MANOMOTION

SDK PRO v_1.4.9

Technical Documentation

Table of contents

Table of contents

Introduction
- ManoMotion SDK PRO
- What is new? - 1.4.9 Patch notes
- Prerequisites
- Glossary

SDK Pro
- HandInfo
  - Hand Tracking (TrackingInfo)
    - Finger Info (BETA)
    - Wrist Info (BETA)
    - Contour Points (BETA)
    - Amount of contour points
    - Bounding Box
    - Palm Center
    - Depth Estimation
    - Skeleton Tracking
    - Joint Orientation
  - Gesture Analysis (GestureInfo)
    - ManoClass
    - Continuous Gestures
    - Trigger Gestures
    - Swipes
    - States
    - Handside
    - Left or Right hand
  - Warnings
  - Session (SDK settings)
    - Flag
    - Orientation
      - Front facing camera orientation
    - Gesture smoothing
    - Tracking smoothing
    - AddOn

- Features management
  - Skeleton
  - Gesture
  - Fast Mode
  - Finger Information
  - Wrist Information
  - Contour
- Deprecated Features
  - Deprecated Hand Tracking Features
  - Deprecated Features Management

Unity
- Prefabs
  - ManomotionManager
  - Gizmo Canvas

Additional resources
- Video tutorials
- AR Foundation
- Unity (Supported versions, documentation)

Minimum requirements
- Platforms supported
- Tested devices

Performance information
- Processing time
- FPS

License keys

Limitations
- App set to background Android
- Processing Time
- Finger information
- Hand contour
- Left and Right hand
- Hand Rotation
- Power Save Mode
- False positive detection
- Tracking does not work for two hands
- Low end devices

RGB Input only
Dual screen rendering

Introduction

ManoMotion SDK PRO

The purpose of this product is to understand the human hand and analyze the gestures performed by the user using only a mobile device and a regular RGB camera. This document describes the features provided by ManoMotion SDK PRO and how the licensing system works. Moreover, it explains how to get started with Unity and how to integrate ManoMotion's SDK PRO into an existing Unity project.

What is new? - 1.4.9 Patch notes

Improvement for finger try on features.
Fixed left and right hand in front facing.
Lowering of the wrist try on position (to get a more natural position).
Improved Windows/Editor support (BETA).
Hand texture cut out (ALPHA)

Prerequisites

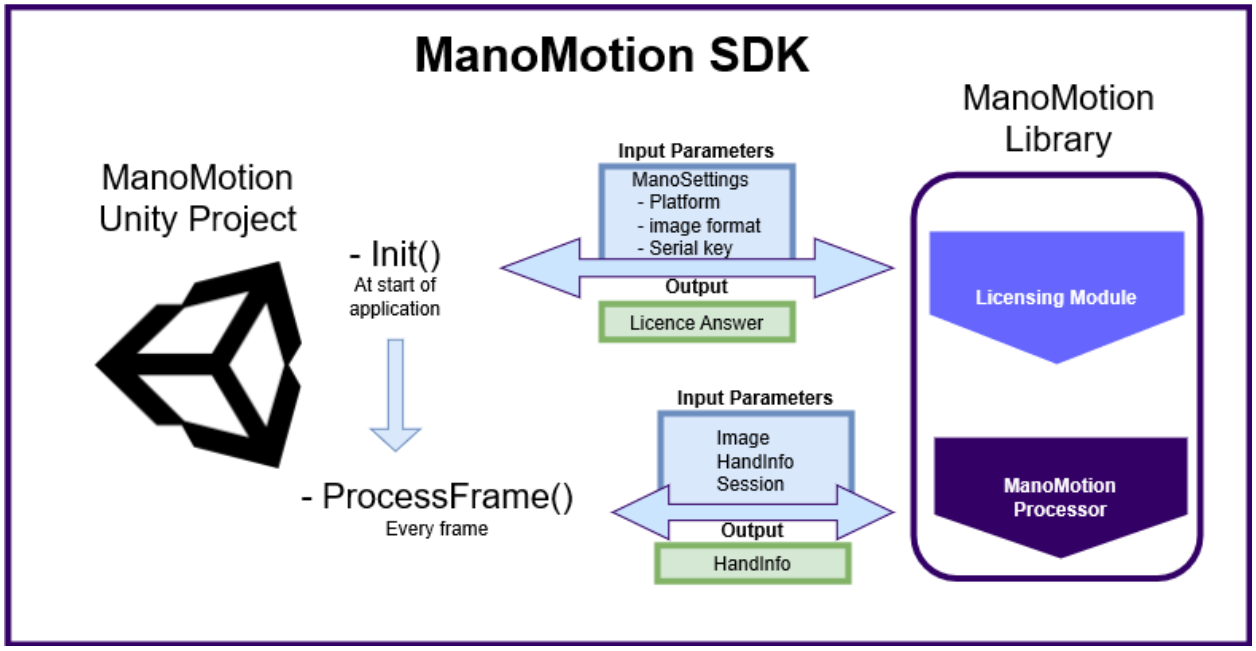
- General knowledge about the Unity 3D engine.
- Being able to deploy applications for Android or iOS using Unity.
- A device of preference (iOS or Android) for deployment (preferably a high end device)

