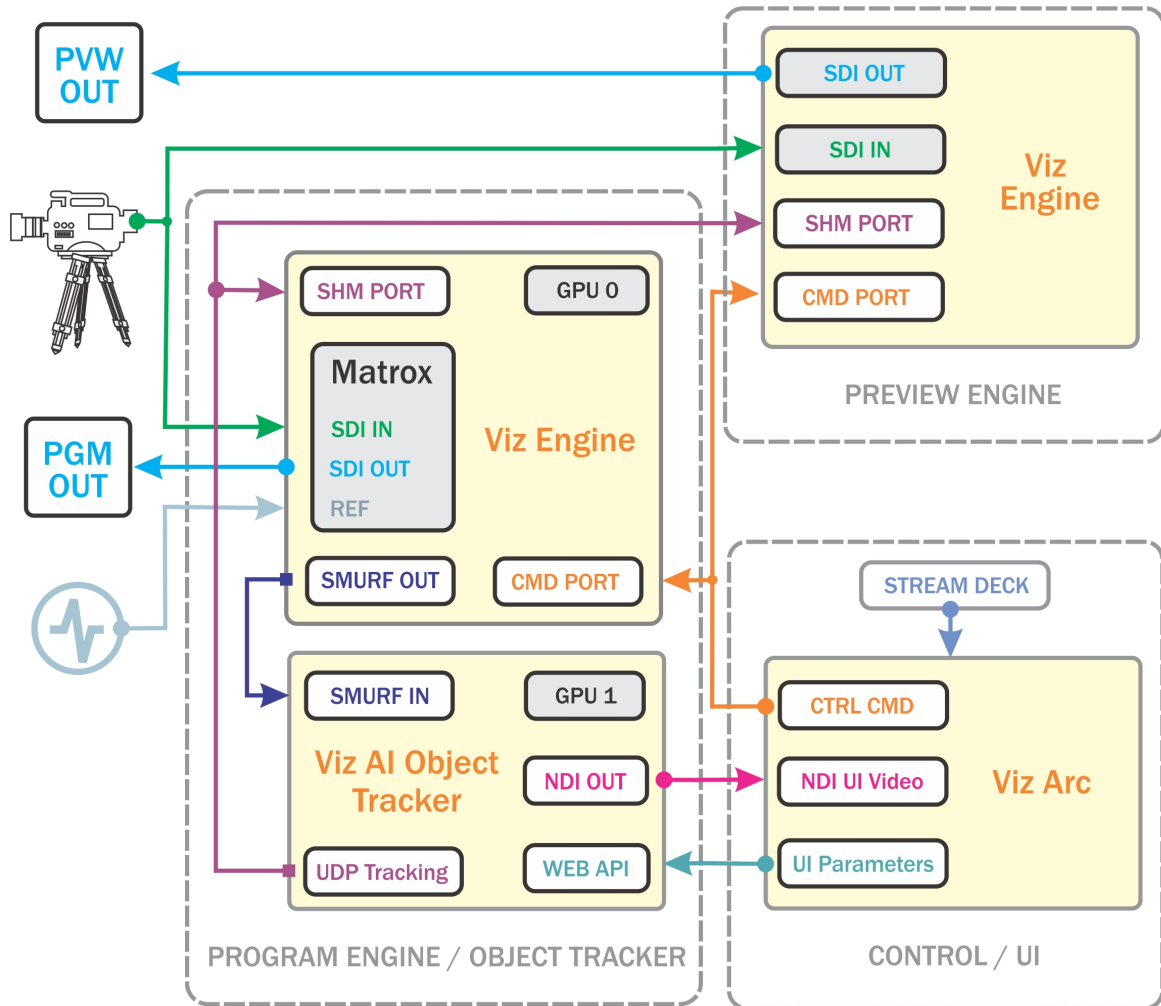


The overall system is composed of at least two machines, one running the Viz Engine and the Object Tracker and another running Viz Arc as the control application.

- **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.
- **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.

## Viz AI Object Tracker



---

## 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
- NVIDIA Cuda v12.1
- NVIDIA TensorRT v8.6.1.6
- NVIDIA CUDNN v8.9.0

### 1.5.2 Vizrt Software

- Viz Engine version 4.4.0 and later.
- Viz Arc version 2.0.0 and later.

### 1.5.3 Minimum Hardware Requirements

720p, 1080i, 1080p

#### Object Tracker

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

#### Viz Arc

- HP Zbook Mobile Workstation

**OR**

- 1x HP Z4
- 1x RTX 2000

 **Information:** Viz Arc must run on 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.



## 1.5.4 Antivirus

Vizrt does not recommend or test antivirus systems in combination with Vizrt products, as the use of such systems can potentially lead to performance losses. The decision for the use of antivirus software and thus the risk of impairments of the system is solely at the customer's own risk.

There are general best-practice solutions, these include setting the antivirus software to not scan the systems during operating hours and that the Vizrt components, as well as drives on which clips and data are stored, are excluded from their scans (as previously stated, these measures cannot be guaranteed).

Services and folders which should be excluded from scanning are:

- All files and executables coming from *C:\Program Files\Vizrt\Viz Object Tracker*.
- Make exception for *VizObjectTracker.exe*.

