BREKEL

BODY

V3

WHAT IS BREKEL BODY V3

Brekel Body v3 is a Windows (x64) application for markerless body motion capture supporting multiple depth sensors from various brands and types. This includes (optional) tracking of face expressions.

Multiple sensors on the same machine (depending on the sensors) and/or across multiple networked machines are supported and their data can be fused to provide better quality than a single sensor.

It is written by Jasper Brekelmans, so in case you're wondering that's what "Brekel" refers to.

"Brekel" is pronounced as "Break–uhl".

SUPPORTED SENSORS

- Kinect for Xbox 360 / Kinect for Windows v1 (with USB2/power adapter)
  - Using the official Microsoft Kinect v1.8 drivers
  - Multiple sensors per machine are supported
  - If your machine has dedicated USB2 ports (black) it's best to prefer these

- Kinect for Xbox One / Kinect for Windows v2 (with USB3/power adapter)
  - Using the official Microsoft Kinect v2 drivers
  - Due to driver/SDK restrictions only a single sensor per machine is supported
  - Multiple networked machines each with 1 sensor are supported
  - Only USB3 ports (blue) with Intel or Renesas chipsets are supported by the drivers

- Azure Kinect
  - Dedicated USB3 port required
  - Nvidia Geforce GTX 1070 or better required (tracking will not work at all on GPUs without CUDA support)
  - Cannot be combined with Orbbec Femto sensor(s) on the same machine

- Orbbec Femto
  - Dedicated USB3 port required
  - Nvidia Geforce GTX 1070 or better required (tracking will not work at all on GPUs without CUDA support)
  - Cannot be combined with Azure Kinect sensor(s) on the same machine

- Orbbec Astra
  - All Astra variants (Pro, non-Pro, S, Mini) are supported
  - Astra Persee is not supported
  - Astra Femto / Astra+ not supported (yet)
  - Due to SDK restrictions only a single sensor per machine is supported for body tracking
  - Multiple networked machines each with 1 sensor are supported
  - If your machine has dedicated USB2 ports (black) it's best to prefer these

SENSOR DRIVER INSTALLATION

Some sensors need drivers in order to work.

The Brekel app will try to automatically detect if drivers are installed on your system already, if not there should be a “download & install” button next to the sensor type in the “Connect” panel on the bottom left.

The app will always download the latest and official drivers from the manufacturer’s website, if that link is down it will fall back to a mirror.

You can also use the options under “Settings” on the top menu to download & install the latest sensor drivers.

Note that you may need to restart your computer for some sensors to be detected after installing drivers.

Note that if you already have 3rd party or non-official drivers installed you may need to uninstall those yourself first to avoid driver conflicts.

USB CONNECTIVITY & BANDWIDTH

Depth sensors are high bandwidth devices that can easily use all the available bandwidth on your USB chipset.

A few tips for best results:

- Many newer sensors (Azure Kinect, RealSense) need a USB3 port to function.
- If your machine has dedicated USB2 (black) ports it's best to prefer those for USB2 sensors like (Kinect v1 & Orbbec Astra), they should however work on USB3 (blue) ports in most cases as well.
- Most desktop/workstation machines have up to 2 internal USB controller, when having problems switching a sensor to a different port on the front/side/back of your machine may help balancing the load to a different controller
- In many cases laptop or mini computers (like Intel NUC) may only have a single USB controller limiting the amount of bandwidth they have.
- You may need to purchase additional PCI USB controllers to expand your (desktop/workstation) machine, the StarTech PEXUSB3S44V is an excellent card with 4 dedicated high bandwidth ports that I can recommend.
- Typical symptoms of reaching your machine's bandwidth include things like connection issues, dropped frames, sensors freezing datastreams.

MINIMUM SYSTEM REQUIREMENTS

It is hard to state minimum required system specifications as this is highly dependent on sensor types and the number of sensors used.

- 64-bit Windows 10 (for older sensors Windows 8/8.1 may work but this is untested)
- 8 GB or more RAM
- Modern Intel i5 or faster CPU (or AMD equivalent)
    - dual core 2.4GHz or i3 may be enough for older sensor types with single sensor setups
    - more cores help for multi sensors setups as the app is multi-threaded
    - 3 Ghz or faster will help for newer sensors and multi sensor setups
- Modern GPU with OpenGL support
    - Microsoft's Azure Kinect body tracking runs on Nvidia CUDA, requiring a GeForce GTX 1070 or better GPU (and currently uses around 150 GFLOPS & 1.9 GB of VRAM per instance)
- 1280×1024 screen (recommended: 1920×1080 or higher)
- CPU with AVX2 instruction support is required for Deep Tracker functionality

NETWORK SENSORS

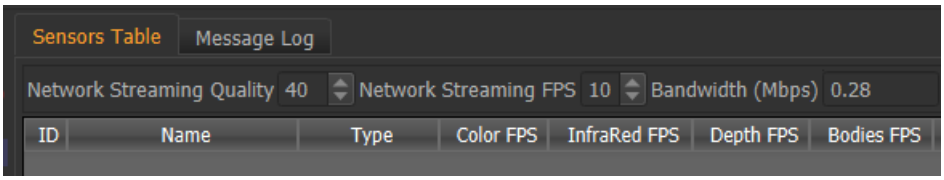
Besides utilizing the sensors directly connected to your machine it's possible to use data from sensors connected to other machines in your network.

Make sure the Brekel app is installed on all machines and the sensors work when running the software locally in GUI mode. (You can use the same license on each machine)

- Pick one machine (the fastest) to run the Brekel GUI application.
- On the other machines start the Brekel app in headless mode (there should be a dedicated shortcut in your Windows Start Menu and on your desktop).
- In the GUI select "Network" sensors and hit "Connect Sensors".
- A broadcast signal will be sent out on your network (UDP port 9875).
- The apps running in headless mode will pick this up and reply back by reporting with details on what sensors are connected to them.
- The GUI and headless apps will then communicate further over TCP port 9876

Note that this should work over a wireless connection but in most practical use cases you'll want a wired connection for maximum reliability and speed.

In the GUI you can use the Network Streaming Quality and FPS settings that will appear above the sensors table to adjust bandwidth vs quality, note that this only affects quality of the points/video in the 3D/2D viewports, body tracking data (as well as BPC pointcloud recording) is always at the highest quality.



NETWORK TROUBLESHOOTING

There can be several reasons why network communication does not work, the most common ones:

- Make sure all your machines are connected to the same network
- Security software and/or firewalls blocking communication
    - make sure UDP port 9875 and TCP port 9876 are not blocked
    - Make sure the apps are not blocked
- Multiple network cards (see below)

The app should automatically try to find the most suitable network adapter (in case there are multiple in your system) and it should prefer ethernet over Wi-Fi.

If auto detection fails, you can manually specify which network adapter to use using Settings > Select Preferred Network Adapter from the top menu.

Please consult with your system/network admin to resolve any of these things.

HEADLESS/CONSOLE MODE

The headless mode exists to run on remote computers sending their sensor data over the network to the main computer running the GUI version. It will consume less CPU/GPU resources as it won't need to draw visualizations on the screen.

You'll find a dedicated shortcut in the Windows start menu and your desktop to start it.

To run the software in headless mode the "-headless" command line option is all that is needed.

By default, it will try to find all supported sensors, by adding the following optional command line options you can control which connected sensors will be used.

-kinect1	find and use Kinect v1 / Kinect for Xbox 360 sensors
-!kinect1	don't use Kinect v1 / Kinect for Xbox 360 sensors
-kinect2	find and use Kinect v2 / Kinect for Xbox One sensors
-!kinect2	don't use Kinect v2 / Kinect for Xbox One sensors
-orbbec	find and use Orbbec Astra sensors
-!orbbec	don't use Orbbec Astra sensors
-azurekinect	find any Azure Kinect sensors
-!azurekinect	don't use Azure Kinect sensors

QUICK START GUIDE

To get started:

- Select which sensor(s) you have on your system in the “Connect” tab on the bottom left
- Hit the “Connect Sensors” button and wait for your sensors to show up in the “Sensors Table”
- Sensors will need to be aligned first, use the features on the “Align Sensors” panel at the top right
    - See the mouse-over tooltips and chapters in this manual regarding sensor alignment
- After sensors are aligned the solver will match bodies seen by multiple sensors and fuse their data
- These “solved bodies” will by default be drawn using bones in the 3D viewport
- Select which output format(s) you want in the “3D Output Formats” tab on the right
- Set the folders to record to for each format
- Set the filename to record to and hit “Start Recording” on the bottom right
- Open your file in your favorite 3D app

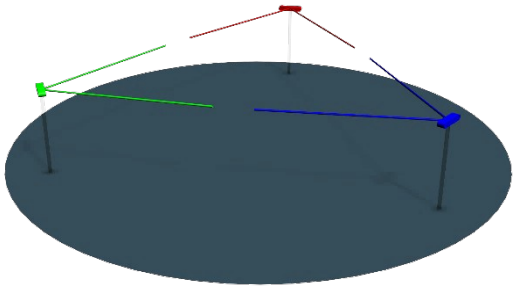
SENSOR PLACEMENT

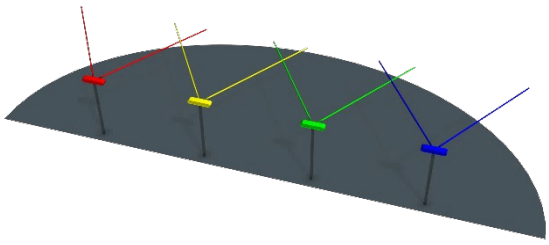
The only requirements the software has on sensor placement are:

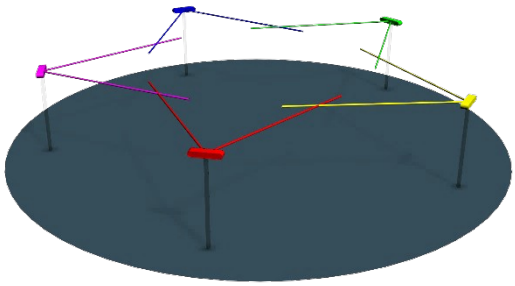
- They need to be able to see the person(s) to be tracked
- Multiple sensors need to have some overlap for calibration and data fusion

Some guidelines for sensor placement:

- Between hip and head is usually a good height for a sensor
- It is recommended that sensors can see part of the floor
- Placing your sensors in a circle, facing inwards, allows for 360-degree tracking
- Placing sensors on a line, facing the same direction and with some overlap allows for expanding the tracking area
- When adding more sensors try to space them evenly
- With enough sensors it may be beneficial to place some of them lower and others higher

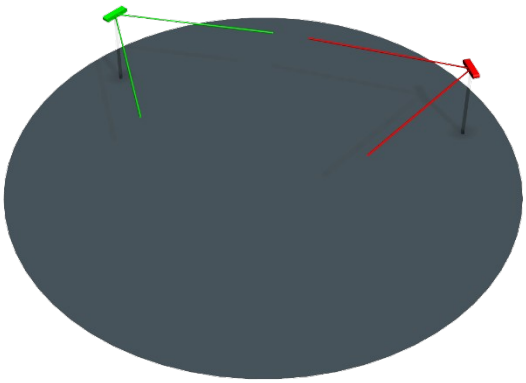




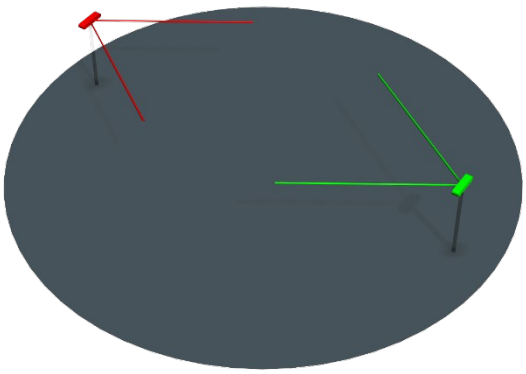


When working with 2 sensors it should still be possible to get 360-degree capture (although much less stable than 3 or more sensors).

It is best to have up to 120 degrees of separation between the sensors (both sensors predominantly in front of the subject).



Avoid a setup with 180-degree separation (one sensor at the front, other at the back) as this will not provide enough (or any) overlap.



TRACKING ENVIRONMENT

All depth sensors (currently) track using infrared light, some things that can negatively impact tracking quality and should be avoided are:

- Direct sunlight in your tracking volume can overpower the sensor's active infrared illuminators and cause tracking noise or tracking to fail completely
- Dark clothing can cause too little infrared light to be reflected and cause noise (note that not all black clothing in visible light shows up as black in infrared, it depends on the material)
- Some floors can be highly reflective (polished wooden floors for example) causing foot tracking noise, in most cases carpeting or a rug will help
- Loose fitting clothing will make joint estimation a bit more uncertain than tight clothing
- Sleeves on shirts and pants that hide the feet and hands will make joint estimation more uncertain on those areas

KINECT V1 (XBOX 360) SPECIFIC

Kinect v1 sensors use USB2 speeds, they can be connected to USB2 or USB3 ports.

Since they use relatively low bandwidth you should be able to use multiple sensors on pretty much all machines.

There may be some interference, in the form of added pointcloud noise, in areas where multiple sensors overlap. In general this should not have much impact on skeleton tracking.

These sensors use a machine learning based tracker which always assumes the user is facing the sensor.

It may be beneficial to enable the app's Deep Tracking functionality to improve tracking stability and be able to distinguish front/back of the user better.

KINECT V2 (XBOX ONE) SPECIFIC

Kinect v2 sensors need to be connected to a USB3 port on your machine.

Due to driver/SDK restrictions only a single sensor per machine is supported, you can of course use multiple networked machines to use setups with multiple Kinect v2 sensors.

There may be some interference at times, in the form of fluctuating distance readings, in areas where multiple sensors overlap. This is caused by these sensors using Time Of Flight and not having hardware sync options.

These sensors use a machine learning based tracker which always assumes the user is facing the sensor.

It may be beneficial to enable the app's Deep Tracking functionality to improve tracking stability and be able to distinguish front/back of the user better.

AZURE KINECT SPECIFIC

Note that you will need an USB3 port for your Azure Kinect.

When running multiple sensors make sure your machine can handle the needed USB bandwidth, laptops may have limited bandwidth, desktop machines can be extended with a card like StarTech PEXUSB3S44V for example.

HARDWARE SYNC

Azure Kinect sensors have sync in/out ports on the back (remove the cover first) that allow multiple sensors to be synchronized. This has the benefit or reducing interference to a minimum so is generally a good idea to use.

All you need to do is run sync cables between devices as indicated here:

https://docs.microsoft.com/en-us/azure/Kinect-dk/multi-camera-sync

(note that the ports are located underneath the cover, remove the screws in the back to take it off)

The Brekel app will automatically identify which sensors are connected as master and which as subordinate (based on if there are connections to the sync in/out ports) and set things up accordingly when accessing the sensors. It will report what it detected in the message log after connecting the sensors.

Note that the body solver was designed to work both with synchronized and non-synchronized data.

When running a single sensor or multiple sensors without sync cables the app will use the Infrared video stream for fastest computation speed.

When running multiple sensors with sync cables the app will use the Color video stream as this is currently required for drivers to use synchronization, this may be a little slower as a bit more computation is needed internally.

COMPUTATION PLATFORM

The tracker can run on different underlying deep learning networks, NVIDA CUDA, DirectML, TensorRT and CPU. Depending on your GPU/CPU configuration you may want to experiment which is the fastest on your machine.