Glossary

Term	Description
2D skeleton	21 joints all with the same depth (z) value.
3D skeleton	21 joints with individual depth (z) value.
Confidence (Unity)	This lets Unity know if the skeleton it detected or not.
Gizmos	Visualization of ManoMotion features.
POI	Point of interest. Interaction point used mainly on SDK Lite.

SDK Pro

This product contains a cross-platform library built in C & C++ which aims at enabling mobile devices to understand what the user is doing with its hand using just an RGB image as input. The library contains ManoMotion's Technology and a License Module which ensures that the developer is already registered in our systems. The technology is delivered as a Unity package that can be deployed for Android & iOS devices. As described in the image below, there are two main methods called from Unity with two different purposes: make sure the developer is using a valid license and feeding the SDK with frames.



In order to initialize the SDK the device is required to be connected to the internet. This process will validate the license and configure the library for starting its process. In addition, some data collection may also be executed in parallel which helps us to understand what is the overall performance of the technology in the device.

Since the SDK can be combined with other plugins such as Unity's AR Foundation, feeding the SDK with images can be done in two different ways:

- **Using AR Foundation.** In this case, AR Foundation takes control over the camera and does not allow the SDK to be fed directly from the camera. Instead, we clone the raw image displayed on the screen and feed the SDK with it.
- **Not using the AR Foundation.** This case is the standard product in which we control the camera and therefore feed the SDK directly using the raw image.

In order to exchange images in the most efficient way in between Unity and the library, we do not pass a new image every time. Instead, from Unity we set a memory address and notify the

SDK when a new image is available. In this process, we use two main data structures to receive output and also to give input in real time to the SDK, HandInfo and Session structures respectively. These structures are explained below.

HandInfo

HandInfo structure is used in order to output the information about a specific frame. Therefore, it contains the result of the frame analysis and is updated every time we call processFrame. The information is separated into three main categories:

- Hand Tracking (Understands the hand position).
- Gesture Analysis (Understands the hand gestures).
- Warnings.

Each of them will be explained in more detail in this document.

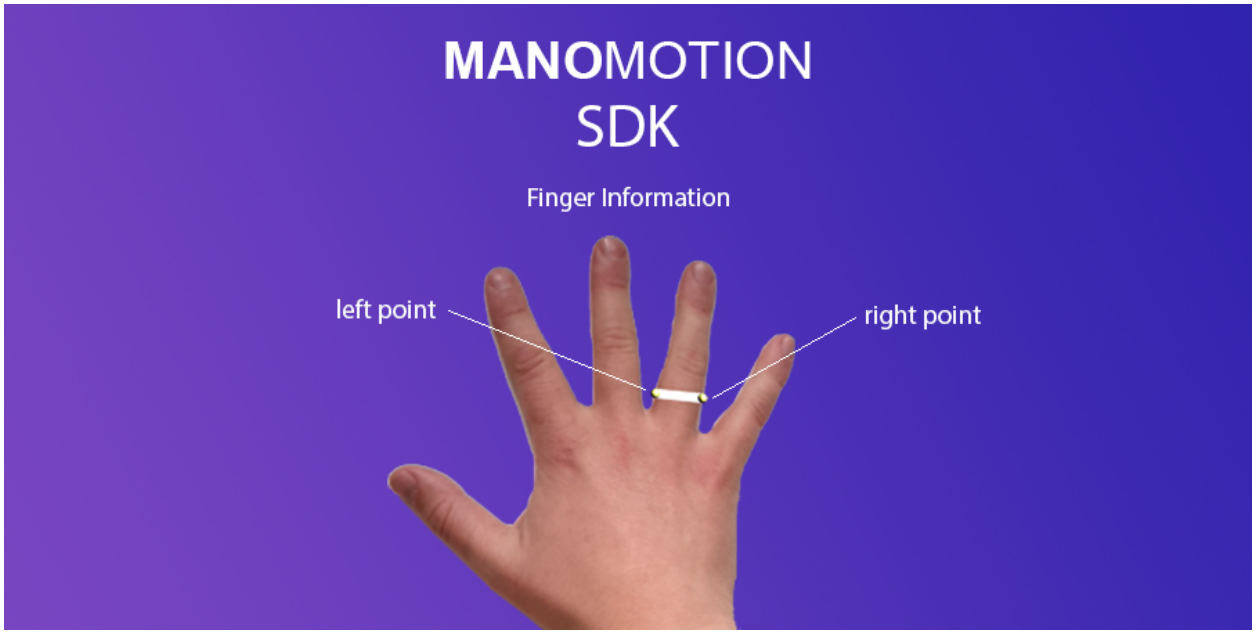
Hand Tracking (TrackingInfo)