---

## 2 Installation

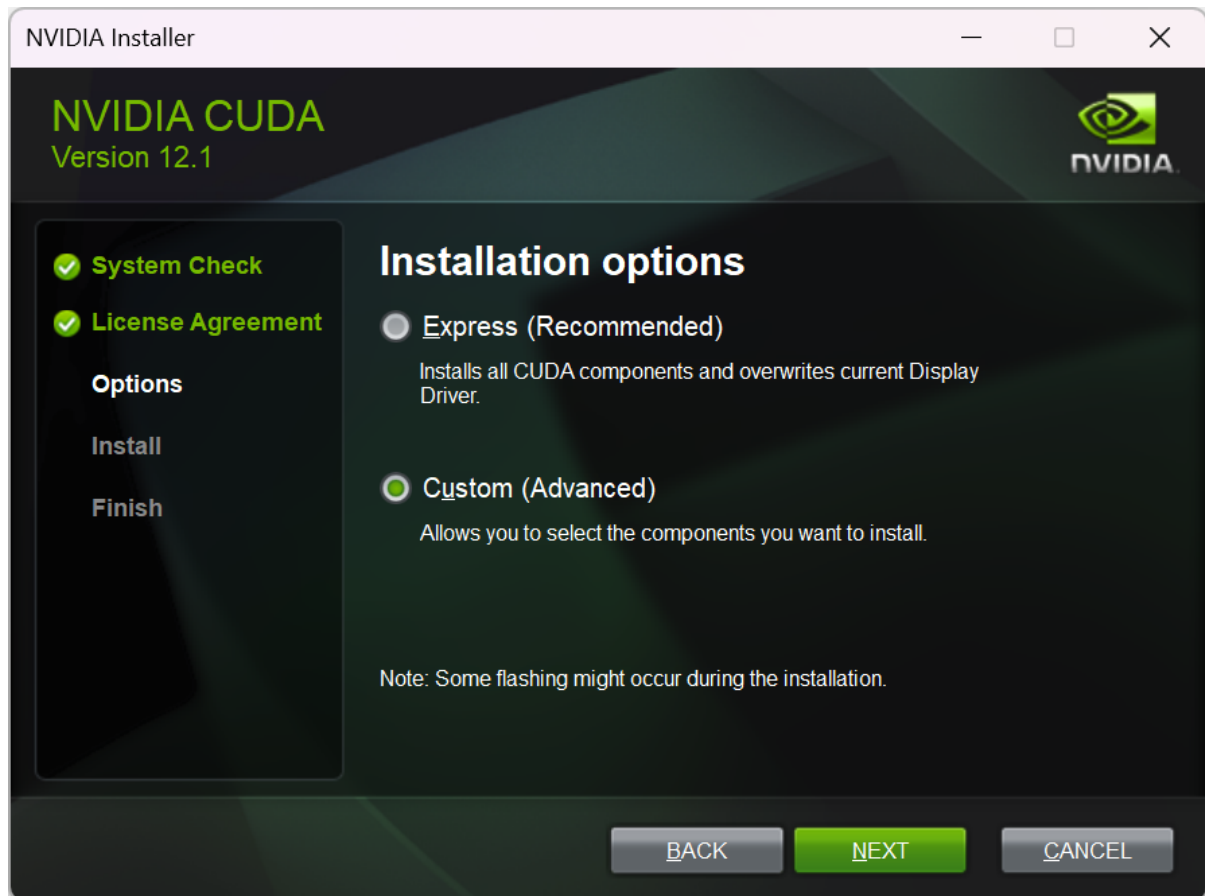
---

### 2.1 Requirements

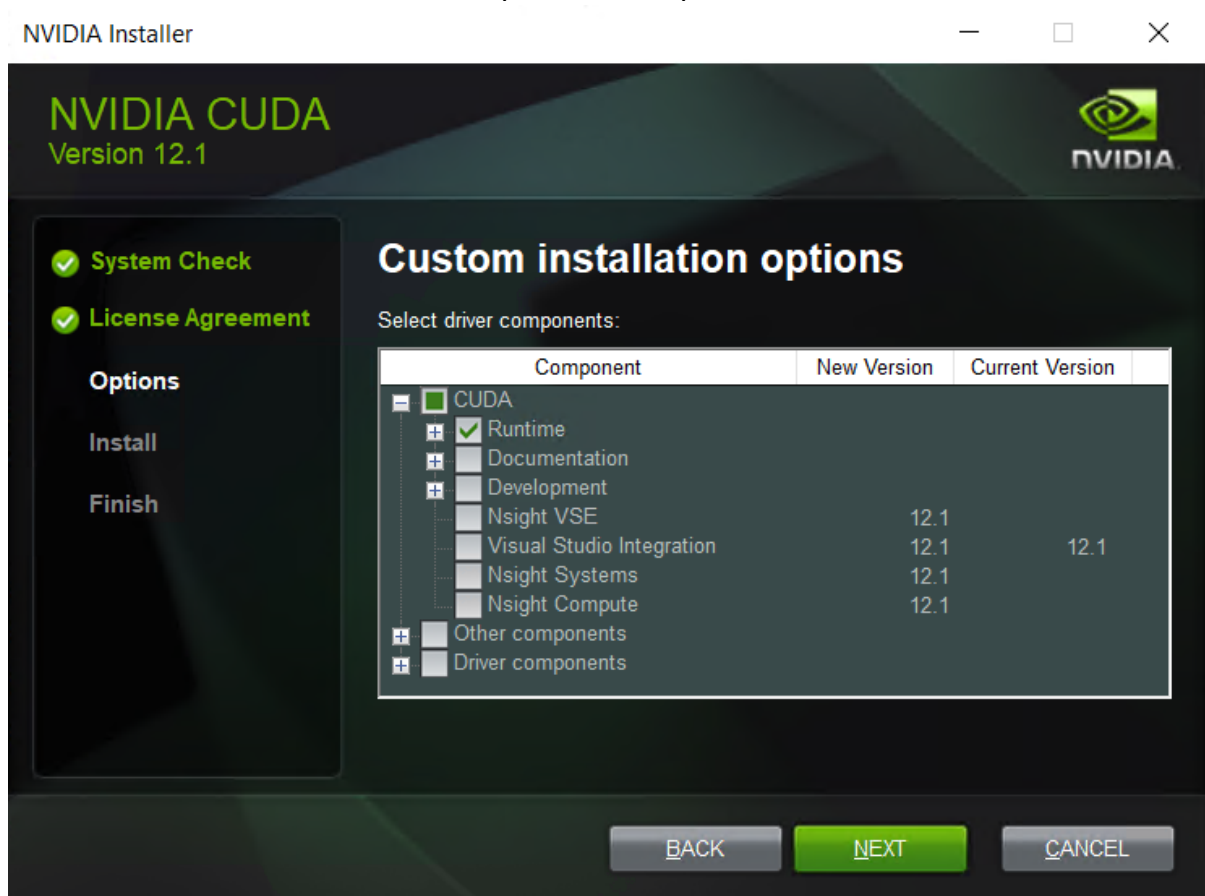
#### 2.1.1 CUDA v12.1

[Download](#) and install CUDA 12.1.

- Select Custom installation.



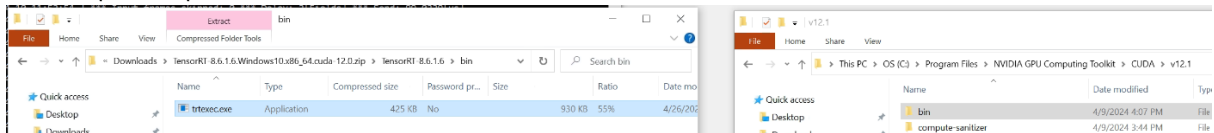
- Choose the custom installation where you select only the *Runtime*.



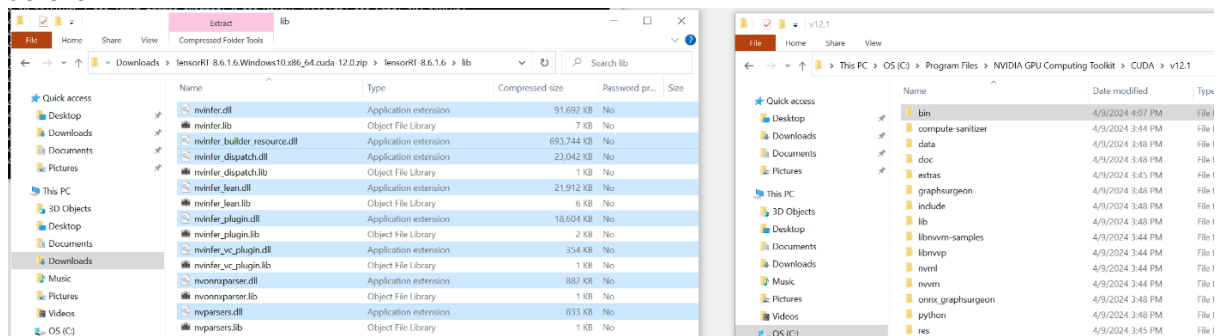
## 2.1.2 TensorRT v8.6.1.6

Download and install TensorRT 8.6.1.6.

- Unzip the file.
- From the unzipped file, copy the `\bin` folder into the `\bin` directory inside the CUDA 12.1 installation folder: Normally located at `C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v12.1`.



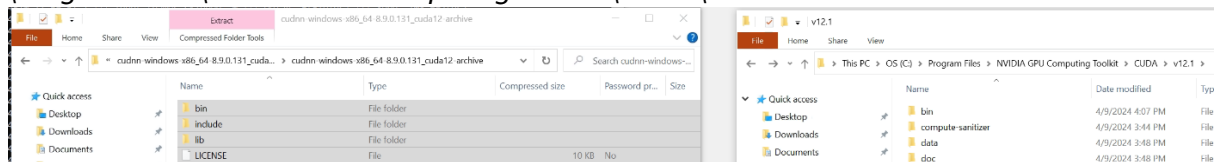
- From the unzipped file, copy the dlls inside the folder `\lib` into the same `\bin` directory as before.



## 2.1.3 CUDNN v8.9.0

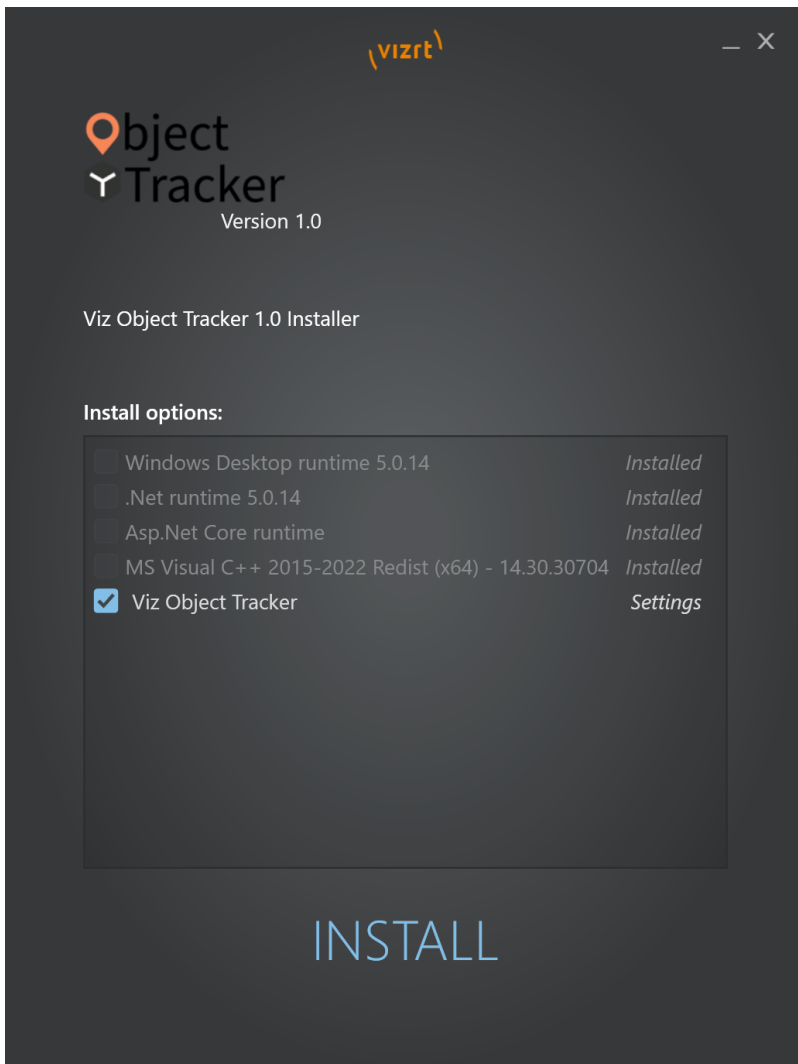
Download and install CUDNN 8.9.0

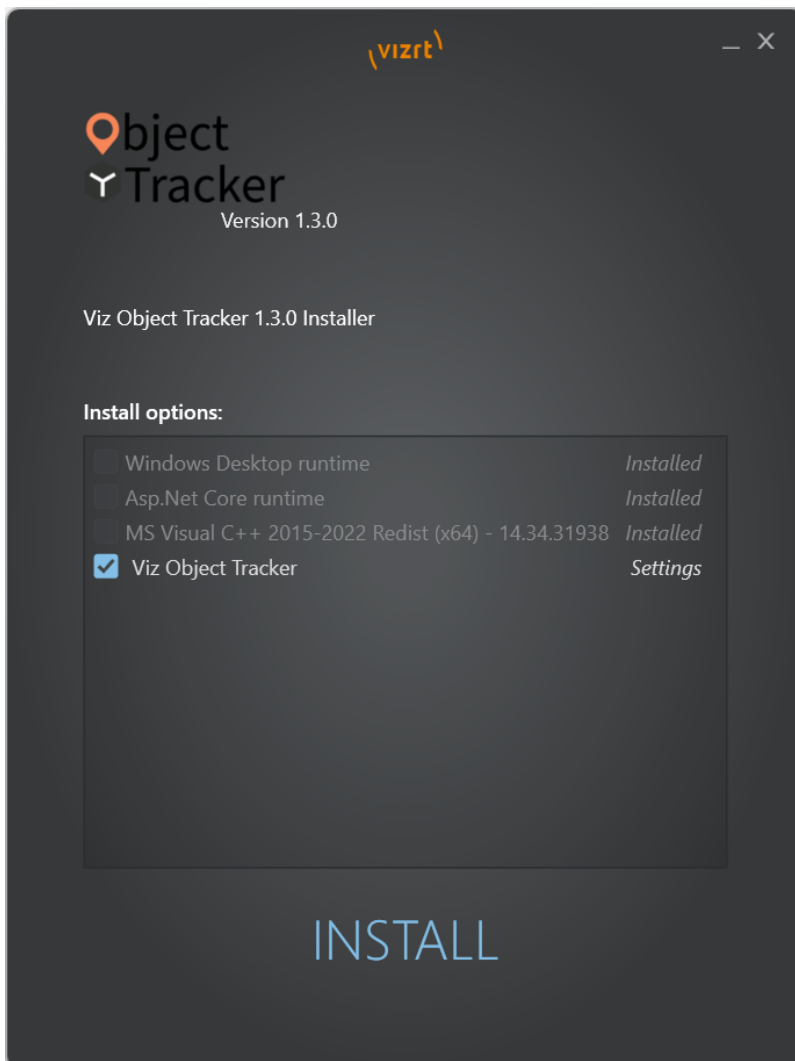
- Unzip the file.
- Copy the unzipped folders inside the CUDA 12.1 installation folder: Normally located at `C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v12.1`.