You can set these options by selecting your Azure Kinect sensor from the sensor table and changing the settings (for the selected sensor) on the “Selected Sensor Settings” tab.

For NVIDIA GPUs CUDA is usually the fastest option, for recent high-end GPUs that support TensorRT (RTX 2xxx/3xxx cards for example) you may want to try TensorRT for higher performance.

For other AMD/Intel GPUs your best option is most likely DirectML. (this should also work on NVIDIA GPUs but generally is slower than CUDA/TensorRT)

As a last resort you can run the tracking model on the CPU but in most cases this is slower than the GPU options.

BODY TRACKING MODEL

Microsoft’s Azure Kinect Body Tracker can run two different models, one that is optimized for accuracy and a lite model that is optimized for speed. The difference is roughly a factor of 2 in performance and about 5% in accuracy.

The tracker uses a deep learning based model that can distinguish the front/back of a person, in general it is not needed to run the app’s Deep Tracking feature in conjunction with these sensors.

Azure Kinect sensors currently cannot be used with Orbbec Femto sensors on the same machine.

Since Orbbec Femto uses a custom version of the Azure Kinect k4a.dll that is not compatible with the Azure Kinect devices. The application contains functionality to easily switch.

ORBBEC FEMTO SPECIFIC

The Orbbec Femto sensor(s) are supported using the K4A Wrapper which wraps the Azure Kinect SDK to the OrbbecSDK, this is used so the Azure Kinect Body Tracker can also be used since it's the best quality option for body pose estimation for these sensors at the moment.

Unfortunately, since both the Orbbec Femto and Azure Kinect sensors use their own k4a.dll file and they are not compatible with one another you can currently only use one type of sensor per machine.

The application contains functionality to easily switch.

ORBBEC ASTRA SPECIFIC

The Orbbec Astra Pro models are currently supported, these are USB2 sensors with comparable specs to Kinect v1 (Xbox 360) sensors.

They can be connected to USB2 or USB3 ports.

Due to a driver/SDK restriction body tracking currently only works on a single sensor per machine.

Due to a restriction with Orbbec's body tracking license you will only work up to 30 minutes at a time, you will need to restart the app after this period.

There may be some interference, in the form of added pointcloud noise, in areas where multiple sensors overlap. In general this should not have much impact on skeleton tracking.

These sensors use a machine learning based tracker which always assumes the user is facing the sensor.

It may be beneficial to enable the app's Deep Tracking functionality to improve tracking stability and be able to distinguish front/back of the user better.

Orbbec Embedded S sensors should work but given their form factor are not recommended.

Orbbec Stereo S sensors are not recommended, since they rely on stereo sensing and are rather noisy they are not well suited for body tracking purposes.

INTEL REALSENSE SPECIFIC

Currently the following Intel RealSense sensors are supported:

SR305 – structured light, short range

D415 – stereo, long range

D435 – stereo, long range

D455 – stereo, long range

L515 – LIDAR, medium range

All sensors work best with USB3 speeds/ports but can work (with restrictions) on USB2 ports.

Body tracking is not natively supported by these sensors so you will have to run the Deep Tracking functionality of the Brekel app. (either in CPU or GPU mode).

Due to this body tracking quality may not be as optimal as the Kinect sensors for example.

STEREOLABS ZED2 SPECIFIC

Currently only the ZED 2 sensor is supported for body tracking, not the ZED 1 or ZED Mini.

This app was built with the v3.8 drivers, you will need to download those from the Stereolabs website yourself, once the v4.0 drivers are released (they are currently still in beta) the Brekel app will switch to those.

A USB3 port is needed on your machine.

The first time the native tracking model is used, it may need up to a few minutes to initialize and optimize things for your machine. On subsequent usage this should be much quicker.

The tracker uses a deep learning based model that can distinguish the front/back of a person, in general it is not needed to run the app's Deep Tracking feature in conjunction with these sensors.

WEBCAM SPECIFIC

Webcams require the Deep Tracker in GPU mode, which means this will only work on an NVIDIA RTX card.

In order to produce a 3D skeleton the depth coordinate will be estimated using the field of view of the sensor.

For the following Logitech webcam models this should be automatically detected: C270, C310, C505, C920, C922, C925, Stream & Brio.

Other webcam brands/types should still work but the user should manually define the “Field of View” setting on the “Sensor Settings” panel.

In general webcams do not produce the same accuracy as depth sensors but they can be a great to get additional viewpoints. They are generally less expensive than depth sensors, require less USB bandwidth and some models can run at 60 frames per second.

It is best to mix webcams to a setup that also uses one or more depth sensors but setups with only webcams should be possible.

Note that due to the nature of deep learning based tracking there is generally no benefit of running really high resolutions. 640x480 is sufficient but since 1280x720 provides a wider field of view this is generally the best resolution. Running at a higher resolution will only increase computation needs, USB bandwidth but will not produce higher tracking accuracy.

TRACKING MULTIPLE PEOPLE

Most depth sensors support tracking multiple people.

Currently the GPU deep tracker only supports tracking a single person (this may change in the future).

Webcams rely on this and will thus only track a single person per camera.

Depth sensors that use this feature will also only use this data for a single person per sensor.

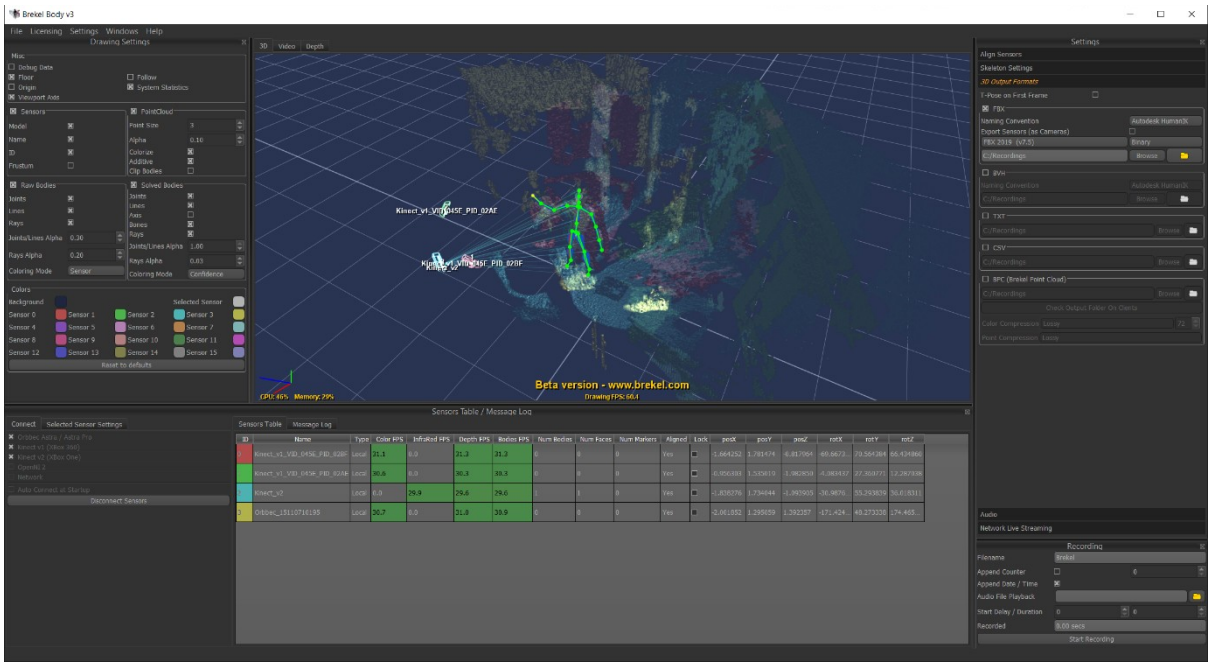
In the case multiple people are in view the deep tracker may randomly pick who to track and may switch between frames, so this use case is not recommended.

The Brekel skeleton solver will collect all the raw skeleton data it gets from the sensors and will fuse data (using the alignment calibration info) of skeletons that are overlapping in 3D space.

This means you can track multiple people that are visible in one or more sensors at a time.

Tracking multiple people with close interaction and occlusions may be challenging depending on the amount of sensors and if they can distinguish between the persons seen.

INTERFACE



Left: Drawing Settings
Middle: Main 3D Viewport / Video / Depth
Right: Settings
Bottom: Sensors Table & Message Log

- The docks on the left, right and bottom can be undocked and turned into a floating window by dragging from their title bar
- Double clicking on the title bar of a floating window will dock it back into the main window
- You can toggle windows on/off under “Windows” in the top menu
- Right Clicking on the title bar of the main window (next to Help) will pop up a list that allows showing hidden windows again
- The overall GUI state will automatically be saved on exit and reloaded on start
- Most buttons and widgets have tooltips if you hover your mouse over them for a few seconds

Most of the data is visualized in the main 3D viewport, the Video & Depth windows show the raw sensor streams and can be of great help to help understand which part of your volume is seen by each sensor especially when setting things up.

3D VIEWPORT NAVIGATION

TO ORBIT

- Left mouse button, click & drag

TO DOLLY IN/OUT

- Middle mouse button, click & drag
- Shift + Left mouse button, click & drag
- Mouse wheel

TO PAN

- Right mouse button, click & drag
- CTRL + Left mouse button, click & drag

TOP MENU

FILE MENU

Check for new version on startup

If enabled the application will automatically checks if a newer version is available for download from the website at startup.

Exit

Closes the application

LICENSING MENU

Show License Info

Shows a window with your licensing information

Install License (Automatic)

Will ask you to browse for your license file (the zip attached to the license email you received within 24 hours after purchase). The app will try to automatically install the license by extracting and copying the .key file to your “ProgramData” folder.

Install License (Manually)

In case automatic license installation (see above) fails you can use this to open the folder where the license file needs to be put. Extract the zip file (attached to the license email you received within 24 hours after purchase) to this folder so the .key files is present.

Remove License File

Remove license file from this machine so you can install it on another machine

SETTINGS MENU

Save Settings Now

Saves all the current GUI settings in the registry immediately (also done automatically on application exit and periodically when changing settings)

Show Advanced Options

Shows/hides additional advanced options in the GUI, for example to tune the skeleton solver. (not needed for most users)

Enable High DPI Scaling

Adjust the GUI for high DPI screens, you need to restart the application for this to take effect.

Use Native File Dialogs

Switches which type of file/folder dialogs to use, Windows native or Qt.

Warn for File Overwriting

When enabled shows a popup message warning if files will be overwritten when starting a new recording.

Force Color for Kinect v2

Forces the use of the color stream (instead of infrared) for Kinect v2 (Xbox One) sensors.

This will be a bit slower for body tracking but will allow you to record BPC pointcloud files with color information. (which can be used in Brekel PointCloud)

Select Preferred Network Adapter

Allows you to overrule which network adapter the software should use (just in case the automatic selection that is used by default doesn't work for you)

Print Network Interface Information

Prints details of your network interfaces to the message log, please provide these details if you are contacting support regarding network issues.

Install Drivers <sensor type>

Manually downloads & installs drivers for the supported sensor types.

WINDOWS MENU

Toggles visibility of the various windows like settings and such

HELP MENU

Downloads Page

Opens up the downloads web page in your default browser.

Brekel Forum

Opens the forum web page (hosted on Google groups) in your default browser.

PDF Documentation

Opens this manual in your default PDF browser.

Unity Live Streaming Example

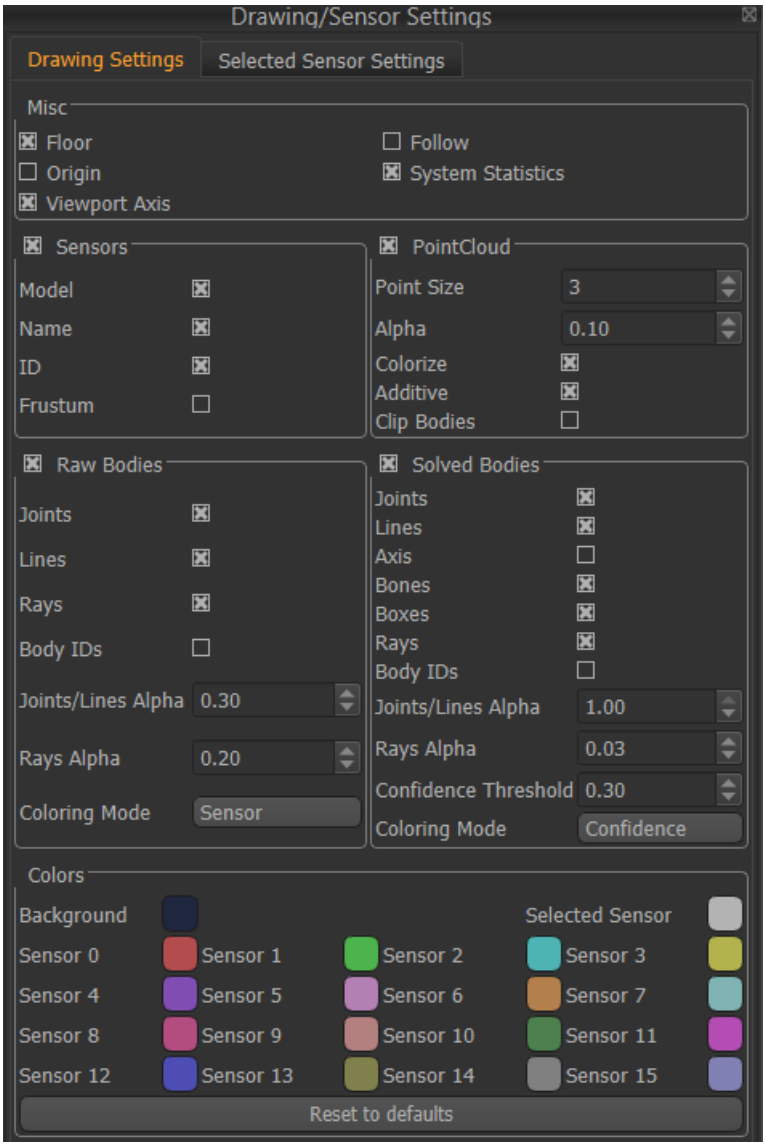
Opens the folder containing the Unity live streaming example script/scene.