The Hand tracking is provided in the structure called TrackingInfo. It provides deep information about the hand such as positioning individual points or the whole skeleton (all these points are normalized values between 0 and 1). These values must be denormalized and adapted to the device screen as explained in this example. It also gives the estimated depth value (distance of the hand from the camera).

Finger Info (BETA)

The finger information structure contains two normalized (0 to 1) Vector3 positions (Z value is 0), left_point and right_point together with an int (fingerWarning). The SDK calculates the location of the left and the right points of a specific finger segment. The FingerGizmo calculates the distance between the points. The tracking smoothing parameter affects the left_point and right_point. The finger warning will provide information if the finger width can not be calculated correctly.

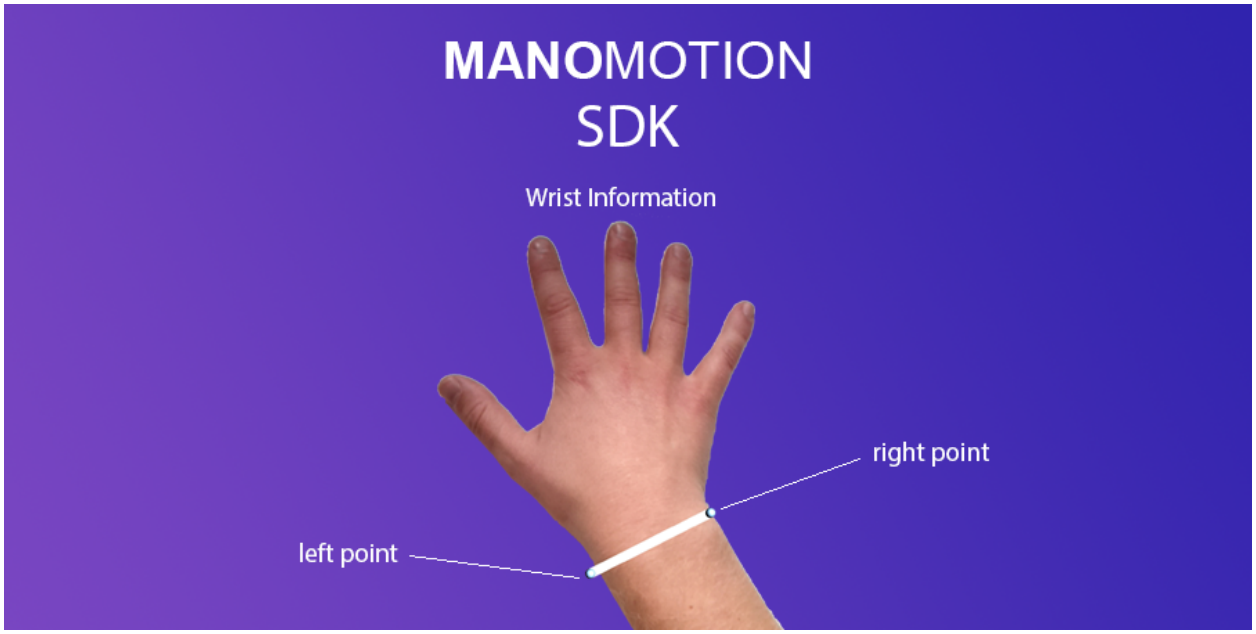
Error Code	Description
1002	The finger points are close to the edge of the frame.
1003	The finger points are outside of the frame.
1004	The finger points are too close to the palm
1001, 1005 - 1013	Calculation Errors



Wrist Info (BETA)

The wrist information structure contains two normalized Vector3 positions (Z values are 0) and an int (wristWarning). With this information it is possible to get the positions and the width of the wrist. This feature can be used for bracelets or watch try on cases. The tracking smoothing will affect the wrist points. The wrist warning will give information if the wrist width can´t be calculated correctly.

Error Code	Description
2000	Wrist outside of frame



Contour Points (BETA)

The hand contour consists of an array of 200 normalized Vector3 positions (Z values are 0) that are used to draw the contour around the hand. The contour points feature could for example be used for Glove try on use cases or hand segmentation. In the Unity demo scene we use a Linerenderer to draw the contour line using the contour points. The amount of contour points used for each frame is defined from the AmountOfContourPoint.