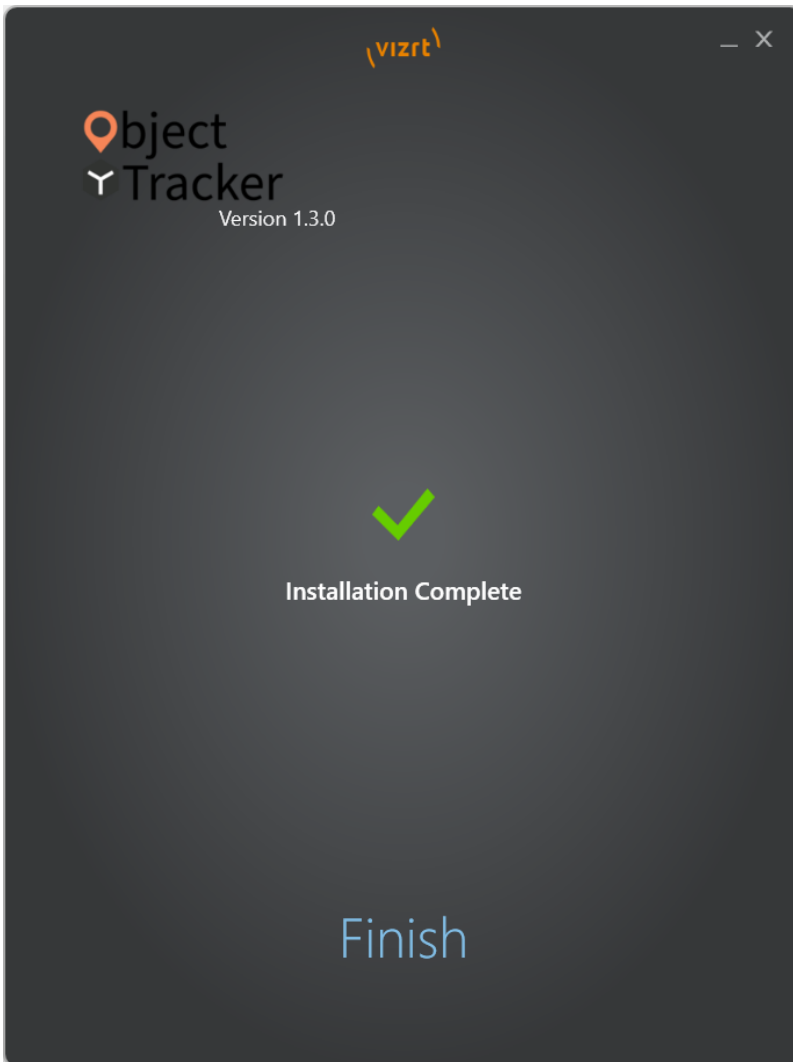
## 2.2 Installer

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




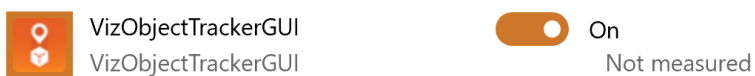


Select the **Object Tracker** for installation and all required prerequisites if not already installed.



Press **Finish** to complete installation.

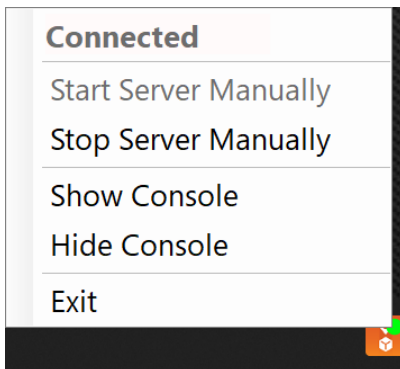
Once the Installation has finished the Object Tracker is started automatically and shows up with a green light in the icon tray .



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

### 2.2.1 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.

## 2.3 First Power Up

Once everything is installed properly, before starting the Object Tracker, the generic dataset model has to be compiled for every CUDA graphic cards located in the workstation. This process takes few minutes for each graphic card.

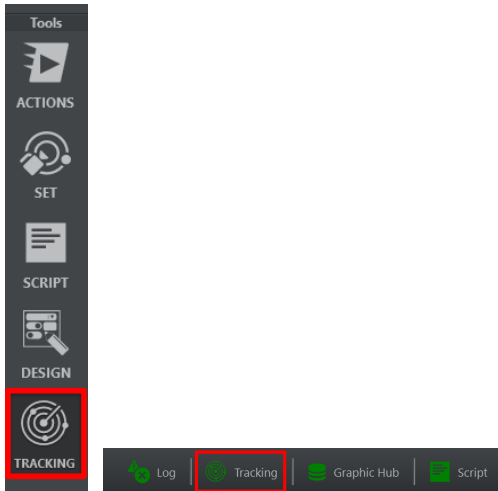
```

D:\CODE\netventure\gargan x + v
&&&& RUNNING TensorRT.trtexec [TensorRT v8601] # --onnx=C:\ProgramData\vizrt\VizObjectTracker\dnn\yolo-generic.onnx --save
Engine=C:\ProgramData\vizrt\VizObjectTracker\dnn\yolo-generic.engine0 --fp16 --device=0
[04/10/2024-11:57:20] [I] === Model Options ===
[04/10/2024-11:57:20] [I] Format: ONNX
[04/10/2024-11:57:20] [I] Model: C:\ProgramData\vizrt\VizObjectTracker\dnn\yolo-generic.onnx
[04/10/2024-11:57:20] [I] Output:
[04/10/2024-11:57:20] [I] === Build Options ===
[04/10/2024-11:57:20] [I] Max batch: explicit batch
[04/10/2024-11:57:20] [I] Memory Pools: workspace: default, dlaSRAM: default, dlaLocalDRAM: default, dlaGlobalDRAM: default
[04/10/2024-11:57:20] [I] minTiming: 1
[04/10/2024-11:57:20] [I] avgTiming: 8
[04/10/2024-11:57:20] [I] Precision: FP32+FP16
[04/10/2024-11:57:20] [I] LayerPrecisions:
[04/10/2024-11:57:20] [I] Layer Device Types:
[04/10/2024-11:57:20] [I] Calibration:
[04/10/2024-11:57:20] [I] Refit: Disabled
[04/10/2024-11:57:20] [I] Version Compatible: Disabled
[04/10/2024-11:57:20] [I] TensorRT runtime: full
[04/10/2024-11:57:20] [I] Lean DLL Path:
[04/10/2024-11:57:20] [I] Tempfile Controls: { in_memory: allow, temporary: allow }
[04/10/2024-11:57:20] [I] Exclude Lean Runtime: Disabled
[04/10/2024-11:57:20] [I] Sparsity: Disabled
[04/10/2024-11:57:20] [I] Safe mode: Disabled
[04/10/2024-11:57:20] [I] Build DLA standalone loadable: Disabled
[04/10/2024-11:57:20] [I] Allow GPU fallback for DLA: Disabled
[04/10/2024-11:57:20] [I] DirectIO mode: Disabled
[04/10/2024-11:57:20] [I] Restricted mode: Disabled
[04/10/2024-11:57:20] [I] Skip inference: Disabled
[04/10/2024-11:57:20] [I] Save engine: C:\ProgramData\vizrt\VizObjectTracker\dnn\yolo-generic.engine0
  
```

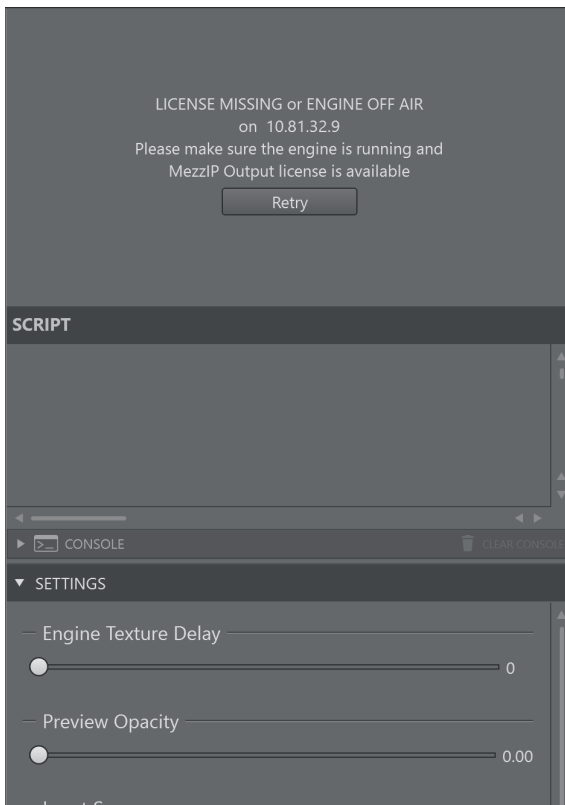


### 3 Prerequisites

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 <version>/Resources/ObjectTracker*).
- Viz Engine needs to run on on Matrox based hardware.
- For **Viz Arc** version **1.8.1** or older, make sure you have the **MezzIP Output license** on the program Viz Engine where the **Object Tracker** is running.
- For **Viz Arc** version **1.9.0** or newer, make sure you have the **Object Tracker license** on the program Viz Engine where the **Object Tracker** is running.