KEYBOARD SHORTCUTS

The following keyboard shortcuts are available:

CTRL + Space:	start/stop recording
CTRL + 1:	toggle display sensors
CTRL + 2:	toggle display pointcloud
CTRL + 3:	toggle display raw bodies
CTRL + 4:	toggle display solved bodies
CTRL + 5:	switch main viewport to 3D view
CTRL + 6:	switch main viewport to video view
CTRL + 7:	switch main viewport to depth view
F1	switch to drawing settings tab
F2	switch to selected sensor settings tab
F5	switch to sensors table tab
F6	switch to message log

DRAWING SETTINGS



MISC

Floor	draw the floor plane/grid in the 3D viewport
Origin	draw an axis at the world's origin (0,0,0) in the 3D viewport
Viewport axis	draw an axis in the bottom left corner of the 3D viewport
Follow	toggles if the camera should keep all tracked bodies in the view by following them
System Statistics	draw statistics like frame rate and CPU/Memory utilization in the 3D viewport

SENSORS

Model	draw sensor 3D models in the 3D viewport
Name	draw sensor names in the 3D viewport
ID	draw sensor IDs in the 3D viewport
Frustum	draw approximations to the sensor lenses in the 3D viewport

POINTCLOUD

Point Size	size of the points (in pixels) of the pointcloud in the 3D viewport
Alpha	transparency of the pointcloud in the 3D viewport
Colorize	colorize pointclouds by their sensor color in the 3D viewport
Additive	use additive shading for pointclouds in the 3D viewport
Clip Bodies	clip pointclouds to areas where bodies are detected in the 3D viewport

RAW BODIES

(raw bodies are skeletons that are detected by each sensor individually)

Joints	draw joints in the 3D viewport
Lines	draw lines between joints in the 3D viewport
Rays	draw rays from the sensor to the joints in the 3D viewport
Body IDs	**d**raws tracking ID numbers for each body
Joints/Lines Alpha	transparency of joints & lines in the 3D viewport
Rays Alpha	transparency of rays in the 3D viewport
Coloring Mode	color according to the joint's tracking confidence or the sensor color

SOLVED BODIES

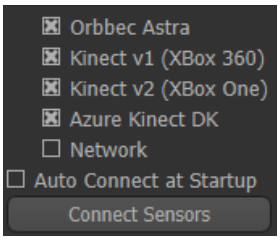
(solved bodies are skeletons that have been matched and fused from multiple sensors)

Joints	draw joints in the 3D viewport
Lines	draw lines between joints in the 3D viewport
Axis	draw axis on each joint to show its rotation
Bones	draw bones between joints in the 3D viewport
Boxes	draw boxes for bones in the 3D viewport
Rays	draw rays from joints towards contributing sensors
Body IDs	**d**raws tracking ID numbers for each body
Joints/Lines Alpha	transparency of joints & lines in the 3D viewport
Rays Alpha	transparency of rays in the 3D viewport
Coloring Mode	color based on tracking confidences or to visualize left/right sides of the body

COLORS

Background	color of the background in the 3D viewport
Selected Sensor	color of the currently selected sensor from the "Sensors Table" in the 3D viewport
Sensor 0 -15	color for all the individual sensors when they're not selected in the 3D viewport

CONNECT



On the “Connect” tab on the bottom left of the GUI you can select which sensors to use.

If a sensor needs a driver and it’s not found on your system a little “download & install” icon will appear next to it if needed.

The various sensors types depict sensors connected to your current machine.

The “Network” sensor depicts sensors connected to other machines on your network, to use these you will need to run the application in “Headless Mode” on that machine, see the chapter about that.

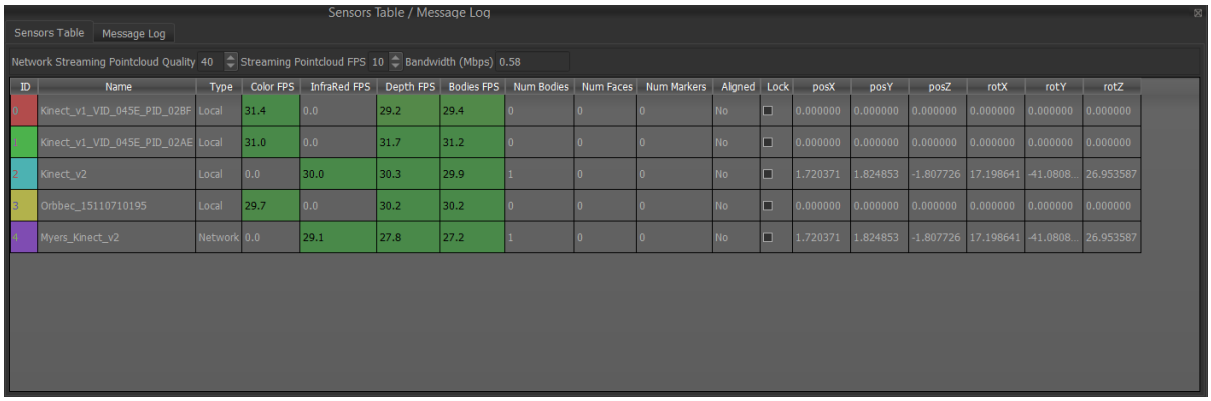
Auto Connect at Startup will automatically hit the “Connect Sensors” when the program has started, which can be handy if you always use the same setup.

“Connect Sensors” will find all the sensors (of the types you selected) it can connect to and start streaming data from them into the application.

After connecting sensors, the button will chance into “Disconnect Sensors” which will allow you to stop streaming data and disconnect.

The software will internally split of each sensor to a different system thread to utilize today’s multicore machines. On the same token some processes like the skeleton solver will also run in different threads.

SENSOR TABLE



On the bottom of the GUI you'll find the "Sensor Table" which shows all connected sensors.

You can select a sensor by clicking somewhere on it in the table with the left mouse button, simply click again to unselect.

You can change some settings of the selected sensor in the tab "Selected Sensor Settings" located in the top left tab (next to Drawing Settings). The specific settings vary depending on the type of sensor.

For each sensor a number of things will be displayed in the table.

ID	The number of the sensor in the list
Name	The name of the sensor (can be changed in the “Selected Sensor Settings” tab
Type	Local (connected to this machine) Network (connected to a network machine)
Color/Infrared/Depth/Bodies FPS	The actual framerate of the data that is streamed.

Note that if your machine can not cope with the amount of data streaming in framerate may drop. This can either mean your USB bandwidth is not sufficient (see chapter on that) or your CPU/GPU is not fast enough to handle the amount of data being generated by the sensors. You can try to connect some sensors to a networked machines to balance the load.

Num Bodies	Number of bodies detected and tracked by a sensor
Num Faces	Number of faces detected on the tracked bodies by a sensor
Num Markers	Number of markers detected (during marker-based alignment) by a sensor
Aligned	Depicts if a sensor has been aligned or not (note that only aligned sensors will contribute to the skeleton solver)
Lock	Locked sensors will not be moved during alignment operations
Pos XYZ	position/translation of the sensor in the shared coordinate system
Rot XYZ	rotation of the sensor in the shared coordinate system Note that you can manually adjust these positions/rotations on the “Selected Sensor Settings”, they will automatically be adjusted by alignment operations

When using networked sensors, you’ll have a few options to control the amount of bandwidth vs quality consumed:

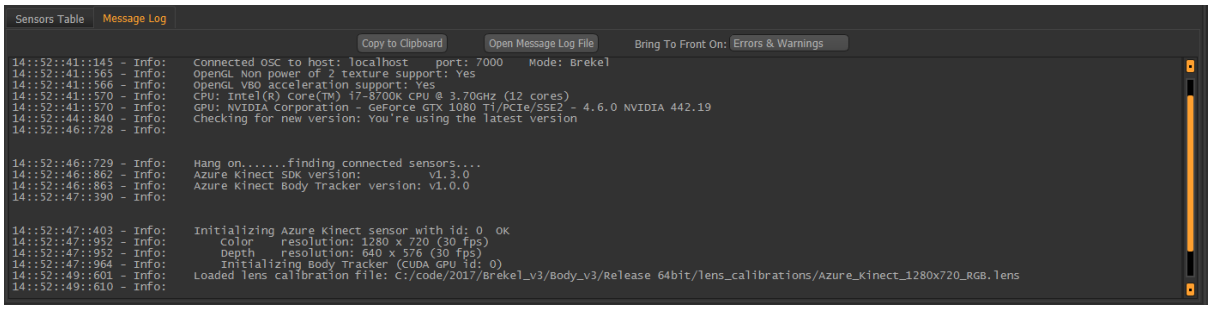
Quality	lower numbers will require less bandwidth but will look less detailed
FPS	lower numbers will stream fewer frames over the network to reduce bandwidth requirements

Note that this will only affect visual quality in the 3D viewport.

Skeleton data is always streamed and solved at full quality and framerate.

Recording BPC pointclouds will always happen at full quality and framerate as well.

MESSAGE LOG



The “Message Log” tab will display a history of all info, warning and error messages the application has reported since startup.

This will provide useful information on many operations as well as indicate if something went wrong.

The log will also be saved to the “Brekel” folder in your Windows Documents folder.

For example in: C:\Users\YourUsername\Documents\Brekel\ Brekel_Tester_v3.log

Copy to Clipboard

Copies the data to the Windows clipboard buffer so you can paste it into other applications.

Open Message Log File

Opens the message log file in the system’s default text editor.

Bring To Front On

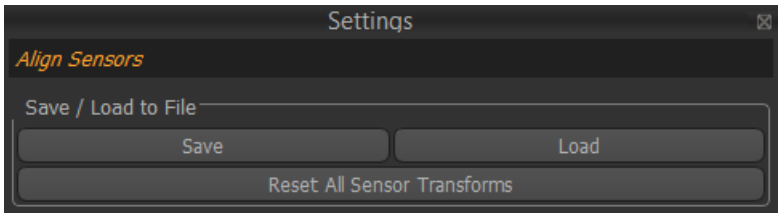
Sets if the message log window will automatically be brought to the front on errors, warnings or never.

SENSOR ALIGNMENT

The skeleton data fusion solver needs sensors to be aligned, you'll see the following message in the main 3D viewport if one or more sensors have not been aligned yet:

Sensor alignment calibration can be done in various ways and the result is that the sensor transforms are set and are no longer at the origin (0,0,0) of the world.

- Once calibration is done do not move your sensors, or you will have to do another calibration
- Your calibration will automatically save/load upon exit/start of the app
- You can tweak a selected sensor's transform on the "Selected Sensor" tab
- It is ok if calibration is not 100% precise, skeleton tracking is inherently noisy and the data fusion algorithms are designed to intelligently solve data



You can find all alignment options on the "Align Sensors" tab in the "Settings" Window.

Let's start with the basic alignment calibration options:

Save

Saves sensor alignment data for all the currently connected sensors to a text file.

Load

Loads a sensor alignment text file and for all applies the alignment data to all currently connected sensors it finds a match for. Note that when your sensors have physically moved, you'll have to perform a new alignment calibration.

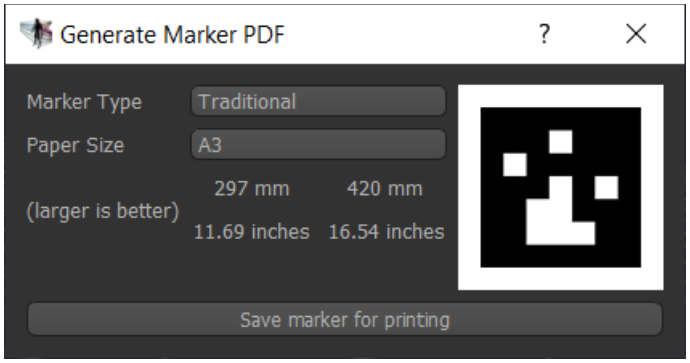
Reset All Sensor Transforms

Resets all sensors and removes all alignment calibration data.

2D PRINTED MARKERS

The most flexible way to calibrate your sensors is by using a 2D printed marker.

“Settings > Save Marker For Printing” from the top menu opens this to generate a PDF file you can print.



- You’ll want your print to be as be big as possible/practical (the more pixels seen by the sensor the more accurately it can calibrate itself)
- In most cases A3 is a good default size.
- It is important that you leave white borders around the marker when it’s printed, the PDF file already has these in place. Note that the exact size of these margins isn’t important.
- After printing you will need to attach the marker to a rigid surface so it cannot bend, cardboard or foamboard are great for this purpose.

It is important that during calibration the paper size (and marker length) in the GUI matches the physical size as it’s internally used for the initial guess of the marker size.

For best results you can measure the large black square on your printed marker and use “Custom Size” instead of the default paper sizes in the tracker settings, or use the “Auto Optimize Marker Length”.

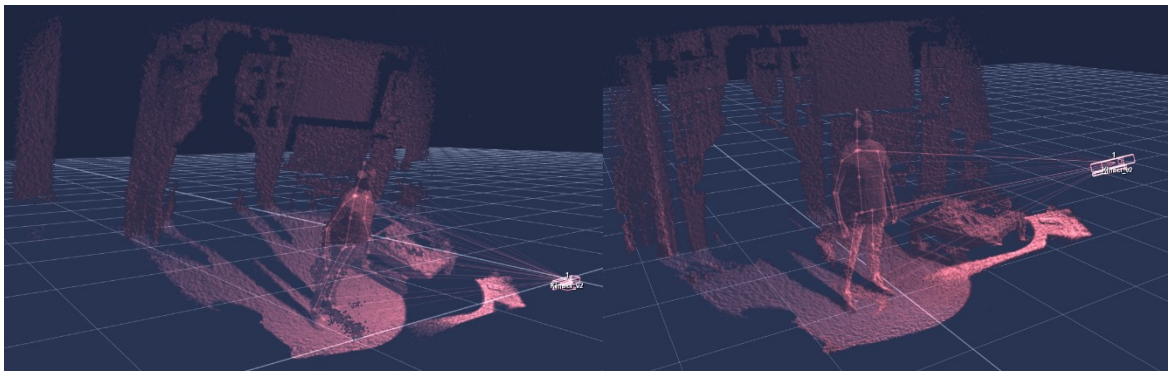
The “traditional” marker type is the default pattern but there is a “new” type which is a bit faster to detect, should in theory be a bit more robust during movement but is still experimental.

During calibration it is important to both **move & rotate** the marker around your capture volume **slowly & smoothly**, while making sure multiple sensors will be able to see it from different positions and orientations to give the calibration algorithm enough variation.

Note that (based on the original transform of the primary sensor) your result may not be in a correct final orientation and position based to the 3D grid/world in the viewport, until the floor level refinement functionality is used.

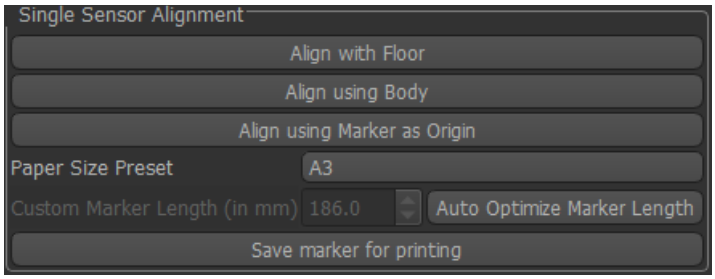
SINGLE SENSOR ALIGNMENT

When dealing with a single sensor setup all you need is the physical floor to match the floor in the 3D viewport. Adjusting for the physical sensor height from the floor and its rotation.



Not Aligned Aligned

When running a single sensor, you can align it in different ways:



Align with Floor

This will try to detect the floor and adjust the sensor's translation and rotation such that the floor is level and at y=0 in the 3D viewport. Note that when the floor cannot be seen this may give incorrect results.

Align using Body

This will ask the performer to stand in a T-pose and adjust the sensor's translation and rotation such that the tracked skeleton is centered, straight up and the feet are roughly at the floor height.

Align using Marker as Origin

This will align the sensor so a visible 2D marker is at the origin.

Please read the chapter “sensor alignment using 2d printed markers” below on more information on markers.

Paper Size

Size of the physically printed marker, the tracker needs this to accurately estimate where it is in 3D space.

Custom Marker Length

Measured length (in millimeters) of the large main black square of your printed marker, the tracker needs this to accurately estimate where it is in 3D space.