Amount of contour points

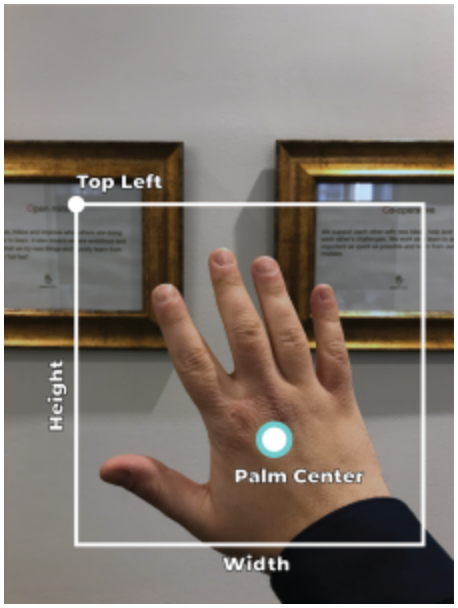
This int lets the contour know how many points that are used for each frame for the contour. This value is used in the ContourGizmo script to update the contour gizmo with the positions that are updated each frame.

Bounding Box

The Bounding Box is a box surrounding the hand provided in the structure called BoundingBox. It gives three values: a point with the top left position of the box (Vector3) and also the Width and Height (Float). The Bounding Box is not visualized in SDK PRO but it is accessible. It is illustrated below.

Palm Center

Gives the estimated normalized Vector3 position of the palm center. Smoothing to the palm center can be added by adjusting the tracking smoothing. See image below, it is not visualized in SDK PRO but it is accessible.



Depth Estimation

As an extension of other input methods such as the Mouse/GamePads/Keyboard, Hand Interaction is capable of supporting all 3 axes. As such, you can access the relative depth detected by Manomotion SDK as the distance of the hand from the camera by a value between 0 and 1. 0 when the hand is close to the camera and 1 when the hand is further away from the camera. As an example a value between 0.5 and 0.9 could be used for the interaction area.

Skeleton Tracking

The Skeleton Tracking is given in the structure SkeletonInfo which contains the confidence and the joint information of the skeleton. The confidence is a float value that will give 0 (when no skeleton is detected) or 1 (when skeleton is detected). The joints x and y (normalized) and z are provided in a normalized Vector3[21] with individual position for each of the 21 joints of the hand skeleton. The SDK can be configured through the variable enabled_features.skeleton_3d in Session in order to return the joint information in 2D or 3D. The SDK is configured by default to return 2D points. In this configuration the Z values of all joints will be 0. If the SDK is configured to output 3D joints, each joint will have a Z value in relation to the wrist joint [0] which always will get a Z value of 0.

Joint Orientation

The Joint Orientation will give each joint its individual orientation value as a Vector3 containing the x,y,z rotation values. The values are set in the SkeletonManager UpdateJointorientation() and used on the eulerAngels of the joints. The joint orientation also uses the handside and left or right hand information. You can see and try this out by enabling the 3D mode in the demo application provided in the Unity project, the jewel for each joint will now align in the direction of the fingers.

Gesture Analysis (GestureInfo)

While the hand tracking is able to follow the hand within the frame, the gesture analysis is able to understand what the user intent is. By combining information from previous and current frames, ManoMotion's technology is able to determine what type of gesture the user is performing. This information is provided in the data structure called GestureInfo and classified into ManoClass, ManoGestureContinuous & ManoGestureTrigger. These three categories aim at helping developers to design experiences that are fully customizable to different behaviours and can be map into the Unity world. They hold information that will allow developers to fire actions such as opening a menu when a user clicks or making a hole when the user keeps its hand closed for a while. The explanation of each category is found below.

ManoClass

The ManoClass represents the raw detection of the hand in every frame. This means that ManoMotion SDK will provide a result which does not depend on the previous information, it is a per frame analysis. For each frame given, the SDK will return a ManoClass result which can be any of the following hand classes (Grab, Pinch, Point, NoHand). The Manoclasses are the fundamental component of the Gesture Analysis. In the following link you can find more details about the supported classes and what type of analysis is done in each category.

Continuous Gestures

The Continuous Gestures are a convenient package of Gesture Analysis aimed to understand and categorize if the user is continuously performing a given Gesture. Despite the ManoClass, a Continuous Gesture is calculated based on present and past information. This means that a Continuous Gesture will be fired only if the user has kept the same hand pose for a certain amount of frames (time). A visual representation of the supported gestures can be found in this link.