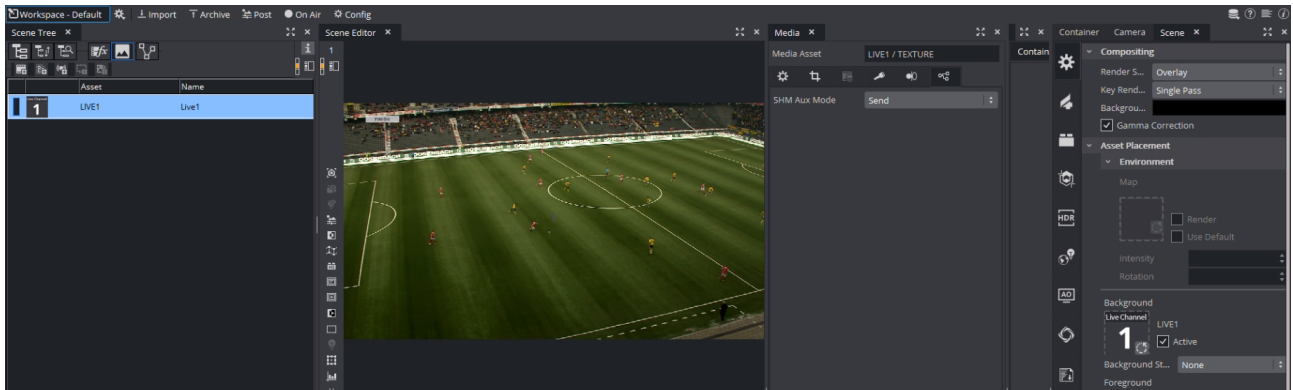


In case the license is missing or Viz Engine is offline, an error message appears as shown above.

## 4 Viz Scene Design

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

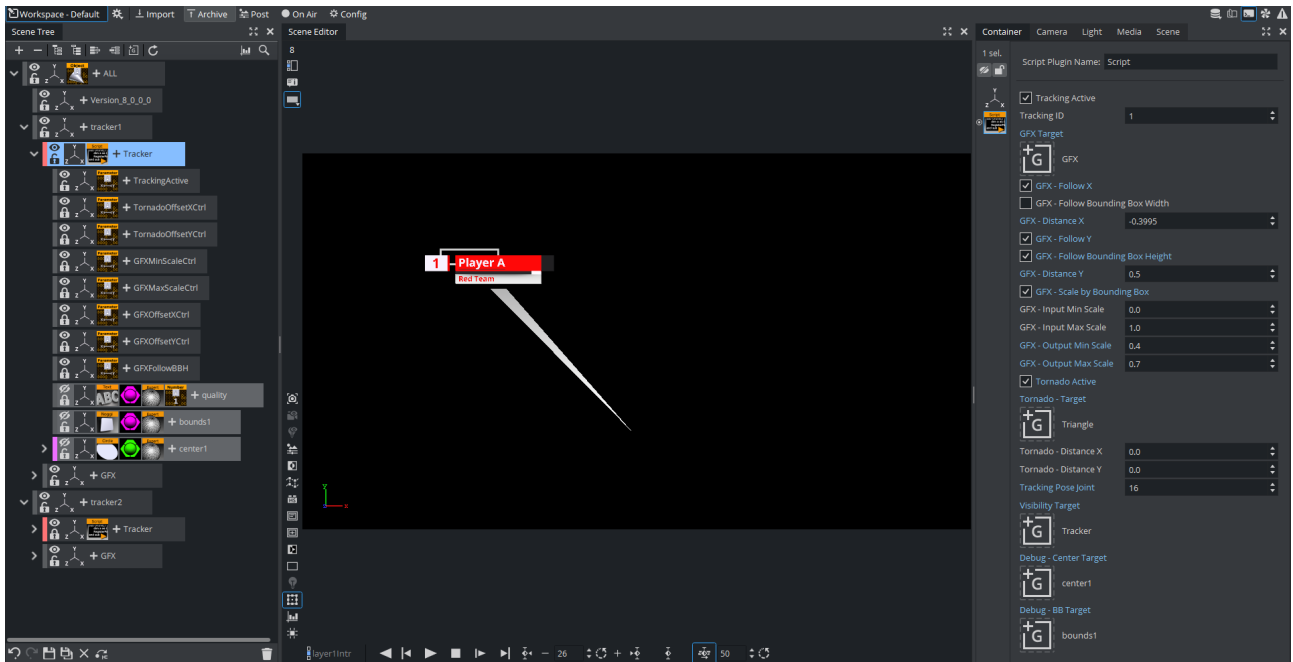
### 4.1 Live Video Input



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.).

## 4.2 Script



### 4.2.1 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 the following script:

```

Dim trackerID = 0
Dim screenWidth as Double
Dim screenHeight as Double
Dim mpi = 3.14159265359
Dim degToRad = mpi / 180.0
Dim currentTrackingType as String

Structure tracked
  x as Double
  y as Double
  width as Double
  height as Double
  valid as Integer
  score as Double
End Structure

Structure trackedObjects
  'TIMECODE
  tc as Integer
  'TRACKED OBJECTS
  tos as Array[tracked]

```

**End Structure**

**Structure** Point3D

x as **Double**

y as **Double**

z as **Double**

**End Structure**

**Structure** face

valid as **Integer**

Outline as Array[Point3D]

EyeLeft as Array[Point3D]

EyeRight as Array[Point3D]

Mouth as Array[Point3D]

BetweenEyes as Point3D

EyeCornerLeft as Point3D

EyeCornerRight as Point3D

PupilLeft as Point3D

PupilRight as Point3D

NoseTip as Point3D

CheekLeft as Point3D

CheekRight as Point3D

MouthCornerLeft as Point3D

MouthCornerRight as Point3D

Chin as Point3D

Roll as **Double**

Pitch as **Double**

Yaw as **Double**

**End Structure**

**Structure** faceObjects

'TIMECODE

tc as **Integer**

'FACE OBJECTS

fs as Array[face]

**End Structure**

**Structure** pose

valid as **Integer**

skeleton as Array[Point3D]

**End Structure**

**Structure** poseObjects

'TIMECODE

tc as **Integer**

'POSE OBJECTS

ps as Array[pose]

**End Structure**

**Sub** OnInit()

'do some trigonometry to get the screen width/height on the 0 plane.

'this calculation works in orto and perspectiv 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.CurrentCamera.Position.z*2
screenHeight =
tan(scene.CurrentCamera.Fovy*0.5*degToRad)*scene.CurrentCamera.Position.z*2

VizCommunication.map.RegisterChangedCallback("FaceObjects")
VizCommunication.map.RegisterChangedCallback("PoseObjects")
VizCommunication.map.RegisterChangedCallback("TrackedObjects")
End Sub

Dim somethingToShow as Boolean
Dim pos as Vertex
Dim rot as Vertex
Dim scale as Double
Dim bbHeight as Double
Dim bbWidth as Double
Dim quality as Double

Sub Move()
  Dim bbTarget      = GetParameterContainer("bbTarget")
  Dim centerTarget = GetParameterContainer("centerTarget")
  Dim gfxTarget     = GetParameterContainer("gfxTarget")

  centerTarget.position.x = pos.x
  centerTarget.position.y = pos.y

  centerTarget.Rotation.x = rot.x
  centerTarget.Rotation.y = rot.y
  centerTarget.Rotation.z = rot.z

  centerTarget.Scaling.x = scale
  centerTarget.Scaling.y = scale
  centerTarget.Scaling.z = scale

  bbTarget.position.x = pos.x
  bbTarget.position.y = pos.y
  bbTarget.Scaling.x  = bbWidth / 100
  bbTarget.Scaling.y  = bbHeight / 100

  if GetParameterBool("gfxFollowX") == true Then
    if(GetParameterBool("gfxFollowBBX") == true) then
      gfxTarget.position.X = pos.X + (bbWidth + GetParameterDouble("gfxDeltaX")
* screenWidth) * 0.5
    else
      gfxTarget.position.X = pos.X + GetParameterDouble("gfxDeltaX") * 0.5 *
screenWidth
    end if
  end If

  if GetParameterBool("gfxFollowY") == true Then
    If(GetParameterBool("gfxFollowBBY") == true) then

```

```

        gfxTarget.position.Y = pos.Y + (bbHeight + GetParameterDouble("gfxDeltaY")
* screenHeight) * 0.5
    Else
        gfxTarget.position.Y = pos.Y + GetParameterDouble("gfxDeltaY") * 0.5 *
screenHeight
    End if
End If

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

    inMin = GetParameterDouble("gfxInMinScale")
    inMax = GetParameterDouble("gfxInMaxScale")

    outMin = GetParameterDouble("gfxOutMinScale")
    outMax = GetParameterDouble("gfxOutMaxScale")

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

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

    gfxTarget.scaling.x = gfxScale
    gfxTarget.scaling.y = gfxScale
    gfxTarget.scaling.z = gfxScale
End If

Dim tornadoActive as Boolean
tornadoActive = GetParameterBool("tornadoActive")
If tornadoActive Then
    Dim tornadoTarget = GetParameterContainer("tornadoTarget")
    Dim loc = tornadoTarget.WorldPosToLocalPos(pos)

    tornadoTarget.GetGeometryPluginInstance().SetParameterDouble("x3", loc.x +
GetParameterDouble("tornadoDeltaX") * bbWidth / gfxTarget.scaling.x)
    tornadoTarget.GetGeometryPluginInstance().SetParameterDouble("y3", loc.y +
GetParameterDouble("tornadoDeltaY") * bbHeight / gfxTarget.scaling.y)
End If
End Sub

Sub FaceTracking()
    Dim fObjects as faceObjects
    fObjects = (faceObjects)VizCommunication.Map["FaceObjects"]