Auto Optimize Marker Length

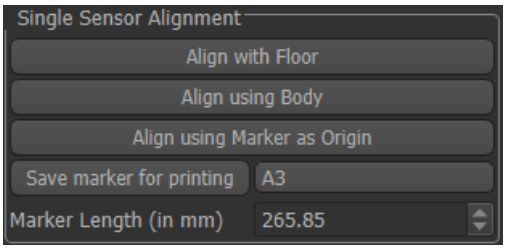
Automatically optimize the marker length (based on the hint you give it) using video and pointcloud data.

This can help getting the most accurate results

Save marker for printing

This opens this to generate a PDF file you can print. (same as “Settings > Save Marker For Printing” from the top menu)

When running a single sensor, you can align it in different ways:



Align with Floor

This will try to detect the floor and adjust the sensor's translation and rotation such that the floor is level and at y=0 in the 3D viewport. Note that when the floor cannot be seen this may give incorrect results

Align using Body

This will ask the performer to stand in a T-pose and adjust the sensor's translation and rotation such that the tracked skeleton is centered, straight up and the feet are roughly at the floor height.

Align using Marker as Origin

This will align the sensor so a visible 2D marker is at the origin.

Please read the chapter "sensor alignment using 2d printed markers" below on more information on markers.

Save marker for printing

Writes a PDF (in the specified paper format) to disk so you can print it.

Paper Size

Specifies the paper size for the PDF to be generated, also sets the initial marker length uses in the internal calibration code.

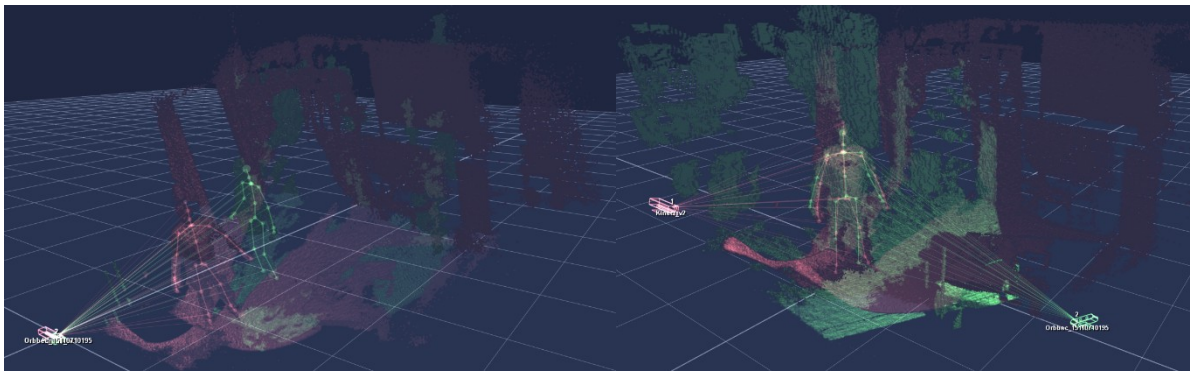
Marker Length

Measured length (in millimeters) of the black outer square from the printed marker.

Setting the paper size sets a rough estimate of this, measuring it yourself is best.

MULTI SENSOR ALIGNMENT

When dealing with multiple sensors (two in this example for clarity but you can of course use more) the sensors need to be aligned with the 3D world coordinate system but also be aligned to each other so their data overlaps correctly.



Not Aligned Aligned

Multi sensor alignment calibration can be performed in 3 ways:

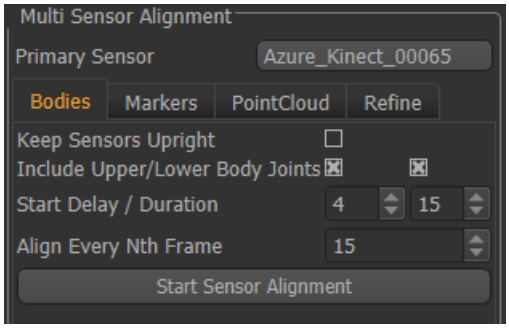
- Using tracked bodies
- Using 2D printed markers
- Using pointcloud data

Note that you can lock specific sensors, using the toggle in the Sensor Table, to prevent them from being adjusted, this also allows you to re-calibrate a single sensor if it has moved for example

SENSOR ALIGNMENT USING BODIES

The easiest, but also least accurate way, to align sensors is by using the body tracking features of the sensors and analyzing the overlap between multiple sensors that can see the same body.

You can find the options under the “Using Bodies” section on the “Align Sensors” tab.



During calibration:

- Make sure only a single person is in the capture volume
- The software will look at the raw skeletons from all the sensors and use these to align them
- During calibration the person should **turn around** in place and move around a bit through the volume
- It is best to raise one hand next to the head to ensure the left/right of the body are different

- The “Primary Sensor” is used as a basis to align other sensors
- The software will try to center the seen bodies around the origin and keep them upright

Primary Sensor

The primary sensor will be used as a basis and starting point for the calibration, other sensors will be aligned to this sensor. It is best to pick a sensor that the user will face at the start of each capture.

Keep Sensors Upright (advanced option)

This does an internal check during alignment to make sure sensors are not flipped upside down. May help for some setups but often is not needed.

Include Upper/Lower Body Joints (advanced option)

Selects which joints to use during calibration

Start Delay / Duration

Start Delay gives you a countdown to walk into the volume after starting the alignment calibration.
Duration will set a fixed amount of time after which the calibration will automatically be stopped. Setting it to 0 will run until manually stopped.
(both are in seconds)

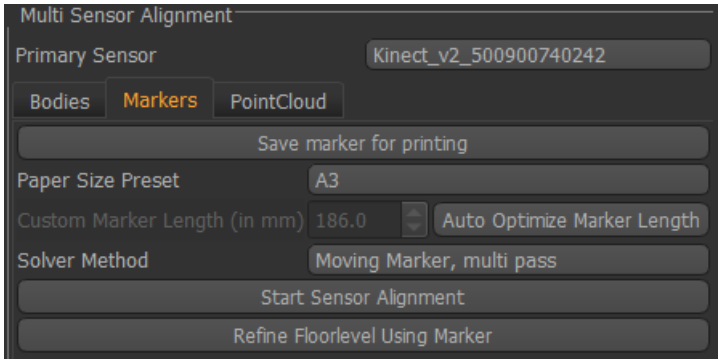
Align Every Nth Frame (advanced option)

Performs an alignment every specified number of frames. Doing this only every few frames will conserve CPU usage yet still give visual feedback during alignment in the 3D viewport.

Start/Stop Sensor Alignment

Starts/Stops the alignment calibration.
A full calibration (all sensors to all others using all collected frames) will be performed when alignment calibration is stopped.

MULTI SENSOR ALIGNMENT USING MARKERS



Save Marker for printing

This opens this to generate a PDF file you can print. (same as “Settings > Save Marker For Printing” from the top menu)

Paper Size

Size of the physically printed marker, the tracker needs this to accurately estimate where it is in 3D space.

Custom Marker Length

Measured length (in millimeters) of the large main black square of your printed marker, the tracker needs this to accurately estimate where it is in 3D space.

Auto Optimize Marker Length

Automatically optimize the marker length (based on the hint you give it) using video and pointcloud data.

This can help getting the most accurate results

Solver Method

"Static Marker" assumes a marker on the floor at the center of your volume where all sensors can see it.

"Moving Marker, single pass" assumes you slowly move and rotate the marker across the capture volume.

The solver will align sensors based on all frames where the marker was seen by the primary sensor and other sensors.

"Moving Marker, multi pass" assumes you slowly move and rotate the marker across the capture volume.

The solver will do an initial alignment based on all frames where the marker was seen by the primary sensor and other sensors (like the single pass method). And then refine it using all data from all frames of all sensors top get an optimal result with the least amount of numerical error.

Start Sensor Alignment

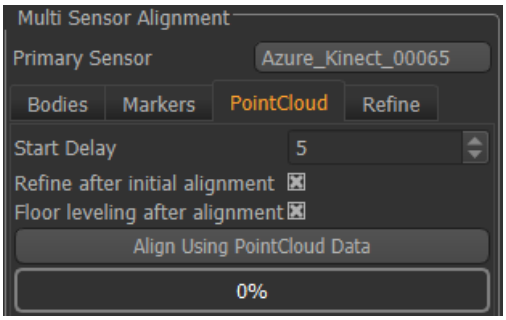
Starts/Stops the alignment calibration.
Note that the solver doesn't know yet where the floor is so while aligning sensors to each other they may still be rotated in regards to the floor

Refine Floorlevel Using Marker

Assumes a marker placed on the floor in the center of your volume where one or more sensors can see it.

All sensors will be adjusted (translation & rotation) so the marker is at the origin (0,0,0) of the virtual world and level with the floor.

SENSOR ALIGNMENT USING POINTCLOUD DATA



Besides using skeleton and marker data it is also possible to align sensors based on pointcloud data. This can be highly accurate but is also very dependent on the angle between sensors as it's highly dependent on overlapping points of neighboring sensors seeing the same parts of a person.

After starting the alignment the app will start a countdown so you have time to walk into the volume.

During alignment you should **stand still**

It helps to stand in an asymmetric pose for example by raising one arm like this:

Initial alignment will be guessed based on your primary sensor.

Note that this is highly dependent on pointcloud overlap and there is a small element of randomness so you may need to try again if alignment fails.

Start Delay

Start delay (in seconds), gives you some time to walk into the volume after starting alignment.

Refine after initial alignment

After estimating the initial (rough) alignment this will run additional refinement passes (same as manually using “Refine Using PointCloud” on the “Refine” tab.

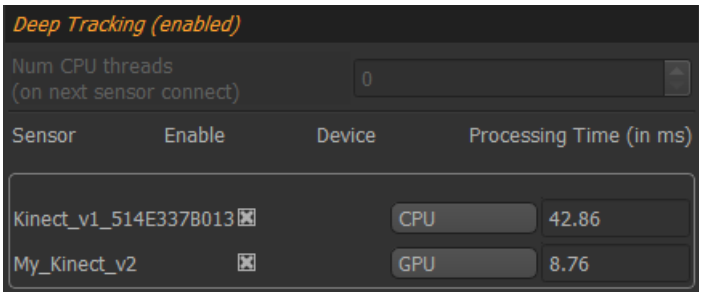
Floor leveling after alignment

After estimating the initial (rough) alignment this will try to detect the common floor plane between pointcloud data from all sensors and both level and center things based on that.

Align Using PointCloud data

Start/Stop alignment process.

DEEP BODY TRACKING



Besides the native body tracking provided by the sensors the Brekel app can run an additional Deep-Learning (AI) based tracker to provide additional joint & pose estimations to help improve body tracking results.

This deep tracking can also help resolve front/back ambiguity as some sensors may always assume a person is facing the sensor and the deep learning tracker can provide additional info on the front/back/left/right side of each joint for the skeleton solver.

Deep Tracking can be enabled/disabled per sensor (for some newer sensors like Azure Kinect and ZED2 which already use a deep learning based tracker there is no added benefit) and you can select to run it on the CPU or GPU.

CPU is generally slower and may not reach the full framerate (which the solver can cope with) but it should work on most machines.

GPU offloads things to NVIDIA Tensor Cores (on RTX cards) and will keep your CPU available for other tasks.

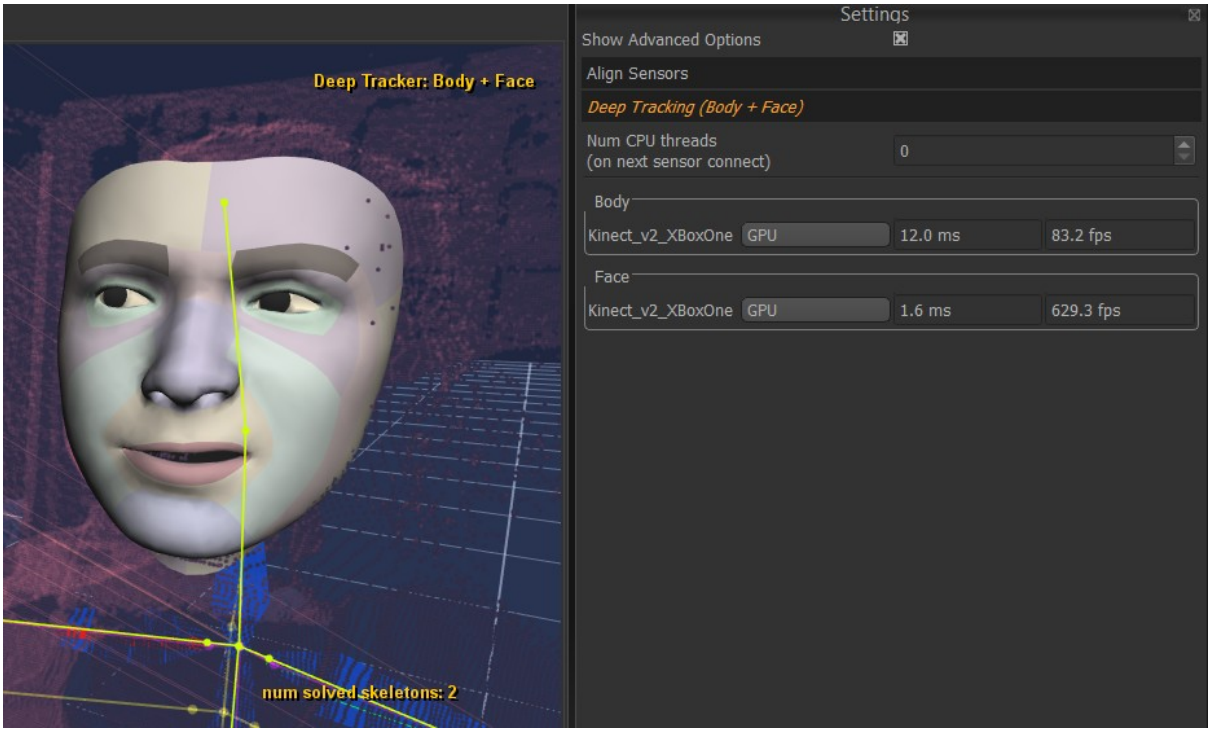
You can mix usage of CPU/GPU and also select Deep Tracking on one or multiple sensors to balance the computational load for your particular machine.

Note that for the GPU option to work you'll have to download the NVIDIA AR SDK.

If this is not installed on your system this button will appear on the Deep Tracking tab to allow you to automatically download/install this from the official NVIDIA webpage

DEEP FACE TRACKING

Deep Face Tracking uses Deep-Learning (AI) algorithms to detect faces and facial features from the Infrared or Color streams and outputs a mesh with a set of blendshapes for expressions and eye directions.

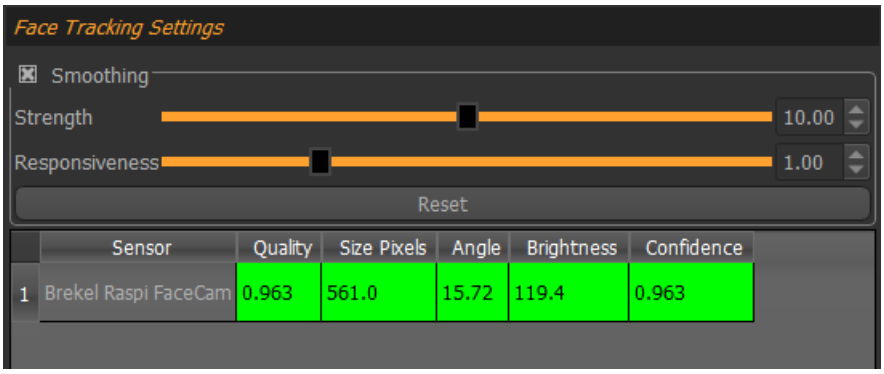


You will need an NVIDIA RTX card (with Tensor Cores) and the NVIDIA AR SDK installed (just like mentioned in the previous chapter on Deep Body Tracking), there is (currently) no CPU option available.