Trigger Gestures

The Trigger gestures are one time gestures. These types of gestures are specific sequences of ManoClasses & hand states (explained below) that when performed sequentially they will be recognised as a trigger/event, similar to a mouse click. In this link, you can find an illustration of the supported Trigger Gestures.

Swipes

There are four different swipes that the SDK will detect: left**, right, up** and **down swipes**. The swipes work with any gesture. In the demo application it's possible to choose if they should show or not when performed. The swipes will trigger when the hand swipes from one side of the screen towards the other side above a certain speed which is controlled by the SDK. For now, this value is defined internally, in future releases of SDK PRO this threshold will be possible to customize from Unity. The trigger that will be detected is the one in which direction the hand moves, the below image illustrates a swipe left.

States

The hand states aim at giving more details regarding what the user is doing with their hands within the same hand pose (ManoClass). The SDK is able to classify the hand transition within the same ManoClass. For instance, in the ManoClass Grab, the hand can transit from open to close; a similar case happens with the Pinch Manoclass. In order to model those transitions, the SDK is capable of determining different states depending on the ManoClass.

- *Grab ManoClass:* in Grab there are three different hand states as illustrated in the image below. The values of these states are 0 (open hand), 6 (mid open/closed), 13 (closed hand).
- *Pinch Manoclass:* in Pinch there are two different hand states, 0 (open pinch), 13 (closed pinch)
- *Pointer ManoClass:* in Pointer the hand state is always 0.

Handside

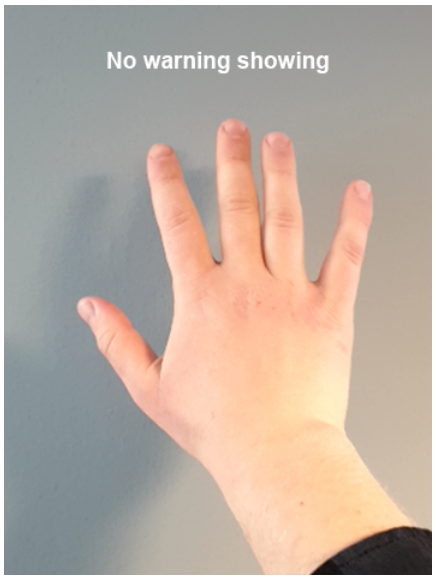
This provides information about the hand side that is facing the camera.
For the ManoClass Grab you can get the hand side information back or palm. In the cases of Pinch and Pointer ManoClasses, the hand side will always be back.

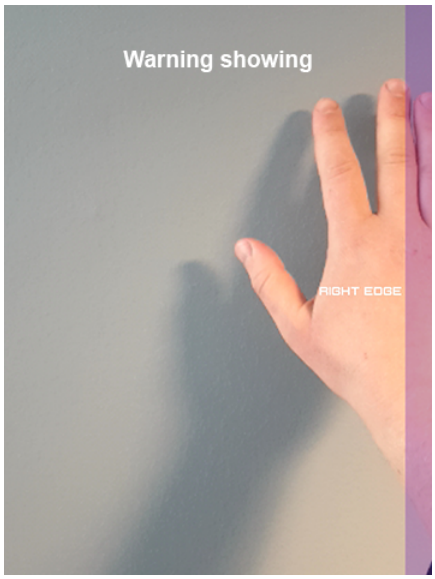
Left or Right hand

See Limitations.
The left or right hand information comes as a LeftOrRightHand enum with either left hand, right hand or none. The information updates every frame and can be found in the GestureInfo.cs

Warnings

The Warnings are used to prevent situations in which the hand may not be detected. For example, the edge warnings which will fire when the hand is reaching to any of the frame borders. To visualize this event, the Unity application will highlight the edge of the screen in which the hand is at risk of leaving the frame and therefore not being detected. It's very important that the whole hand is within the camera view for the SDK to work properly.





Session (SDK settings)

Session is a data structure used to communicate with the SDK in every frame. It is responsible for input and output operations such as setting the different smoothing values in order to get raw signals or more stable signals. Moreover, it communicates changes in regards to phone orientation. Finally, it is also used to enable/disable features such as Gesture Analysis, 2D to 3D joints or Fast Mode.

Flag

Provides information about the current image that is being sent to the SDK making sure the image is not too small. Smaller than 16x16. Also gives information if the Power Saver Mode is enabled when using an Android device.

Orientation

Gives the SDK information about the phone orientation, Landscape, Portrait etc. The SDK supports all orientations.

Front facing camera orientation

Starting from 1.4.8, our SDK supports front facing camera orientation. If you want to make use of it, set the Session's orientation to their corresponding front facing versions so the library knows you're using the front camera.