    If fObjects.fs.Size > trackerID Then
        somethingToShow = True
    Else
        somethingToShow = False
    End If
End Sub

```

**End If**

**If** somethingToShow **Then**

    somethingToShow = fObjects.fs[trackerID].valid == 1

**End If**

**If** somethingToShow **Then**

    rot.x = -fObjects.fs[trackerID].Pitch / degToRad

    rot.y = fObjects.fs[trackerID].Yaw / degToRad

    rot.z = -fObjects.fs[trackerID].Roll / degToRad

    pos.x = fObjects.fs[trackerID].NoseTip.x \* screenWidth

    pos.y = -fObjects.fs[trackerID].NoseTip.y \* screenHeight

    dim cheekLeft as Vertex

    dim cheekRight as Vertex

    cheekLeft.x = fObjects.fs[trackerID].CheekLeft.x \* screenWidth

    cheekLeft.y = fObjects.fs[trackerID].CheekLeft.y \* screenHeight

    cheekLeft.z = fObjects.fs[trackerID].CheekLeft.z \* screenWidth

    cheekRight.x = fObjects.fs[trackerID].CheekRight.x \* screenWidth

    cheekRight.y = fObjects.fs[trackerID].CheekRight.y \* screenHeight

    cheekRight.z = fObjects.fs[trackerID].CheekRight.z \* screenWidth

    dim cheek\_distance as **Double**

    cheek\_distance = Distance(cheekLeft, cheekRight)

    cheek\_distance /= 100

    scale = cheek\_distance

    Move()

**End If**

**End Sub**

**Sub** ObjectTracking()

**Dim** trObjects as trackedObjects

    trObjects = (trackedObjects)VizCommunication.Map["TrackedObjects"]

    somethingToShow = trObjects.tos.Size > trackerID

**If** (somethingToShow) **Then**

        somethingToShow = trObjects.tos[trackerID].valid == 1

**End If**

**If** (somethingToShow) **Then**

        pos.x = trObjects.tos[trackerID].x \* screenWidth

        pos.y = -trObjects.tos[trackerID].y \* screenHeight

        bbWidth = trObjects.tos[trackerID].width \* screenWidth

        bbHeight = trObjects.tos[trackerID].height \* screenHeight

        rot.x = 0.0

        rot.y = 0.0

        rot.z = 0.0

        scale = 1.0



```

    Move()

    quality = trObjects.tos[trackerID].score
    findSubContainer("quality").GetFunctionPluginInstance("ControlNum").SetParameterString("input", cStr(quality*100.0))
  End If
End Sub

Sub PoseTracking()
  Dim jointIdx = GetParameterInt("trackingJoint")
  Dim pObjects as poseObjects
  pObjects = (poseObjects)VizCommunication.Map["PoseObjects"]

  somethingToShow = pObjects.ps.Size > trackerID
  If (somethingToShow) Then
    somethingToShow = pObjects.ps[trackerID].valid == 1
  End If

  If (somethingToShow) Then
    pos.x = pObjects.ps[trackerID].Skeleton[jointIdx].x * screenWidth
    pos.y = -pObjects.ps[trackerID].Skeleton[jointIdx].y * screenHeight
    'pos.z = pObjects.ps[trackerID].Skeleton[jointIdx].z

    rot.x = 0.0
    rot.y = 0.0
    rot.z = 0.0
    scale = 1.0
    bbHeight = 0
    bbWidth = 0

    Move()
  End If
End Sub

Sub OnExecPerField()
  trackerID = GetParameterInt("trackingID") - 1

  Dim trackingActive as Boolean
  trackingActive = GetParameterBool("trackingActive")

  If(trackingActive) Then
    If( currentTrackingType == "FaceObjects") Then
      FaceTracking()
    ElseIf( currentTrackingType == "TrackedObjects") Then
      ObjectTracking()
    ElseIf( currentTrackingType == "PoseObjects") Then
      PoseTracking()
    End If
  End If

  Dim visibilityTarget = GetParameterContainer("visibilityTarget")

```

```
visibilityTarget.Active = trackingActive AND somethingToShow
this.Active = trackingActive AND somethingToShow
```

**End Sub**

**Sub** OnInitParameters()

```
RegisterParameterBool("trackingActive", "Tracking Active", True)
RegisterParameterInt("trackingID", "Tracking ID", 1, 1, 5)
```

```
RegisterParameterContainer("gfxTarget", "GFX Target")
RegisterParameterBool("gfxFollowX", "GFX - Follow X", False)
RegisterParameterBool("gfxFollowBBX", "GFX - Follow Bounding Box Width", False)
RegisterParameterDouble("gfxDeltaX", "GFX - Distance X", 0.0, -1000000, 1000000)
RegisterParameterBool("gfxFollowY", "GFX - Follow Y", False)
RegisterParameterBool("gfxFollowBBY", "GFX - Follow Bounding Box Height", False)
RegisterParameterDouble("gfxDeltaY", "GFX - Distance Y", 0.0, -1000000, 1000000)
```

```
RegisterParameterBool("gfxDoScale", "GFX - Scale by Bounding Box", False)
```

```
RegisterParameterDouble("gfxInMinScale", "GFX - Input Min Scale", 0.0, 0, 1.0)
RegisterParameterDouble("gfxInMaxScale", "GFX - Input Max Scale", 1.0, 0, 1.0)
```

```
RegisterParameterDouble("gfxOutMinScale", "GFX - Output Min Scale", 0.0, 0,
1000000)
```

```
RegisterParameterDouble("gfxOutMaxScale", "GFX - Output Max Scale", 1.0, 0,
1000000)
```

```
RegisterParameterBool("tornadoActive", "Tornado Active", False)
```

```
RegisterParameterContainer("tornadoTarget", "Tornado - Target")
```

```
RegisterParameterDouble("tornadoDeltaX", "Tornado - Distance X", 0.0, -1000000,
1000000)
```

```
RegisterParameterDouble("tornadoDeltaY", "Tornado - Distance Y", 0.0, -1000000,
1000000)
```

```
RegisterParameterInt("trackingJoint", "Tracking Pose Joint", 0, 0, 32)
```

```
RegisterParameterContainer("visibilityTarget", "Visibility Target")
```

```
RegisterParameterContainer("centerTarget", "Debug - Center Target")
```

```
RegisterParameterContainer("bbTarget", "Debug - BB Target")
```

**End Sub**

**Sub** OnSharedMemoryVariableChanged(map As SharedMemory, mapKey As String)

```
currentTrackingType = mapKey
```

**End Sub**

Tracking Active

Tracking ID

GFX Target

 GFX

GFX - Follow X

GFX - Follow Bounding Box Width

GFX - Distance X

GFX - Follow Y

GFX - Follow Bounding Box Height

GFX - Distance Y

GFX - Scale by Bounding Box

GFX - Input Min Scale

GFX - Input Max Scale

GFX - Output Min Scale

GFX - Output Max Scale

Tornado Active

Tornado - Target

 Triangle

Tornado - Distance X

Tornado - Distance Y

Tracking Pose Joint

Visibility Target

 Tracker

Debug - Center Target

 center1

Debug - BB Target

 bounds1

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 keys.

The **TrackingID** parameter selects the tracked point to follow in the tracking points data structures (it represents also the Tracked Object Input ID in the Viz Arc interface).

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

- **GFX - Follow X:** Allows the graphics follow the horizontal movement of the tracked object.
- **GFX - Follow Bounding Box Width:** Positions the graphics on the right hand side of the bounding box.
- **GFX - Distance X:** Adds constant horizontal offset.
- **GFX - Follow Y:** Allows the graphics follow the vertical movement of the tracked object.
- **GFX - Follow Bounding Box Height:** Aligns the graphics on top of the bounding box.
- **GFX - Distance Y:** Adds constant vertical offset.
- **GFX - Scale by Bounding Box:** Uses Bounding Box size to scale graphics.
- **GFX - 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.).
- **GFX - 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.).
- **GFX - Output Min Scale:** Maps minimum output scale of the graphics.
- **GFX - 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.

The checkbox **Tornado Active** enables both the visualization and the evaluation of the position of the **Tornado - Target**.

It can be customized through:

- **Tornado - Distance X:** The horizontal offset from the tracked point.
- **Tornado - Distance Y:** The vertical offset from the tracked point.

The **Tracking Pose Joint** selects the landmark to be used as tracked point.

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 **Debug - Center Target** is updated on every field to the "check" the actual tracking position.

**i 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 **Debug - BB Target** container parameter (which might contain a Noggi or Rectangle plug-in) gets resized according to the tracked width and height of the object.

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

## 4.2.2 Data Structures

### Detection and Tracking, Simple Tracking and Manual Tracking

Each frame the Object Tracker updates the VizCommunicationMap variable with key *TrackedObjects*. It contains the structure shown below:

```
Structure tracked
  x as Double
  y as Double
  width as Double
  height as Double
  valid as Integer
  score as Double
End Structure

Structure trackedObjects
  'TIMECODE
  tc as Integer
  'TRACKED OBJECTS
  tos as Array[tracked]
End Structure
```

The *trackedObjects* structure contains an array of five *tracked* structures (the maximum number of objects that can be tracked); the parameter *valid* shows whether the object is actually tracked. The position inside the array matches the InputID used when the object was selected.

### Face Tracking

Each frame the Object Tracker updates the VizCommunicationMap variable with key *FaceObjects*. It contains the structure shown below:

```
Structure Point3D
  x as Double
  y as Double
  z as Double
End Structure
```

```

Structure face
  valid as Integer
  Outline as Array[Point3D]
  EyeLeft as Array[Point3D]
  EyeRight as Array[Point3D]
  Mouth as Array[Point3D]
  BetweenEyes as Point3D
  EyeCornerLeft as Point3D
  EyeCornerRight as Point3D
  PupilLeft as Point3D
  PupilRight as Point3D
  NoseTip as Point3D
  CheekLeft as Point3D
  CheekRight as Point3D
  MouthCornerLeft as Point3D
  MouthCornerRight as Point3D
  Chin as Point3D
  Roll as Double
  Pitch as Double
  Yaw as Double
End Structure

Structure faceObjects
  'TIMECODE
  tc as Integer
  'FACE OBJECTS
  fs as Array[face]
End Structure

```

The *faceObjects* structure contains an array of five *face* structures (the maximum number of objects that can be tracked); the parameter *valid* shows whether the face is actually tracked. The position inside the array matches the InputID used when the face was selected.

## 2D Pose Tracking

Each frame the ObjectTracker updates the VizCommunicationMap variable with key *PoseObjects*. It contains the structure shown below:

```

Structure Point3D
  x as Double
  y as Double
  z as Double
End Structure

Structure pose
  valid as Integer
  skeleton as Array[Point3D]
End Structure

Structure poseObjects

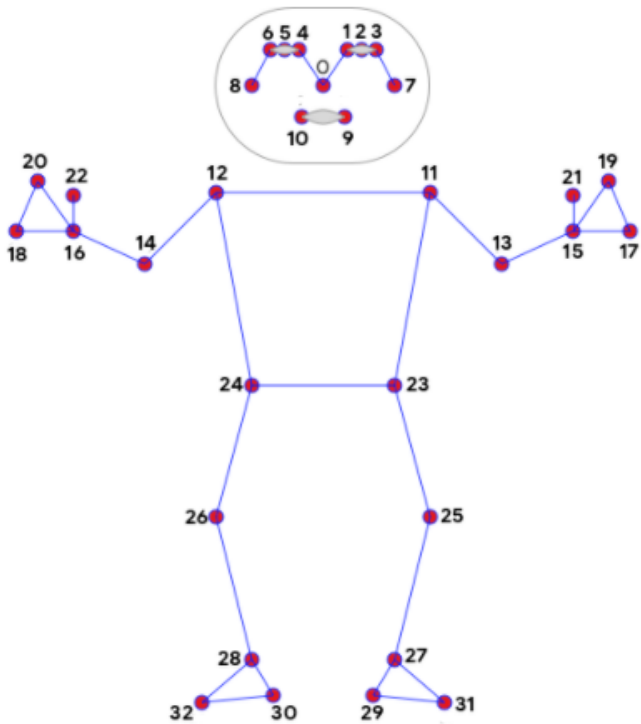
```

```

'TIMECODE
tc as Integer
'POSE OBJECTS
ps as Array[pose]
End Structure

```

The *poseObjects* structure contains an array of five *pose* structures (the maximum number of objects that can be tracked); the parameter *valid* shows whether the object is actually tracked. The position inside the array matches the InputID used when the person was selected. The *skeleton* array is filled as follows:

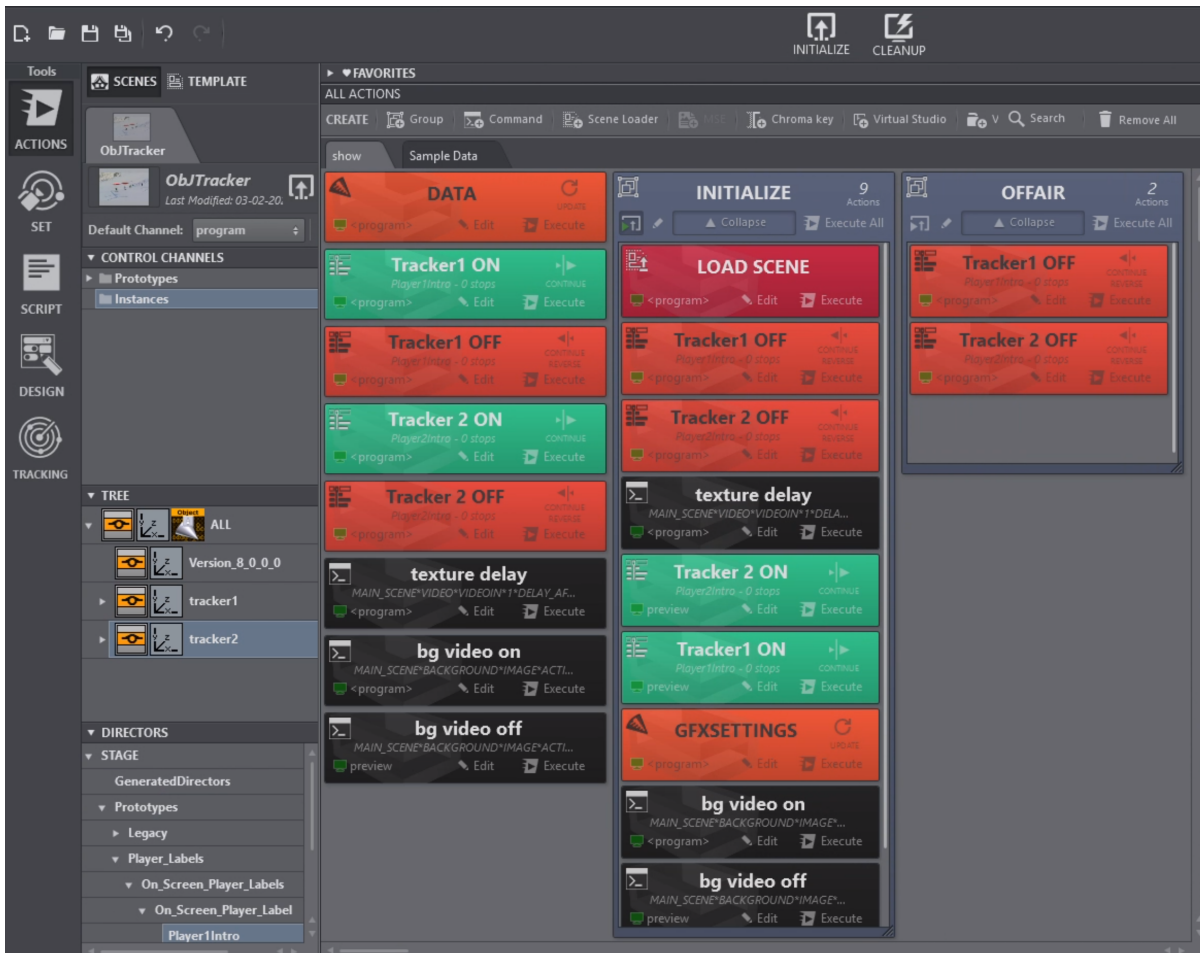


- |                    |                      |
|--------------------|----------------------|
| 0. nose            | 17. left_pinky       |
| 1. left_eye_inner  | 18. right_pinky      |
| 2. left_eye        | 19. left_index       |
| 3. left_eye_outer  | 20. right_index      |
| 4. right_eye_inner | 21. left_thumb       |
| 5. right_eye       | 22. right_thumb      |
| 6. right_eye_outer | 23. left_hip         |
| 7. left_ear        | 24. right_hip        |
| 8. right_ear       | 25. left_knee        |
| 9. mouth_left      | 26. right_knee       |
| 10. mouth_right    | 27. left_ankle       |
| 11. left_shoulder  | 28. right_ankle      |
| 12. right_shoulder | 29. left_heel        |
| 13. left_elbow     | 30. right_heel       |
| 14. right_elbow    | 31. left_foot_index  |
| 15. left_wrist     | 32. right_foot_index |
| 16. right_wrist    |                      |

If a landmark is not detected its coordinates are outside the screen (x:-1.0 y:-1.0 z:-1.0).

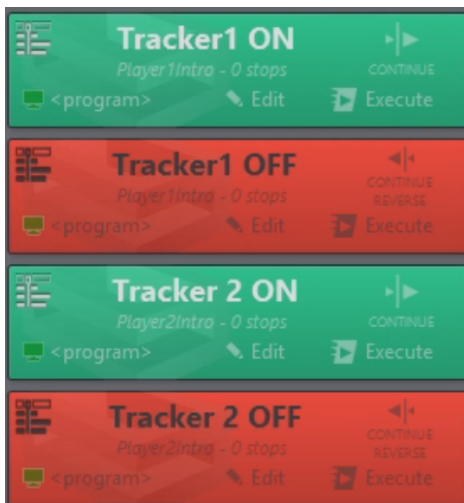
## 5 Preparing Operations

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.

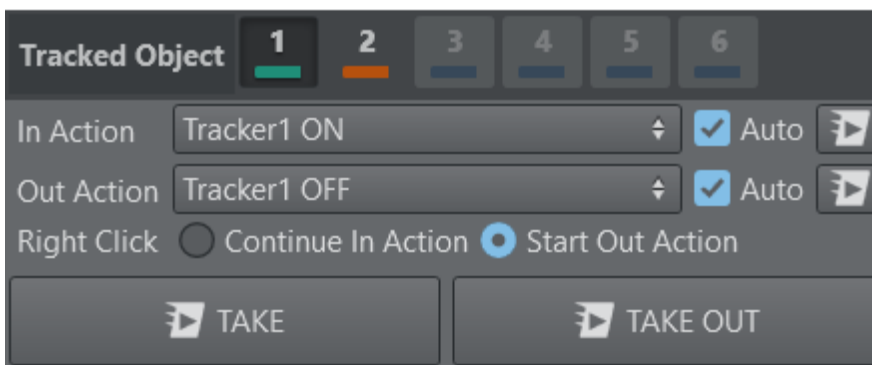




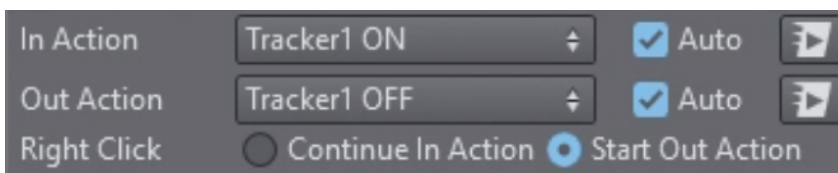
Those actions can be used to trigger an in animation when starting to track object and an out animation when the tracking is lost or the operator takes it Off Air.

## 5.1 Tracking Panel

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



### 5.1.1 In/out Actions



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.

### 5.1.2 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.

### 5.1.3 Tracked Object

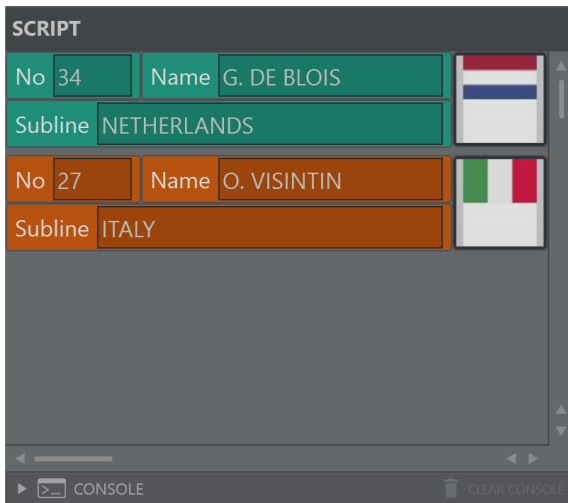
Select the active tracker from the available trackers.



The **Keyboard Shortcut Tab** allows you to switch from one active tracker to the next. It is also possible to directly activate a tracker by pressing the keyboard numbers from **1** to **6**.

Right clicking the toggle buttons of the trackers changes the assigned color.

## 5.2 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.

## 6 Configuration Panel

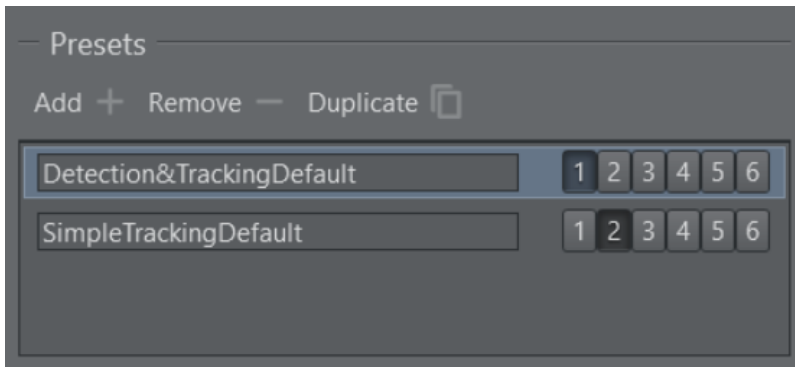
The Configuration Panel is organized into several sections:

- CONFIGURATION** (Dropdown menu)
- Presets**
  - Buttons: Add +, Remove -, Duplicate [icon]
  - Table of presets:

Preset Name	1	2	3	4	5	6
Detection&TrackingDefault						
SimpleTrackingDefault						
- Tracker settings**
  - Analysis: DETECTION&TRACKING
  - Detection DB: generic (0.5)
  - BB filter size: 15 (Slider)
  - Tracking: AdaptiveCenter
  - Filter: AdaptiveLowPass (size: 3)
  - Delay offset: fields: 4 (Slider)
  - Apply Changes button
- Global settings**
  - Preview opacity: 1.00 (Slider)
  - Cut detection: [Toggle]
  - Pointers offset: [Toggle] [Reset button]
- Input Source**
  - SMURF
  - video path- [Browse button]

## 6.1 Presets

It is possible to create multiple presets where you can select which tracking algorithm to use and which neural network to work with for the AI tracker. Each algorithm has specific settings which are exposed in the *Tracker Settings*. One or more Tracking ID(s) can be assigned to a preset. All active presets (the ones that have at least one tracking ID assigned) run altogether and the Object Tracker switches algorithms seamlessly according with the current tracking ID.