This can be enabled on a per-sensor and can be used with or without using deep body tracking.

Note that the face mesh is part of “solved bodies” so you will need to align your sensor(s) before it will show up in the 3D viewport.

FACE TRACKING SETTINGS



On the Face Tracking Settings tab you'll find settings to enable the Smoothing filters:

Smoothing Strength

Amount of smoothing to apply (more is smoother)

Smoothing Responsiveness (advanced setting)

Less is smoother but lags behind in time

You will find a table listing all the faces detected on all sensors with different metrics:

- Quality – overall quality ranging from 0.0 for low quality to 1.0 for best quality (this is based on the size, angle, brightness & confidence metrics)
- Size Pixels – the size of the face (in pixels) in the Color/Infrared stream, larger faces produce more accurate results, the fewer pixels the less accurate the facial features and expressions can be detected
- Angle – the angle towards the sensor, low values for faces looking straight at the sensor produces more accurate results, turning away from the sensor may occlude facial features for example
- Brightness – the average brightness of the pixels of the face, when the image is too dark the tracking results become worse
- Confidence – the internal overall confidence level of the tracker indicating how well it could find the facial features in the image, for example if the face is partially in shadow confidence will be lower since those features may be hard to identify.

The color will roughly indicate good (green) vs worse (red) metrics.

Tracking quality is directly dependent on how well faces can be seen.

The solver will automatically try to fuse the results of multiple sensors by utilizing the best quality metrics.

Depending on your sensor it can help to use “Advanced Sensor Settings” and enable the Color stream instead of the InfraRed stream and/or increase the resolution to provide more pixels to track.

Infrared provides an evenly lit face though which helps, when using the Color stream make sure the actor’s face is well lit.

Generally the closer you are to your sensor the better it can detect your face so in practice this will work best for upper-body scenarios and not so well if you are so far away that the full body can be seen (and the face is only a few tens of pixels high in the sensor image).

Face tracking is ONLY exported to FBX files as it relies on a face mesh with animated blendshapes.

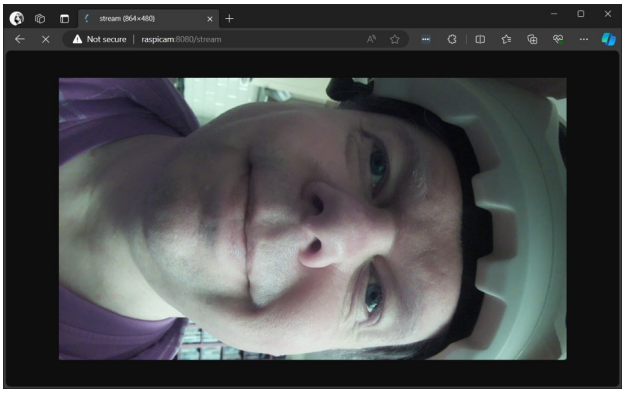
The following 53 shapes are tracked, eye shapes are also converted to angles and available on two transforms

browDown_L	browDown_R		
browInnerUp_L	browInnerUp_R	browOuterUp_L	browOuterUp_R
cheekPuff_L	cheekPuff_R	cheekSquint_L	cheekSquint_R
eyeBlink_L	eyeBlink_R		
eyeLookDown_L	eyeLookDown_R	eyeLookIn_L	eyeLookIn_R
eyeLookOut_L	eyeLookOut_R	eyeLookUp_L	eyeLookUp_R
eyeSquint_L	eyeSquint_R	eyeWide_L	eyeWide_R
jawForward	jawLeft	jawOpen	jawRight
mouthClose	mouthDimple_L	mouthDimple_R	
mouthFrown_L	mouthFrown_R	mouthFunnel	
mouthLowerDown_L	mouthLowerDown_R	mouthPress_L	mouthPress_R
mouthLeft	mouthPucker	mouthRight	
mouthRollLower	mouthRollUpper	mouthShrugLower	mouthShrugUpper
mouthSmile_L	mouthSmile_R	mouthStretch_L	mouthStretch_R
mouthUpperUp_L	mouthUpperUp_R	noseSneer_L	noseSneer_R

RASPBERRY PI FACE CAMERA

Tracking faces from the same camera as used for body tracking has the disadvantage that the face may be too small (far away), occluded or not turned towards the camera.

Another solution is to use a helmet mounted camera to ensure the face is always properly visible and well lit.



With off-the-shelf components like a Raspberry Pi Zero 2W, Camera Module 3, USB power bank and some 3D printed parts this can be built for roughly $125 - $150.

It can stream video at 40-60 fps over WiFi, integrates with Body v3's face tracking, can run for many hours on a small USB power bank and can be mounted to an off-the-shelf helmet (like a bicycle helmet for example)

You can find more detailed info on how to build one for yourself in the "RasPi FaceCam" PDF file.

3D print files for the camera case can be found on Printables and MakerWorld here::
https://www.printables.com/model/805075-rasberry-pi-zero-face-camera
https://makerworld.com/en/models/387990

3D print files for the helmet mounts can be found on Printables and MakerWorld here::
https://www.printables.com/model/823454-hmc-helmet-mounted-camera-mounting-solution
https://makerworld.com/en/models/400309

Scripts to download & install camera-streamer software for the Raspberry Pi can be found here:
https://github.com/Brekel/raspi-facecam

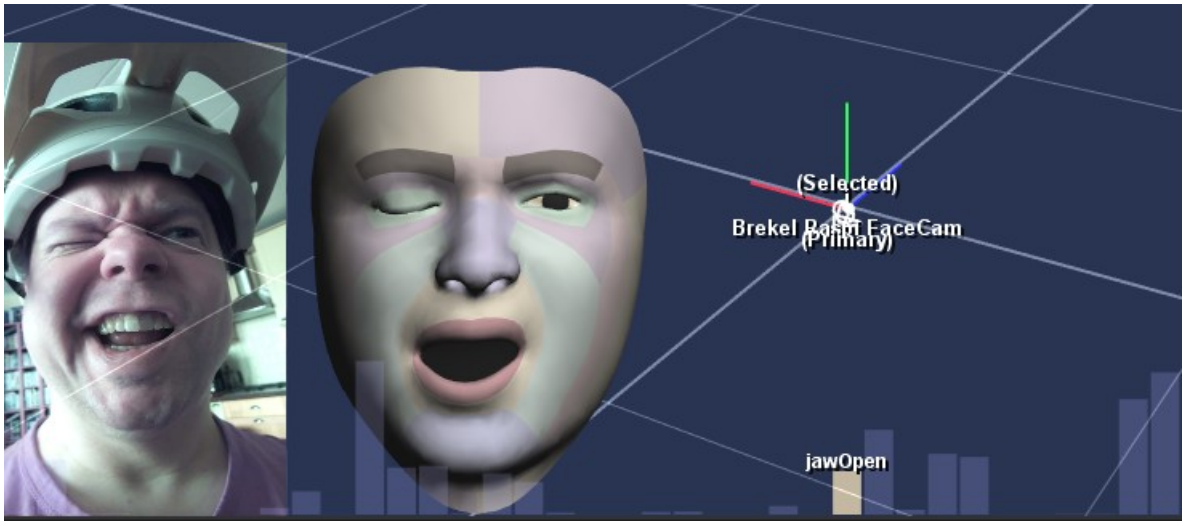
On the “Drawing Settings” tab on the left of the GUI you’ll find a “Raspi FaceCam” section.

Show for selected FaceCam

Shows the Video and/or Mesh in the 3D viewport only when a FaceCam sensor is selected

Show for first FaceCam

Always shows the Video and/or Mesh in the 3D viewport of the first FaceCam in the sensors list



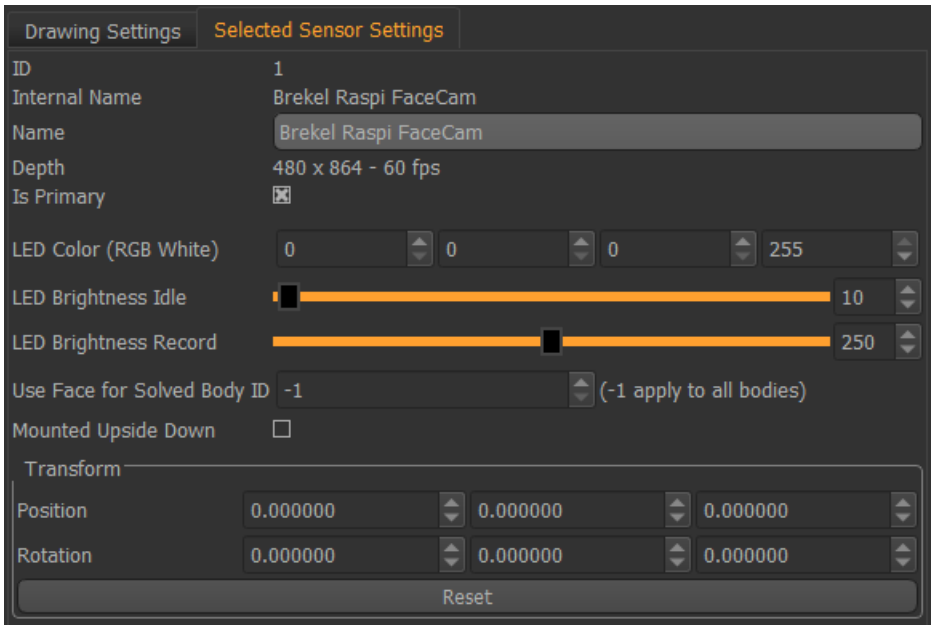
Show Video

Shows the video stream of the FaceCam in the bottom left corner of the 3D viewport

Show Mesh

Shows the tracked mesh of the FaceCam in the bottom left corner of the viewport

Raspi FaceCams have some unique settings.
When a FaceCam is selected in the Sensor Table you can find them in the “Selected Sensor Settings” panel:



LED Color (RGB White)
Specifies the color of the RGBW LEDs (if installed) in 8-bit 0-255 range.
It is usually best to keep the RGB LEDs disabled (0) and the White LED fully enabled (255).

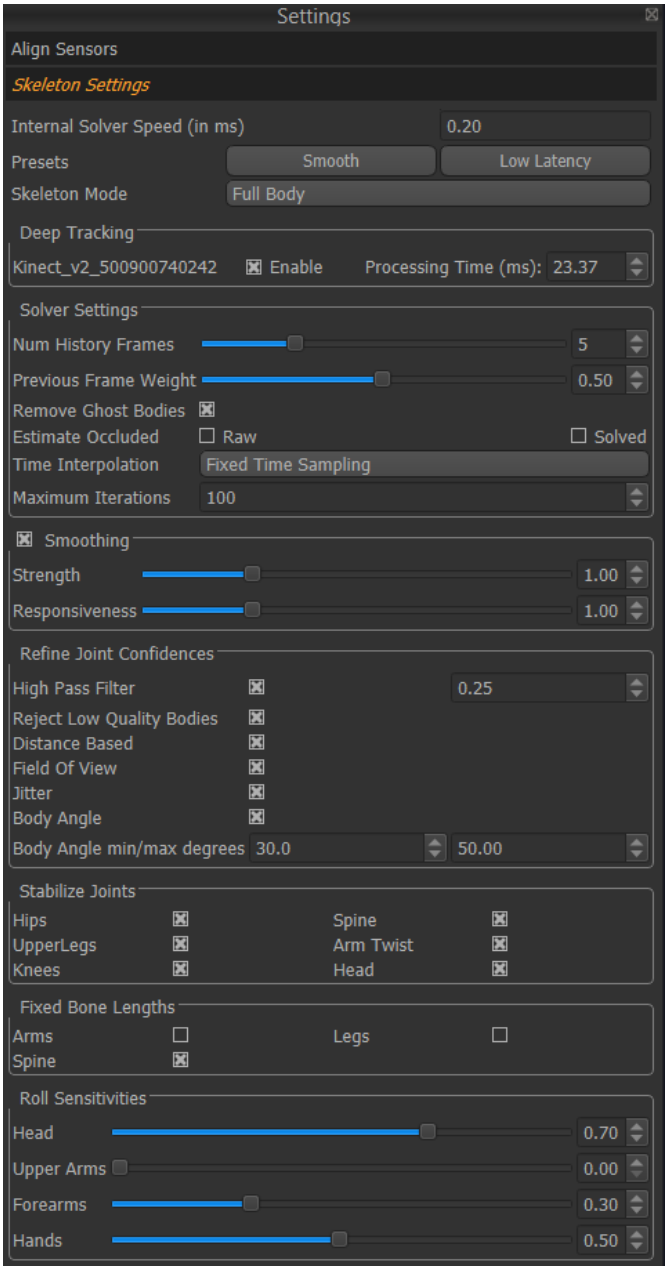
LED Brightness Idle
Specifies the brightness of the LEDs when not recording.
It can be convenient to keep this low so you can still see that the device is powered but not hinder the actor.

LED Brightness Recorded
Specifies the brightness of the LEDs during recording.
It is generally best to turn the LEDs on during recording as this provides more stable tracking results.
How bright to set them depends on the overall lighting in your room.

Use Face for Solved Body ID
The body solver can not automatically determine which FaceCam is worn by which actor
This setting allows you to specify which Solved Body ID to apply this FaceCam’s data to
(in case of multiple actors/FaceCams)

Mounted Upside Down
Check this box if your sensor is upside down, this will internally flip the image

SKELETON SETTINGS



Smooth Preset

Apply default settings that favor smoother results but with a bit more latency, good for recording.

Low Latency Preset

Apply default settings that favor low latency output at the cost of more noise.

Skeleton Mode

Sets if tracking output should be full body or just the upper body (with a fixed pose for the lower body bones.

Deep Tracking

When enabled (per-sensor) runs an additional deep-learning based tracker for additional tracking quality.

(CPU with AVX2 instruction support is required!)

The deep tracker is quite CPU intensive but runs asynchronously from other streams so running at a lower frame rate is fine.

Benefits are:

- improved overall quality

- better tracking when not facing the sensor

- more certainty to identify left/right and front/back of people

Num History Frames

The number of previous frames to consider by the data fusion solver.

Using several previous frames in the solve will help for 360 degree turns but tends to smooth the data more. A good starting point is 5. Azure Kinect will provide much more stable data regarding front/back/left/right detection and would generally need fewer history frames to provide 360 degree tracking.

Previous Frame Weight (advanced setting)

Weight of previous frames for the solver compared to the current frame.

Higher values provide smoother results but increase latency.

Lower values favor current frame more increasing noise but decreasing latency.

Remove Ghost Bodies (advanced setting)

Tries to identify and remove bodies in unrealistic poses that don't belong to the actual person(s) being tracked, so they don't contribute in the solve.

Estimate Occluded Raw

Tries to estimate occluded joints on raw skeletons based on previous and parent data.

Estimate Occluded Solved

Tries to estimate occluded joints on solved skeletons based on previous and parent data.

Time Interpolation

Tries to align time samples before feeding them to the solver.