Gesture smoothing

The Gesture Smoothing takes a float value between 0 and 1 of the response time of the trigger and continuous gesture detection. A low value will detect triggers instantly. If the value is too low it will not be able to detect the click trigger gesture. It will instead detect pick and drop gestures. A suggested value to use for detecting all trigger gestures is between 0.4 and 0.65. The value can be set from the InstantiateSession() method in the ManomotionManager.

Tracking smoothing

The Tracking Smoothing takes a float value between 0 and 1. A 0 value will use the raw signal without any smoothing applied while 1 will apply the maximum smoothing to the tracking signal. A suggested value to use for tracking smoothing is between 0.1 and 0.2. A higher value will make the skeleton follow the hand with a delay but it can reduce jittering that can occur when a low smoothing value is set. The Tracking smoothing not only controls the smoothing for the skeleton joints it also controls the smoothing for the finger, wrist points and palm center. Try out which value fits your application best. The value can be set from the smoothing slider in the menu of the demo scene.

AddOn

Used to notify the SDK what type of the image is sent. For example, ARFoundation. Since the frames may be different, the SDK needs to know its characteristics in order to normalize to an image understandable for the core technology. Potential differences on images are the resolution, the number of channels and the color space. The SDK expects an RGB image with a minimum resolution of 320x240. In case a different image is sent to the SDK, it may not be able to process it and the developer will have to contact ManoMotion support.

Features management

The Features hold a number of settings that allow the developer to configure the SDK. When a feature is enabled it will add to the processing time so they are recommended to be active only if they are used, otherwise the performance of the application will be affected.

Skeleton

Integer used to configure the SDK to return 2D or 3D skeleton joints. Since 3D joints are not always needed, the developer can choose to enable/disable them. The main difference is that calculating the 3D joints takes slightly more time than not having them, so it is a choice for the developer whether to use them or not. The values are 2D (0) & 3D (1).

Gesture

Integer used to configure the SDK to run the Gesture analysis. With a 1 the Gesture analysis is enabled, with a 0 it is disabled. If it is enabled, the processing time will be affected as the SDK will also be calculating the gestures on every frame. This is set to 1 by default and some features like joint orientation and warnings work better with gestures turned on. Therefore the gesture toggle button has been removed from the UI in the demo scene but can still be set from the GizmoManager attached to the Gizmo Canvas in the Unity Scene.

Fast Mode

See Limitations. Integer used to configure the SDK in order to run faster in the hand tracking module. With a 1, the fast mode will be enabled and will optimize the tracking algorithms to perform faster operations. Depending on the hardware available, the fast mode will decrease the processing time while trying to keep up the quality of the tracking. It is not recommended in all cases as the user experience might drop. We encourage everyone to test this mode and provide feedback!

Finger Information

Integer used to configure the SDK if Finger width information should be calculated and sent up to Unity. The finger info will accept a number between 0 and 5. It is possible to choose which finger that should use the finger information, this can be controlled by sending the numbers described in the chart. This can be changed during runtime.

0	OFF
1	THUMB
2	INDEX FINGER
3	MIDDLE FINGER
4	RING FINGER
5	PINKY

Wrist Information

Integer used to configure the SDK if the Wrist information should be calculated and sent up to Unity. 1 will tell the SDK to calculate while 0 will stop the calculations.

Contour

Integer used to configure the SDK if the hand contour should be calculated and sent up to Unity. 1 will tell the SDK to calculate while 0 will stop the calculations.

Deprecated Features

Deprecated Hand Tracking Features

POI: it gives the same value as Palm Center.
Rotation: Float that only gives 0 values.
Finger tips: Vector3[5] all finger tips only give 0 values. Use SkeletonInfo.joints[4,8,12,16,20].
List of deprecated features.

Deprecated Features Management

Pinch poi
List of deprecated features.