## 6.2 Tracker Settings

### 6.2.1 Analysis

#### Manual

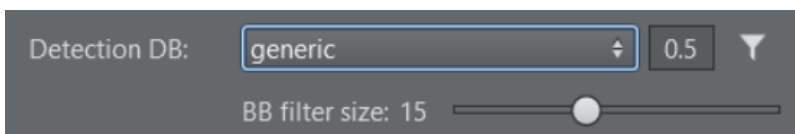
This mode allows you to track manually using the mouse pointer, touch screen or pen input.

#### Detection and Tracking

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

#### Detection DB

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



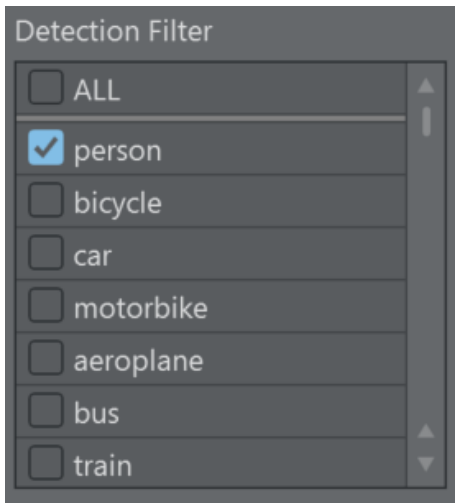
The **generic** network is always available and offers a couple of standard objects that it recognizes (for example, people, cars, etc.).

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 if the button **Detection Filter** is pressed. This is the list of objects that are recognized by the neural network.

## Detection Filter

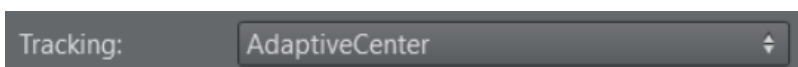


Select which objects of you want to detect. In most cases, you would like to select one or two kind of objects in a video stream.

## BB (Bounding Box) Filter Size

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 aligned to the bounding box. If you're using the *Simple Tracking*, the bounding box size slider is disable since there are no bounding boxes.

## Tracking

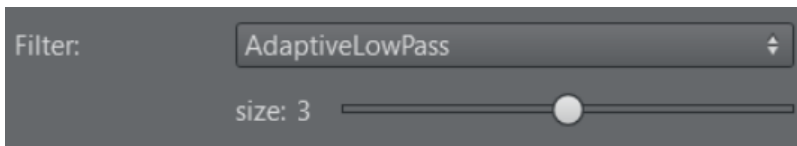


The Tracking Types dropdown lets you choose what kind refined tracking to use inside of the detected bounding box. The button next to the drop down sends the new settings to the Object Tracker without having to use the global initialize button.

- **AdaptiveCenter:** Extracts color/luminance 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.
- **AdaptiveTop:** Extracts color/luminance 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.

- **MagneticCenter:** Works as *AdaptiveCenter* but when the object is not detected it uses the same algorithm of the *Simple Tracking - Adaptive* to try to follow the object; the downside is that the tracking is never lost, the operator has to manually turn off the tracking.
- **MagneticTop:** Works as *AdaptiveTop* but when the object is not detected it uses the same algorithm of the *Simple Tracking - Adaptive* to try to follow the object; the downside is that the tracking is never lost, the operator has to manually turn off the tracking.
- **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. A gravity force re-centers the tracking point toward the center of the bounding box over time to avoid drifting.
- **Simple:** Uses the center of the detected bounding box as tracked reference.

## Filter



## Filter Types

The filter types allow you to filter the tracked point over time. Several 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.

- **AdaptiveLowPass:** Minimizes jitter and lag using a low pass filter.
- **MovingAverage:** Calculates the average of the last points.
- **ExponentialMovingAverage:** Gives more weight to the most recent points. Similar to *MovingAverage*.

## Size

Increasing the size of the filter helps to smooth noisy results.

**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.

### 1 Detected object's bounding box



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

## Simple Tracking

### Tracking Types

This option tracks objects without the aid of a neural network: it uses the same algorithm that are used for the refined tracking in the *Detection and Tracking*.

- **Adaptive** : Extracts color/luminance features around the clicked point.
- **KCF**: Extracts generic features around the clicked item.

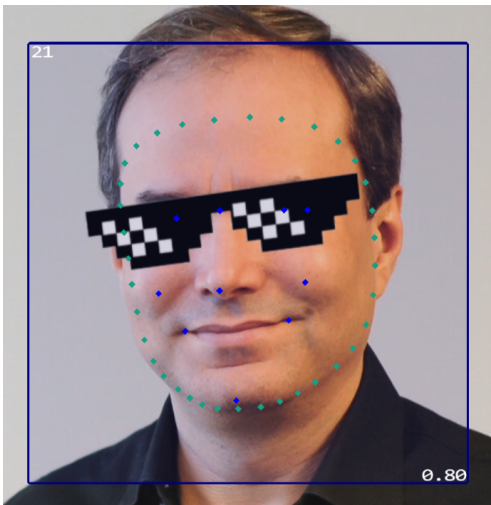
### Filters

same as the Filter in "Detection&Tracking"

### Face Tracking

This mode allows you to track up to two faces (the ones with the best results).



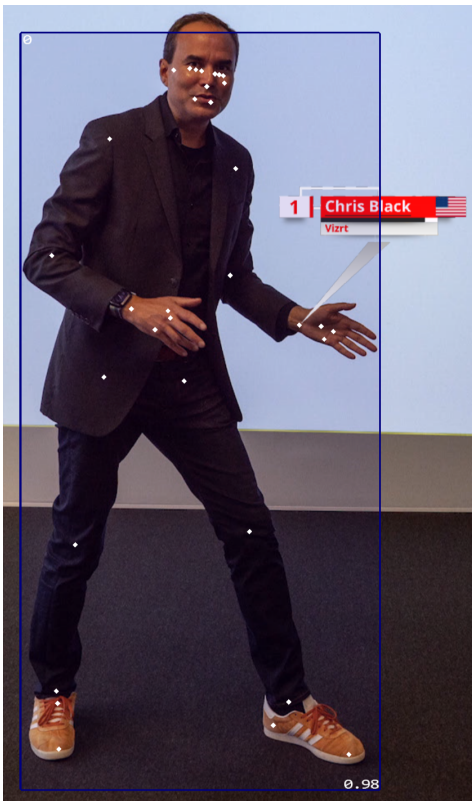


The points sent to Viz Engine(s) are:

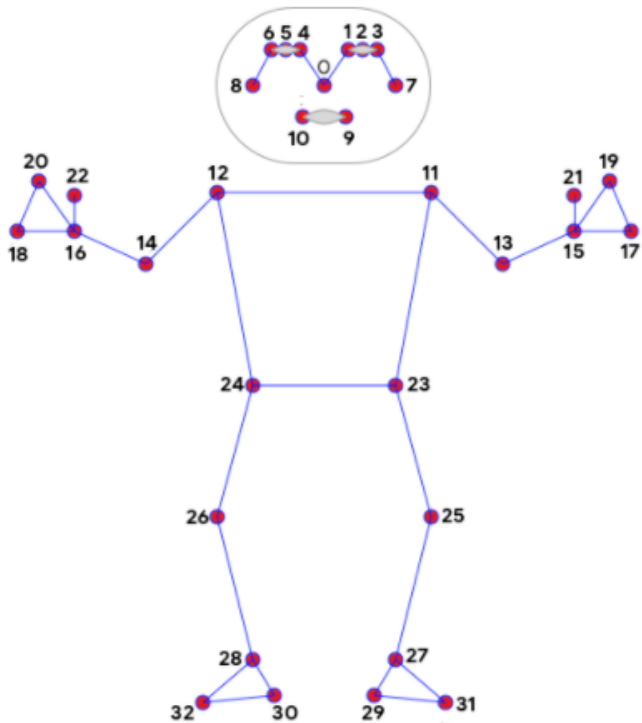
- Outline of the face
- EyeLeft
- EyeRight
- Mouth
- BetweenEyes
- EyeCornerLeft
- EyeCornerRight
- PupilLeft
- PupilRight
- NoseTip
- CheekLeft
- CheekRight
- MouthCornerLeft
- MouthCornerRight
- Chin

## 2D Pose Tracking

This mode allows you to track the pose of one person.