- None: no interpolation
- Fixed Time Sampling: uses FBX resampling rate to interpolate to a fixed frame rate
- Variable Time Sampling: interpolates to a variable frame rate based on incoming data

Maximum Iterations

Maximum number of iterations the internal solver can use per frame

Smoothing Strength

Amount of smoothing to apply (more is smoother)

Smoothing Responsiveness (advanced setting)

Less is smoother but lags behind in time

High Pass Filter (advanced setting)

Only uses joints with a confidence higher than this

Reject Low Quality Bodies (advanced setting)

When enabled skips data from skeletons with low quality confidence scores while fusing data.

Distance Based (advanced setting)

When enabled adjusts weights based on distance to the sensor, joints very far away from a certain sensor have less influence for the data fusion solve.

Field of View (advanced setting)

When enabled adjusts weights based on where a body is in the sensor's field of view, joints close to the edges of the sensor reduce the influence for the data fusion solve.

Jitter (advanced setting)

When enabled adjusts weights based on how much jitter there is in a joints position, more jitter will reduce the influence for the data fusion solve.

Body Angle (advanced setting)

When enabled adjusts weights based on the body angle towards the sensor.

Body Angle min/max degrees (advanced setting)

Range of angle of body towards sensor.

Angles above the maximum threshold will get a reduced influence for the data fusion solve.

Angles below the minimum threshold will keep a 100% influence, values in between are linearly interpolated.

Stabilize Joints (advances setting)

Tries to stabilize the selected joints internally before using them in the data fusion solve.

Fixed Bone Lengths (advanced setting)

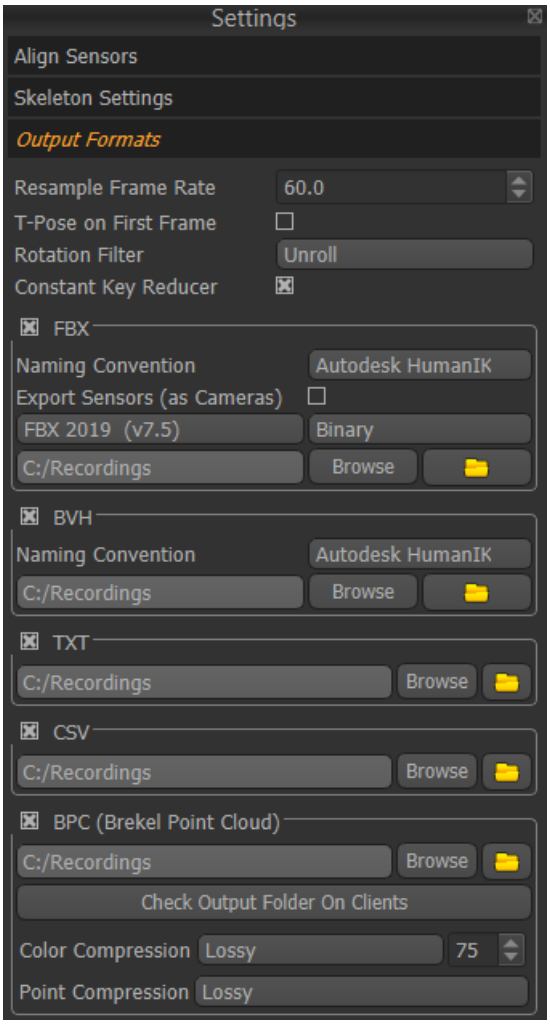
Tries to keep a fixed bone length during the solve, note that using this for legs may produce sliding feet as the ankles may no longer be able to reach their optimal position.

Retargeting functionality in your favorite 3D application may be able to do a more accurate job when doing this during mapping to your 3D character. (which is why arms/legs are disabled by default here)

Roll Sensitivities (advanced setting)

Sensitivity for roll rotations (rotation around the axis of the bone), lower percentages will stiffen the bone more.

3D OUTPUT FORMATS SETTINGS



Resample Frame Rate

Resample all animation curves to this frame rate, when this is set to 0 no resampling will be applied.

T-Pose on First Frame

Inserts a T-Pose on the first frame, note that in general it's a better idea to record a T-Pose with your actor on at least one take instead of relying on a mathematically generated one to initialize your retargeting.

Rotation Filter

Filter to apply on rotation animation curves to prevent Euler gimble flips.

Constant Key Reducer

When enabled animation curves with static values will be emptied to save space and reduce clutter.

FBX

Records data to the widely adopted FBX file format using Autodesk's official FBX SDK.

This should be supported by pretty much all 3D animation applications and game engines in existence.

Naming Convention

Joint naming convention to use, bone structure and hierarchy will always remain the same.

These are stored in text files in the "namingConventions" in the application's installation folder.

You can change/copy/rename/remove these files yourself, the app will read the list of available files at startup.

Export Sensors (as Cameras)

Export each sensor as a camera in the FBX file.

FBX Version

Depicts which version of the FBX file specification to use, note that older 3D animation applications may only support older FBX file versions.

FBX Type

Type of FBX file content, Binary or ASCII.

Folder

Which folder to save the FBX files in.

Browse

Brings up a window to browse for the folder to save the FBX files in.

Folder Icon

Opens the specified folder.

BVH (BIOVISION HIERARCHY)

Note that this is a 20-year-old file format and depending on how well it is supported and implemented in your favorite 3D application quality can vary.

In most cases you'll be better of using the modern FBX file format.

Naming Convention

Joint naming convention to use, bone structure and hierarchy will always remain the same.

These are stored in text files in the "namingConventions" in the application's installation folder.

You can change/copy/rename/remove these files yourself, the app will read the list of available files at startup.

Folder

Which folder to save the BVH files in.

Browse

Brings up a window to browse for the folder to save the BVH files in.

Folder Icon

Opens the specified folder.

TXT (TEXT)

Records data to a text file, which can be useful if you want to read the data in your own code or scripts.

Timestamps are in seconds.

Confidences range from 0 – 1, higher values mean the tracker was more confident on the joint's location.

All values are in local coordinates, so in reference to the joint's parent.

Positions are in meters.

Rotations are in Euler angles and in degrees (not radians).

Folder

Which folder to save the TXT files in.

Browse

Brings up a window to browse for the folder to save the TXT files in.

Folder Icon

Opens the specified folder.

CSV (COMMA SEPARATED VALUES)

Records data to a CSV file, which can be useful if you want to load data into a spreadsheet.

Timestamps are in seconds.

Confidences range from 0 – 1, higher values mean the tracker was more confident on the joint's location.

All values are in local coordinates, so in reference to the joint's parent.

Positions are in meters.

Rotations are in Euler angles (not radians).

Folder

Which folder to save the CSV files in.

Browse

Brings up a window to browse for the folder to save the CSV files in.

Folder Icon

Opens the specified folder.

BPC (BREKEL POINTCLOUD)

Records data (pointclouds, audio, raw skeletons and markers) to BPC files which can be used in Brekel PointCloud to export to meshes, particles, image and video formats.

Folder

Which folder to save the BPC files in.

Browse

Brings up a window to browse for the folder to save the BPC files in.

Folder Icon

Opens the specified folder.

Check Output Folder On Clients

Sends a signal to networked clients to check if the specified BPC output folder exists

(they will create the folder if needed)

Note that if your output path is “C:/Recordings” each network client will save to it’s own local “C:”drive, this is usually preferable over saving to a network drive to prevent frame drops.

Color Compression

Lossless compression will produce much larger files but will not drop any quality.

Lossy compression will produce smaller files but reduces quality a bit, you can use the percentage setting to depict the quality level.

0% = lowest quality and smallest file size

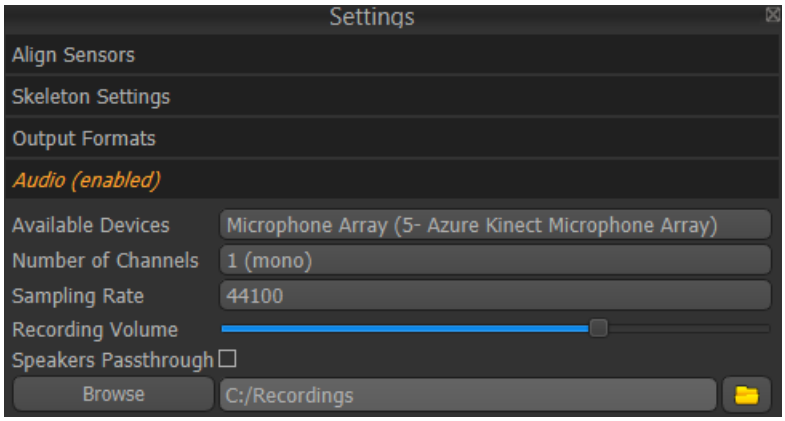
100% = highest quality at larger file sizes

Point Compression

Lossless compression will produce much larger files but will not drop any quality.

Lossless will produce smaller files at slightly reduced quality.

AUDIO SETTINGS



On this panel you can set audio related settings, note that audio is always recorded in the WAV file format.

Available Devices

Lists all detected audio devices (from sensors and other sources) from your system and selects which one to use for audio recording. Select “None” (first option in the list) to not record audio.

Number of Channels

Select to record mono or stereo audio.

Sampling Rate

Select which sampling rate to use, higher is better but produces larger file sizes.

Recording Volume

Sets the volume used for audio recording.

Speakers Passthrough

When enabled passes the sound from the input device through to your speakers.

Browse

Brings up a window to browse for the folder to save the audio files in.

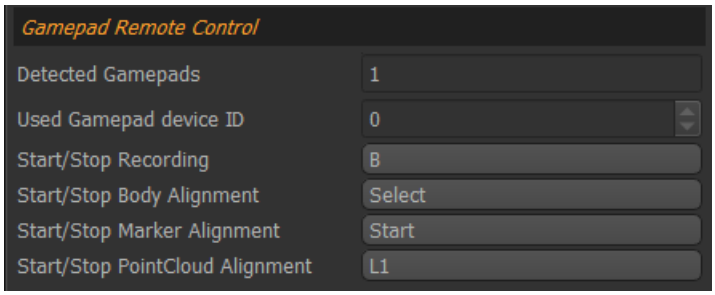
Folder

Which folder to save the audio files in.

Folder Icon

Opens the specified folder.

GAMEPAD REMOTE CONTROL



If you own a gamepad, for instance a wireless or wired Xbox controller, it can be used as a remote control for certain features in the app.

Any device that shows up as a gamepad device in Windows should work.

For example it can be handy to start/stop recording or alignment functions using a wireless gamepad while standing in the capture volume.

Detected Gamepads

Gamepads are automatically recognized when they connect/disconnect, this lists how many devices were detected by Windows.

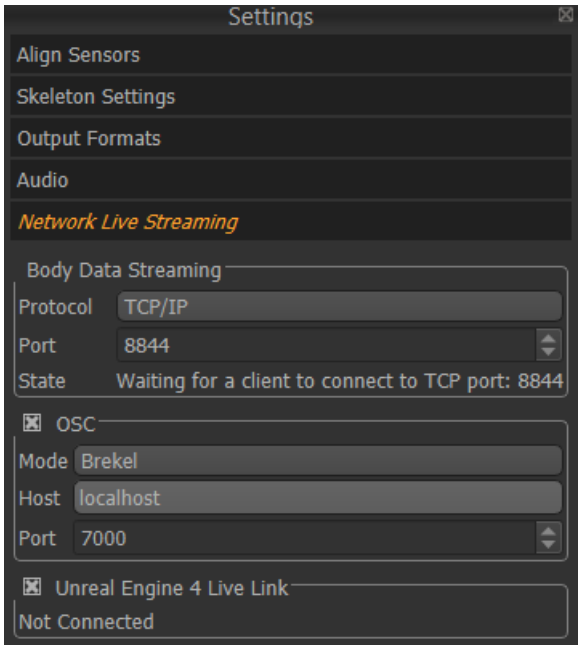
Used Gamepad Device ID

In case you have multiple gamepads connected to your system this settings selects which one is used.

Start/Stop

Available standard gamepad buttons can be connected to actions to start/stop various options in the software here. A particular button can only be connected to a single action at a time.

NETWORK LIVE STREAMING SETTINGS



Brekel Body v3 supports multiple ways to live stream data out over network ports to applications running on the same machine or on other machines on your network.

The first section “Body Data Streaming” refers to streaming into Unity, MotionBuilder or your own code using TCP/UDP ports. The format of the data packets is described in the chapter “Live Streaming Protocol” below.

The OSC section refers to the “Open Sound Control” protocol, which is an opensource UDP based protocol, you will find more information on it in the similarly labeled chapter below.

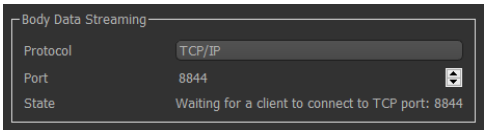
LIVE STREAMING INTO UNITY

You can find the Unity example script/scene for streaming live network data in the “Unity3D” folder inside the Brekel installation folder, usually something like here:

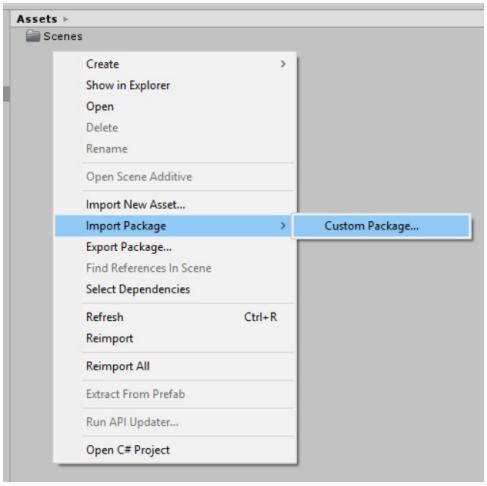
C:\Program Files\Brekel Body 3 x64\ Unity3D

To use it:

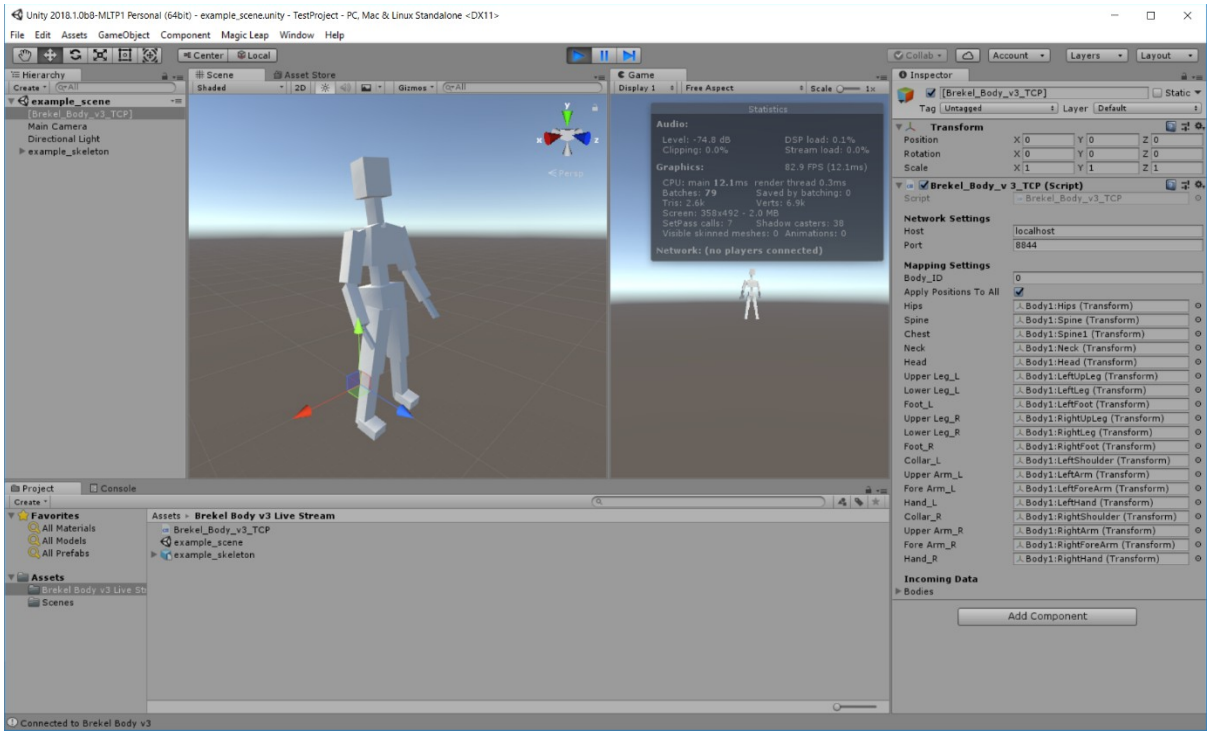
- Make sure in the Brekel Body v3 "Network Live Streaming" tab "Protocol" is set to "TCP/IP".