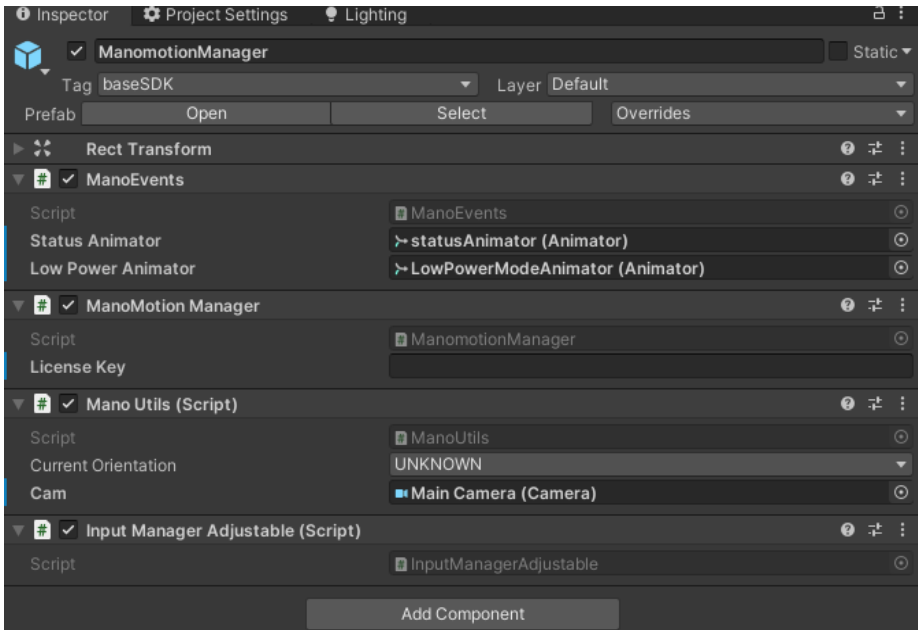
Unity

Prefabs

Inside the Unity package you can find a number of pre-made prefabs to make it easier to put the ManoMotion SDK into your project. Here is an explanation of the most important ones.