This is the map of the points sent to Viz Engine(s):



- |                    |                      |
|--------------------|----------------------|
| 0. nose            | 17. left_pinky       |
| 1. left_eye_inner  | 18. right_pinky      |
| 2. left_eye        | 19. left_index       |
| 3. left_eye_outer  | 20. right_index      |
| 4. right_eye_inner | 21. left_thumb       |
| 5. right_eye       | 22. right_thumb      |
| 6. right_eye_outer | 23. left_hip         |
| 7. left_ear        | 24. right_hip        |
| 8. right_ear       | 25. left_knee        |
| 9. mouth_left      | 26. right_knee       |
| 10. mouth_right    | 27. left_ankle       |
| 11. left_shoulder  | 28. right_ankle      |
| 12. right_shoulder | 29. left_heel        |
| 13. left_elbow     | 30. right_heel       |
| 14. right_elbow    | 31. left_foot_index  |
| 15. left_wrist     | 32. right_foot_index |
| 16. right_wrist    |                      |

## 6.2.2 Delay Offset



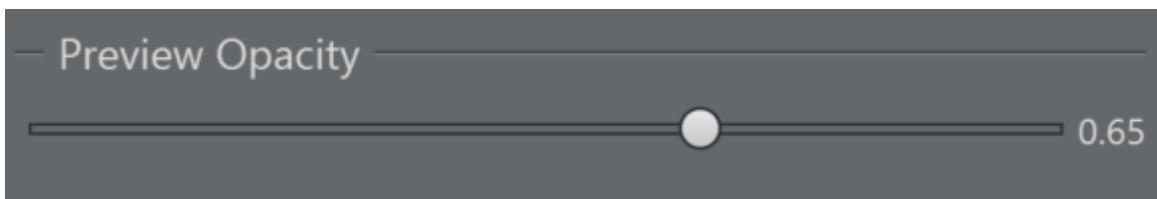
The Object Tracker delays the results to match the video images with the tracking information; this slider increases (max +10) or decreases (min -10) the delay, in fields, applied. This can be useful to compensate the lag due to aggressive filtering.

**i Apply Changes:** Please note that the settings that can be changed in the tracker panel do not get applied as you change them in the Object Tracker. A yellow triangle informs you if the parameters shown in the panel do not respect the values used by the running algorithms; some parameters can but updated pushing **Apply Changes** for other a re-initialization is needed. A tooltip tells you what you need to do.



## 6.3 Global Settings

### 6.3.1 Preview Opacity

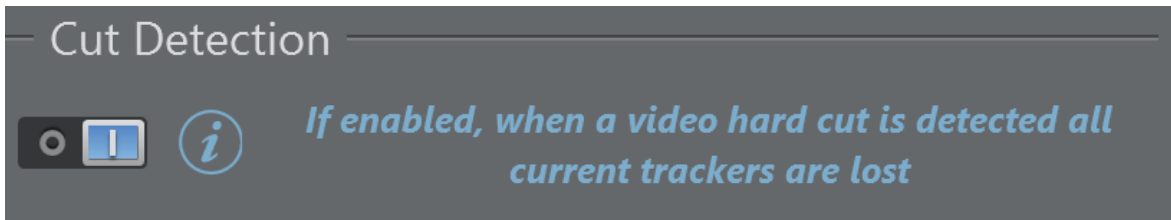


The preview output that has been configured in Viz Arc can be rendered on top of the NDI stream coming from the Object Tracker. This allows the operator to preview the graphics on screen. The opacity of this preview can be adjusted here.

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

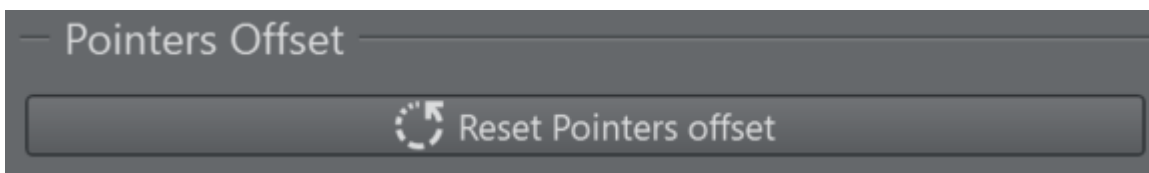
**i Information:** For performance reasons, the composed on-screen preview might not look the same as on the actual program output. Make sure the fill channel of the preview is black where it is supposed to be transparent. The timing between the underlying NDI stream and preview is not guaranteed to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

### 6.3.2 Cut Detecion



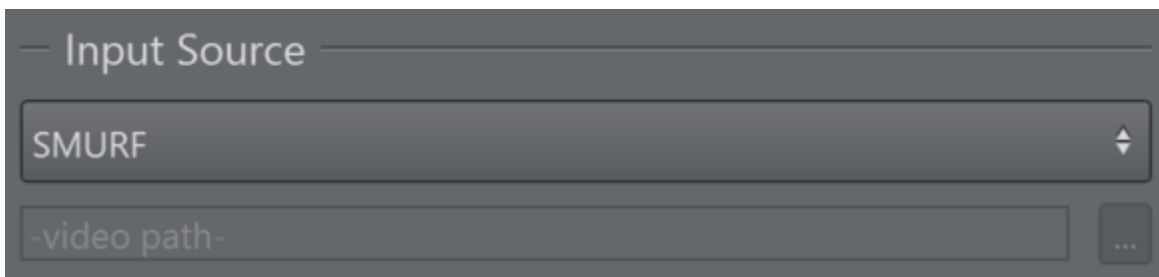
If enabled, when a video hard cut is detected all current trackers are lost.

### 6.3.3 Pointers Offset



It resets the pointers offset of each tracking input.

### 6.3.4 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.

---

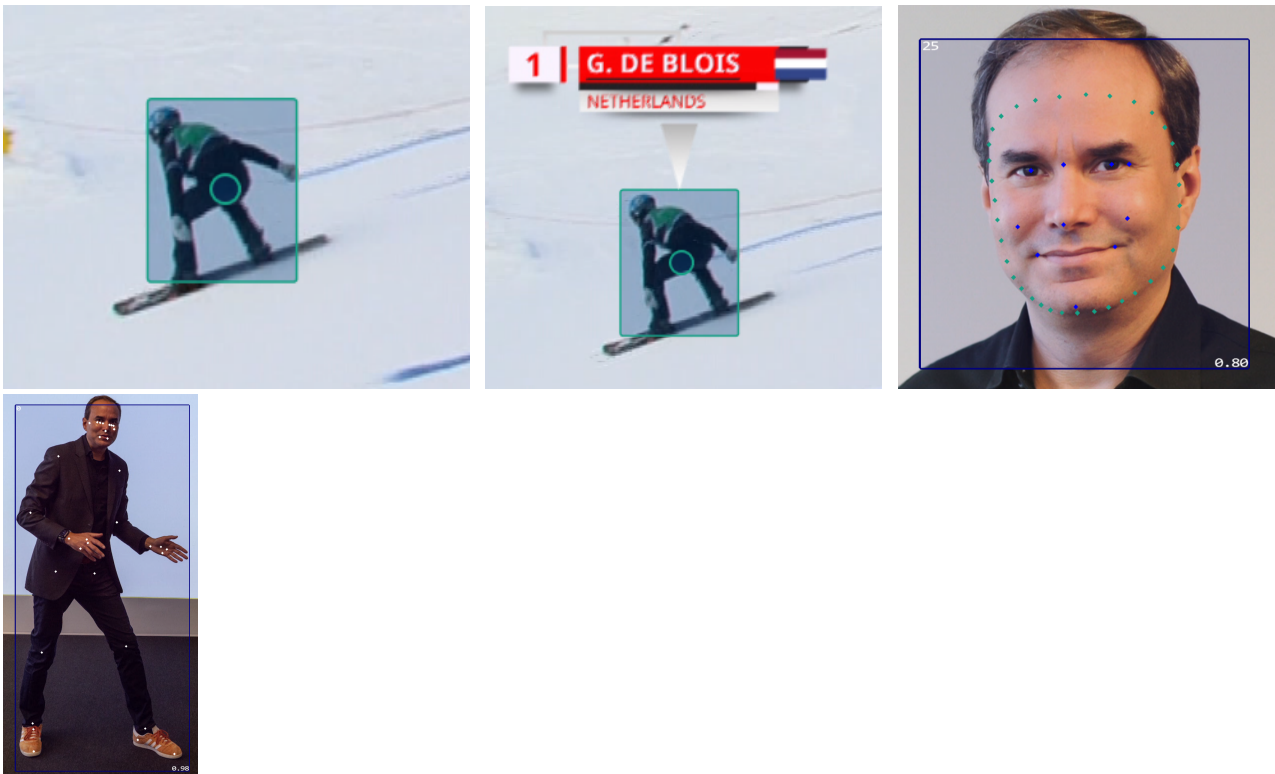
## 7 Operating The Object Tracker

---

### 7.1 Detection And Tracking, Face Tracking And 2D Pose Tracking



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 channel.

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.



## 7.2 Simple Tracking



To use it, click on the point you want to follow, the algorithm attempts to go after it. Obstructions are the main reason it gets lost.

---

## 7.3 Manual Tracking

It is possible to work with manual tracking, if there are cases where both *Detection and Tracking* and *Simple Tracking* fail. To use it, just hold down the left mouse button to start tracking and release it to lose tracking.

---

## 7.4 Offsets

If you want to apply an offset to displace the pointer, the steps are:

- First, you have to enable the option in the config menu of the Tracking Object.
- When you click on the frame, you have to keep the mouse down (the image freezes).
- Move the mouse around until you are satisfied with the offset.
- Release the mouse to apply the offset.
  - If the object is still visible, you'll see the pointer with the displacement applied.
  - If the object is no longer visible, you'll have to click on a new object to see the result.