- Make sure Brekel Body v3 is running on the same machine as Unity (at least for a first test).
- Make sure your security software (firewall) isn’t block this port/protocol.
- Import "Brekel Body v3 Live Stream.unitypackage" into by right clicking in the Asset window.



- Or copy the assets from "Brekel Body v3 Live Stream.zip" into your project.
- Open "example_scene.unity".
- Hit Play.
- This should work for pretty much any Unity version on any platform as it uses standard C# functions to receive the network data.



Some pointers as to how this works:

- Note that the "Brekel_Body_v3_TCP.cs" script is added to the "[Brekel_Body_v3_TCP]" object (could be any object).
- "Host" on this script points to the machine name or IP of the machine running Brekel Body v3 (localhost if all runs on one machine).
- "Port" should match the port in the "Network" tab in Brekel Body v3.
- "Body ID" selects which body from the network stream is used, in case there are more than one.
- "Apply Positions To All".
- When turned ON this will apply positions (and rotations) to all transforms.
    - This can cause stretching of body parts if your actor/character proportions differ.
- When turned OFF this will apply positions & rotations to the root (hips) and rotations only the other joints.
    - The skeleton will not be stretched but if there are big proportional differences this can cause the feet to slide.

Some pointers to the script

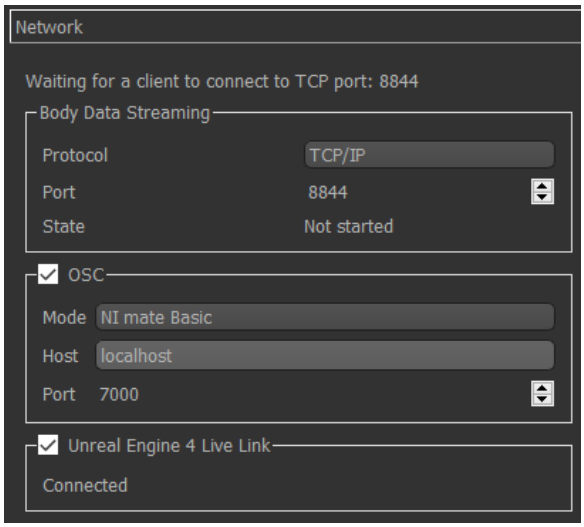
- All network functionality is encapsulated in the Connect() Disconnect() and FetchFrame() functions and generally wouldn't need to be touched.
- Fetching data from the network runs in a separate thread as to not slow down Unity.
- This is only an example script showing how to get live streaming data into your scene.
- A better way to 'retarget' data to your own character rig may be to use inverse kinematics for example.

LIVE STREAMING INTO UNREAL ENGINE

Live data can be streamed into the Unreal Engine using its Live Link feature.

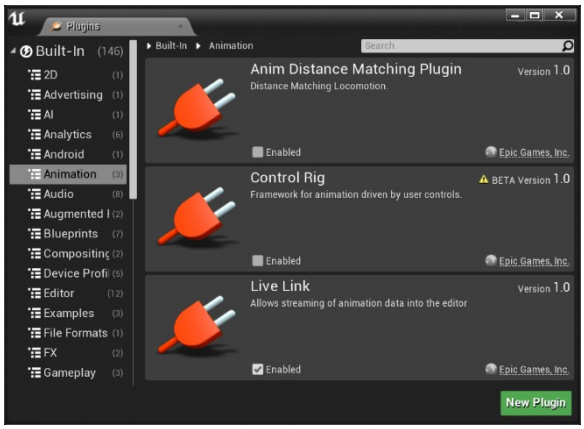
Note that the screenshots below are taken with UE4 but the same workflow is valid for UE5 (some things look slightly different)

In the Brekel app make sure "Unreal Engine Live Link" is enabled on the "Network" panel



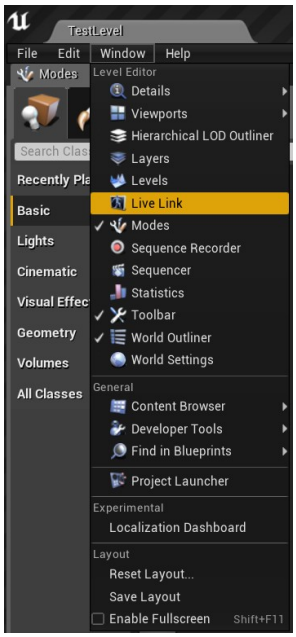
First make sure you enable this feature in the UE engine:

- Click Edit > Plugins
- Enable "Live Link" under "Animation"



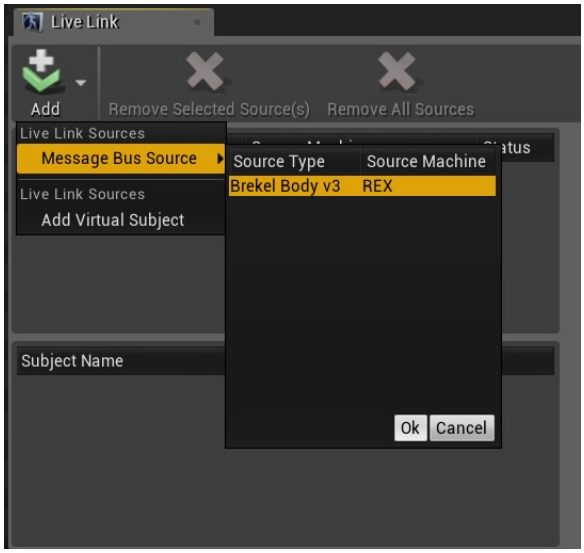
For UE4.x click Window > Live Link to open the Live Link Window

For UE5.x this can be found under Window > Virtual Production > Live Link

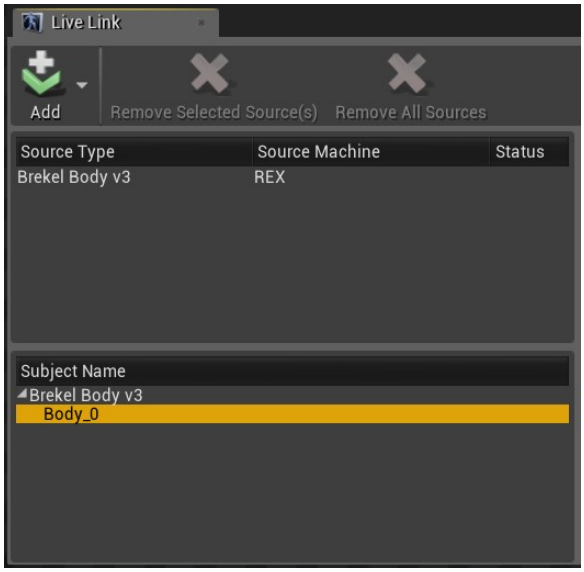


In Live Link Window

- click Add > Message Bus Source
- "Brekel Body v3" should be listed under Source Type/Machine, click it and hit Ok



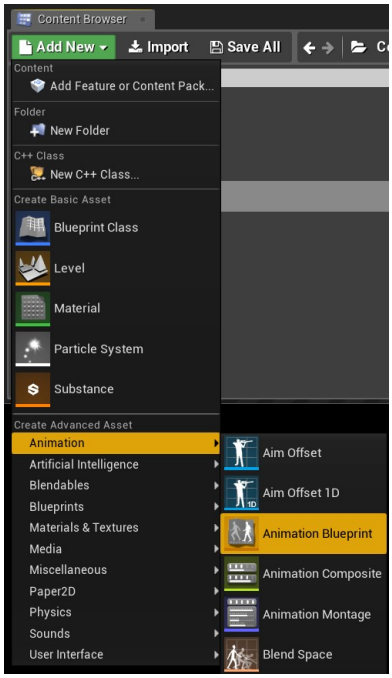
- Any visible and tracked bodies should be listed under "Subject Name"
- Take note of the body name you want to use ("Body_0" most probably)



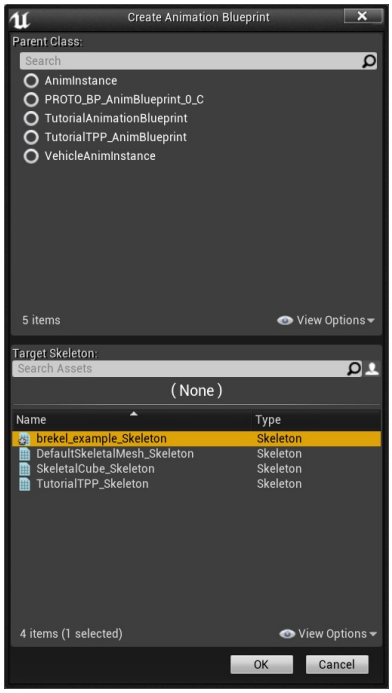
In the Content Browser:

- Import "brekel_example.fbx"
- You can find this file in the “UnrealEngine4” folder inside the Brekel installation folder, usually something like here:
- C:\Program Files\Brekel Body 3 x64\UnrealEngine

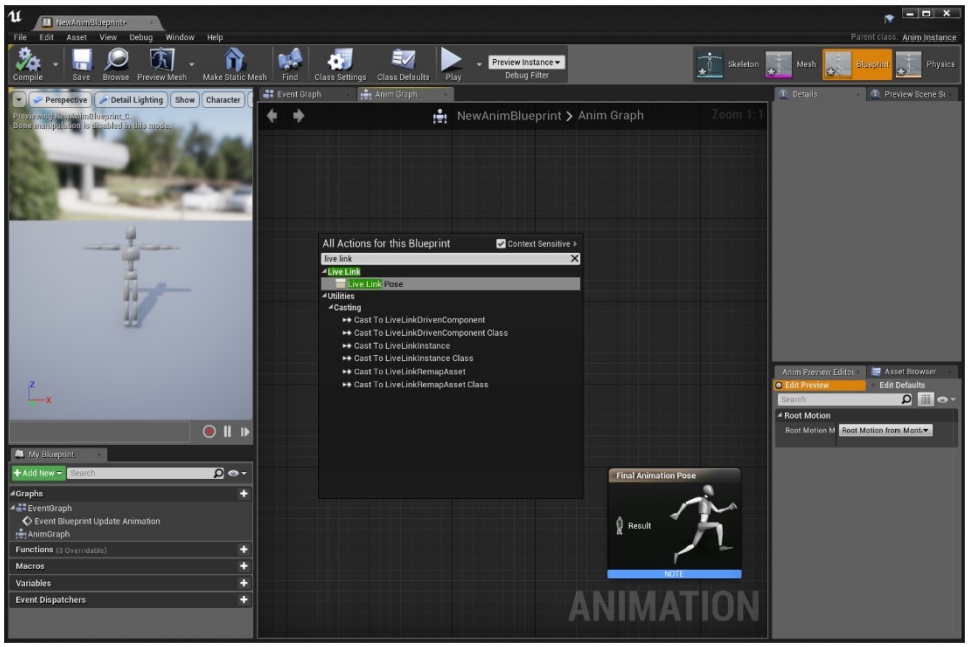
- Add New - Animation BluePrint



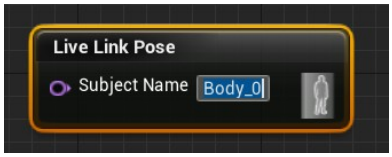
- Pick “brekel_example_Skeleton” from the list



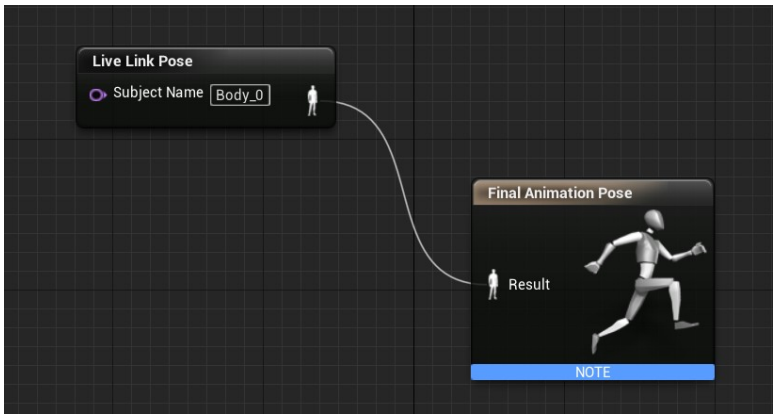
- Open the newly created blueprint by double clicking on it
- Right click in empty blueprint area
- Create a new "Live Link Pose" node



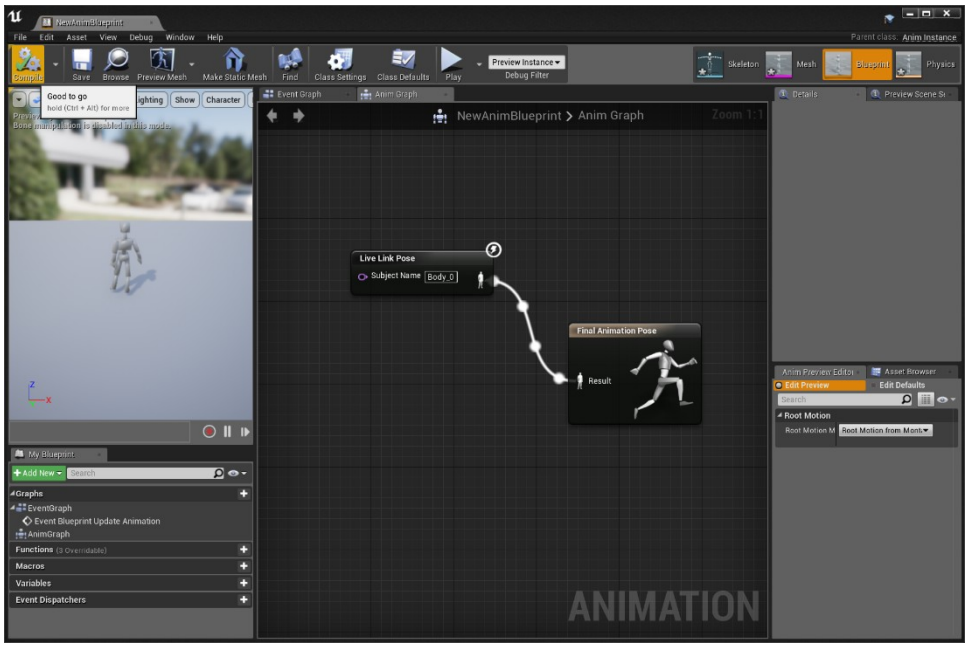
- Change Subject Name - None into the body you want to use ("Body_0" most probably)



- Connect Live Link Pose node to Final Animation Pose node

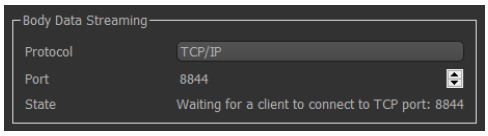


- Hit save and compile
    - You should now see a live data connection between the nodes
    - You should now see live data in the 3D window



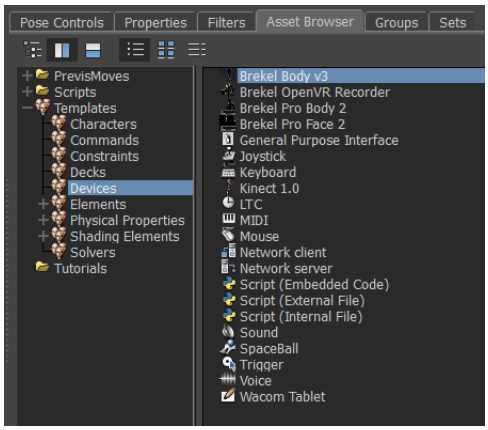
LIVE STREAMING INTO MOTIONBUILDER

- Make sure in the Brekel Body v3 "Network Live Streaming" tab "Protocol" is set to "TCP/IP".



- Make sure Brekel Body v3 is running on the same machine as Unity (at least for a first test).

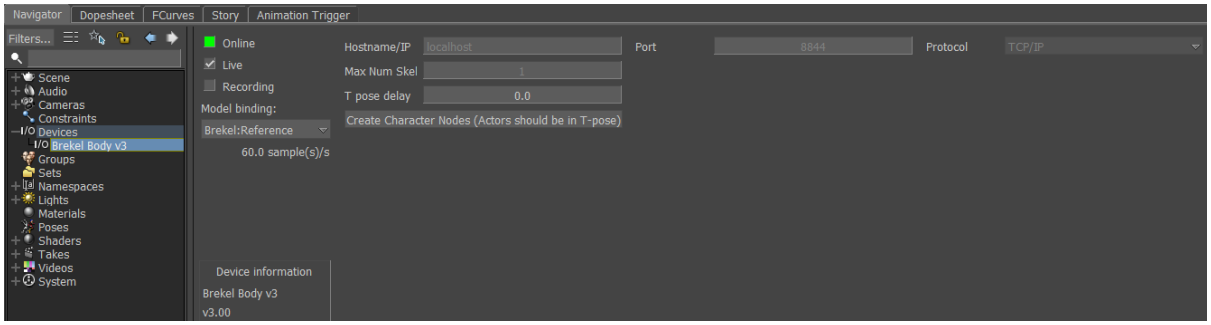
- Drag a Brekel Body v3 device from the Devices folder in the Asset Browser into your scene (the Brekel installer should have autodetected which MotionBuilder versions were installed on your system and has copied the appropriate plugins to the required folders)



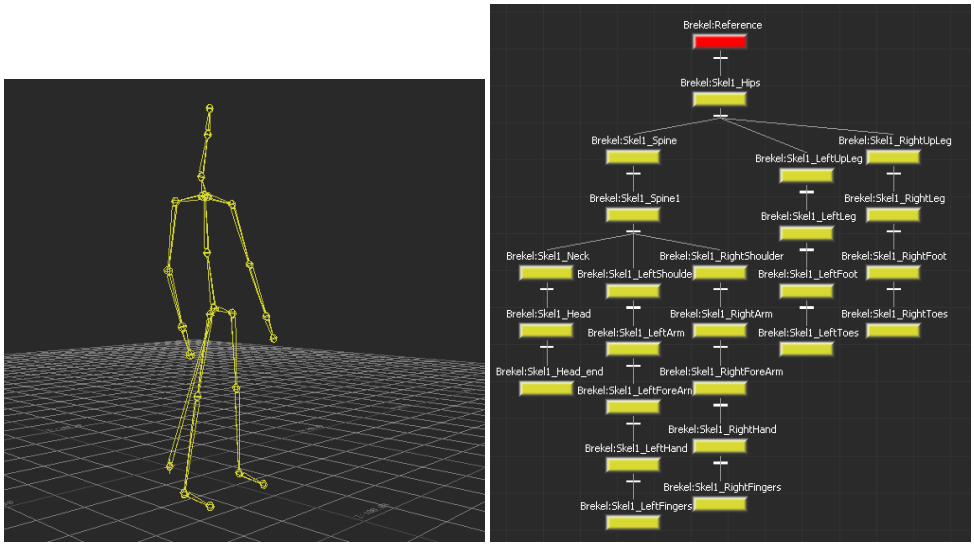
- In the GUI for the device make sure the Hostname/IP points to the machine running the Brekel application

  (Note: if both applications are running on the same machine you can simply use the default Hostname/IP of “localhost” or “127.0.0.1”)

- Make sure Port and Protocol match in both Brekel and MotionBuilder apps.
- Make sure your security software (firewall) isn’t block this port/protocol.



- Set the “Max Num Skel” setting to the number of skeletons you want to create/stream
- Toggle the “Online” button to turn the device on
- If it doesn’t turn green go back to the steps above
- Toggle the “Live” button to start receiving data
- Under “Model binding:” hit the “None” option and then “Create”, this will create a model hierarchy in your scene containing the skeletons (depending on how many subjects were seen at that time)



- You should now see one or more moving skeletons in your 3D viewport.

LIVE STREAMING PROTOCOL

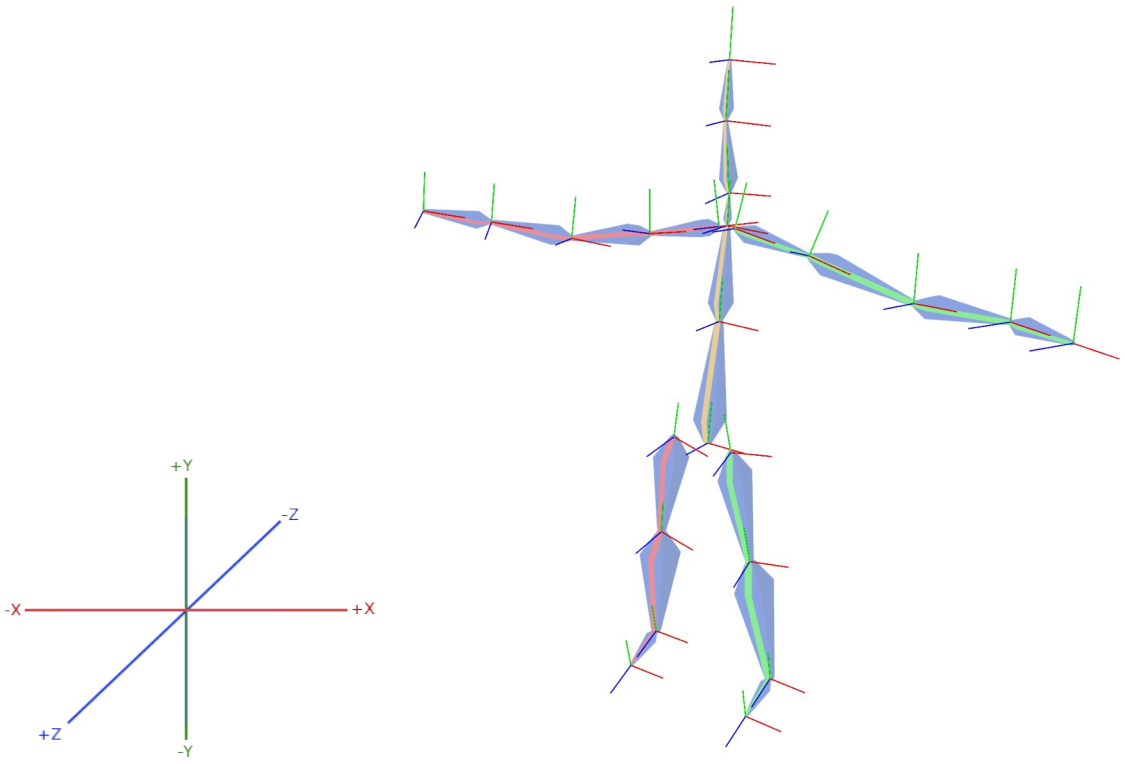
Network packets can be sent through TCP or UDP, on a user definable port (default 8844)

Note that the free trial version does not include network streaming for security reasons (only the retail and full evaluation version do)

The contents of a network package is as follows:

```
string    BRKL_S
int       num_joints
int       num_bodies
foreach body in num_bodies
{
        float     timestamp
        foreach joint in num_joints
        {
                float     confidence
                float     position X
                float     position Y
                float     position Z
                float     rotation X
                float     rotation Y
                float     rotation Z
        }
}
string    BRKL_E
```

- Each packet starts with “BRKL_S” and ends with “BRKL_E”, it is advised when receiving/processing packets in your own code you check for these string to identify valid packets.
- Note that these start/end strings contain an extra null termination character so they are each 7 bytes long (not 6)
- All ints and floats are 4 bytes each (in little-endian byte order)
- The number of joints is fixed, the order is listed below (keep in mind this could potentially change between app versions, in that case this manual will be updated)
- The number of bodies can vary per frame, depending on how many people were tracked
- The timestamp is in seconds, internally these originate from your system clock
- The joint confidences range from 0.0 – 1.0
    - Higher values depict higher accuracy
- Position x,y,z depicts the joint’s position in meters
- Rotation x,y,z depicts the joint’s rotation in Euler angles and degrees (not radians)
- All coordinates are in local space meaning they are in relation to their parent (see hierarchy below)
- When a body is in T-pose all axis align with the global/world axis which is:
    - Positive X axis to screen right
    - Positive Y axis pointing up
    - Positive Z axis pointing out of the screen
    - This is the same as OpenGL and apps like Maya/MotionBuilder for example
    - This is also described as a right-handed, Y-up coordinate system



Joint Name	Parent
Waist (aka Hips)	None (coordinates are always in global/world space)
Spine	Waist
Chest	Spine
Neck	Chest
Head	Neck
Head Tip	Head
Upper Leg Left (aka Hips Left)	Waist
Lower Leg Left (aka Knee)	Upper Leg Left
Foot Left (aka Ankle)	Lower Leg Left
Toes Left	Foot Left
Upper Leg Right (aka Hips Right)	Waist
Lower Leg Right	Upper Leg Right
Foot Right	Lower Leg Right
Toes Right	Foot Right
Collar Left (just left of Chest)	Chest
Upper Arm Left (aka Shoulder)	Collar Left
Forearm Left (aka Elbow)	Upper Arm Left
Hand Left (aka Wrist)	Forearm Left
Fingertips Left	Hand Left
Collar Right (just right of Chest)	Chest
Upper Arm Fingertips Right (aka Shoulder)	Collar Right
Forearm Fingertips Le Right ft (aka Elbow)	Upper Arm Right
Hand Fingertips Right (aka Wrist)	Forearm Right
Fingertips Right	Hand Right

OSC (OPEN SOUND CONTROL)

Open Sound Control (OSC) is a protocol for networking sound synthesizers, computers, and other multimedia devices for purposes such as musical performance or show control. OSC's advantages include interoperability, accuracy, flexibility and enhanced organization and documentation.

Brekel Body v3 can send OSC messages in various formats, including emulating other programs:

- Brekel
- Synapse
- OSCeleton
- NI mate (should work with their Blender / Cinema4D / Maya live streaming scripts)

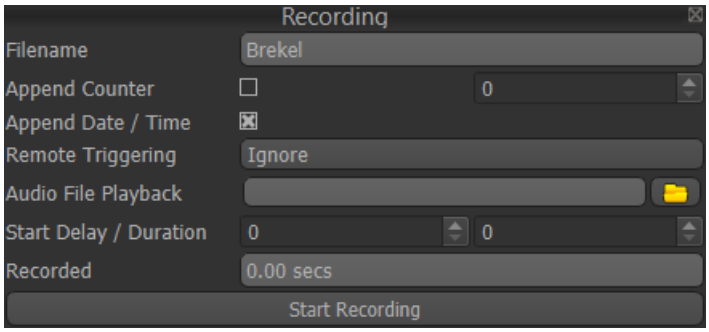
BREKEL OSC MESSAGES:

```
Address pattern: /<jointName>_joint_global              (see list below for possible jointNames)
int:    ID of the body this belongs to
float:  timestamp
float:  confidence      0.0 not confident - 1.0 very confident
float:  X position      in global space (relative to sensor)
float:  Y position      in global space (relative to sensor)
float:  Z position      in global space (relative to sensor)
float:  X rotation      in global space (relative to sensor) and in euler angles
float:  Y rotation      in global space (relative to sensor) and in euler angles
float:  Z rotation      in global space (relative to sensor) and in euler angles

Address pattern: /<jointName>_joint_local               (see list below for possible jointNames)
int:    ID of the body this belongs to
float:  timestamp
float:  confidence      0.0 not confident - 1.0 very confident
float:  X position      in local space (relative to parent joint)
float:  Y position      in local space (relative to parent joint)
float:  Z position      in local space (relative to parent joint)
float:  X rotation      in local space (relative to parent joint) and in euler angles
float:  Y rotation      in local space (relative to parent joint) and in euler angles
float:  Z rotation      in local space (relative to parent joint) and in euler angles

Available joint names for <jointName>
waist
spine
chest
neck
head
head_tip
upperLeg_L
lowerLeg_L
foot_L
toes_L
upperLeg_R
lowerLeg_R
foot_R
toes_R
collar_L
upperArm_L
foreArm_L
hand_L
collar_R
upperArm_R
foreArm_R
hand_R
```

RECORDING SETTINGS



Filename

Output filename to use for recording.

Note that counters and date/time can be appended to this (see below) to make sure filenames are unique.

Files will be recorded to the folders set on each file format in the “3D Output Formats” and “Audio” tabs.

Append Counter

When enabled adds a counter to the end of the filename to make then unique and prevent overwriting older files. You can adjust the current counter manually using the setting.

Append Date / Time

When enabled adds the current date and time to the end of the filename to make then unique and prevent overwriting older files.

Remote Triggering

Allows synchronized recording across multiple Brekel applications.

One application can be in Primary mode, all others in Secondary or Ignore mode.

The Primary application will send a signal when recording is started and stopped so all applications start/stop at the same time and are using matching filenames.

Note that this works across multiple apps on the same machine and even across multiple machines on the same network.

Make sure your firewall is not blocking port 8880-8890.

Audio File Playback

Plays back an audio file to during recording so your actor can use it for timing reference (in WAV file format)

Start Delay / Duration

Start Delay will postpone the start of the recording for the set number of seconds after pressing the “Start Recording” button. A countdown will appear in the viewport and audio beeps will play for reference. A setting of 0 will disable this and start recording immediately after pushing the “Start Recording” button.

Duration sets a fixed length (in seconds) for the recording, after reaching it the recording will automatically stop. A setting of 0 will disable this.

Recorded

Shows the number of seconds of the current (or last) recording.

Start Recording

Starts the recording, hit the button again to stop recording.