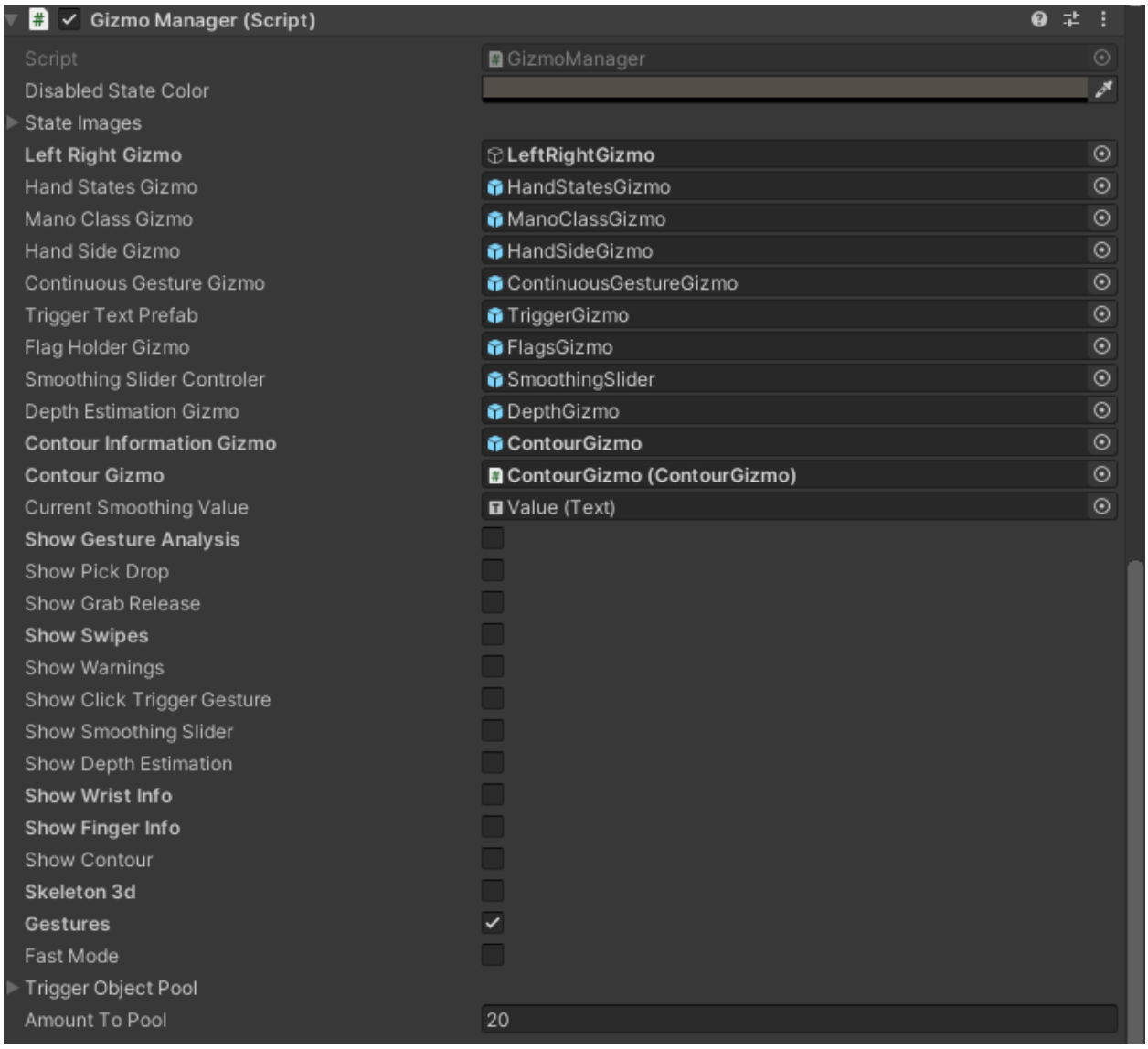
ManomotionManager

The ManomotionManager prefab holds some key components for the ManoMotion SDK.
The ManoEvents script will let you know if there are any problems with the license answer received from License Module. Also handles the notification for LowPower (iOS) / Save Power (Android) mode.
The ManoMotion Manager script is where the communication with the ManoMotion Library happens, it's also where you need to put in your license key.
Mano Utils holds a number of calculations for visualization of our features in relation to the screen.
From the ManoMotionManager visualization info you can access the rgb_image (the camera image) or the occlusion_rgb (contains the hand Cut out image), this is still a ALPHA feature.
The Input Manager collects information about the different cameras on the device, this is a new BETA feature in the SDK. This makes it possible to change which camera to use at runtime, both back facing and front facing.
The Input Manger Phone Camera handles the phone camera and sends the images to the ManoMotion Manager so the images can be processed by the SDK through the ProcessFrame().



Gizmo Canvas

In the Gizmo Canvas prefab you will find the Gizmo Manager, from this it is possible to set the features that you want to use/visualize in your project.



Additional resources

Video tutorials

ManoMotion YouTube tutorial playlist

AR Foundation

"*AR Foundation allows you to work with augmented reality platforms in a multi-platform way within Unity.*" from AR Foundation documentation.

If you are using the SDK PRO together with AR foundations you can find the documentation for AR Foundation here About AR Foundation | AR Foundation | 2.2.0-preview.6

You will need to add the AR Foundation Unity package from the Unity Package manager and also ARCore (for Android) or ARKit (for iOS).
You can get specific documentation for ManoMotion + ARFoundation in the specific package from our website.
We recommend using AR Foundation versions 2.

Unity (Supported versions, documentation)

We recommend that you use official Unity versions 2019.2.11f1 or above.
Unity - Scripting API:

Minimum requirements

Platforms supported

ManoMotion SDK PRO runs on Android & iOS devices + (Windows BETA).

Tested devices

Android
Samsung S8
Samsung S9 / S9+
Samsung S10
OnePlus 7 Pro
Xiaomi Pocophone F1

Xiaomi MI8
Xiaomi mi10 Pro
Huawei Mate 30 pro
Huawei Mate 20 pro
Asus ROG phone
OnePlus Nord
OnePlus 5T

iOS
iPhone X
iPhone XS
iPhone XS Max
iPhone SE 2020

Performance information

Processing time

This will show the time SDK takes to process the images in milliseconds.

FPS

The FPS counter will show the current frame rate of the application.

License keys

As mentioned earlier, the SDK requires a license in order to run. This license is formed with a key and a Bundle ID. The license key is a string in the following format: XXXX-FFFF-WWWW-YYYY which is generated automatically. The Bundle ID is a given string that will be used to identify the application in the stores such as *com.manomotion.example*. For the SDK Pro, the license keys are usually created by ManoMotion and delivered to the client, together with the documentation and the required packages. These types of licenses fall under the OEM business plan and therefore are generally created for testing purposes with limitations. The limitations are expiration date and number of devices in which the technology can be deployed (credits = devices). In the OEM testing licenses, the license key expires after 30 days and contains 10 credits which means it can be deployed to up to 10 devices.
For more information follow this link.

Limitations

App set to background Android

When the application goes to background mode and back it sometimes causes a crash.

Processing Time

At some times the processing time can increase for one or two frames but gets back to normal after that.

Finger information

The finger width information works best on a uniform white background. When used on other backgrounds the quality of detection might decrease.

Wrist information

The wrist width information works best on a uniform white background. When used on other backgrounds the quality of detection might decrease.

Hand contour

The hand contour works best on a uniform white background. When used on other backgrounds the quality of detection might decrease.

Left and Right hand

The left and right can sometimes give inconsistent values.
When running fast mode right hand classification does not work.
When using front facing mode the left/right does not work as expected.

Hand Rotation

Tracking might not work as well when the hand is rotated.

Power Save Mode

Android manufacturers don't use the same standard for the Power Save Mode, due to that Power Save Mode notification will not work for all android devices.

Tested brands:
Samsung
Lenovo

False positive detection

Sometimes the SDK detects something as a hand, even though that might not be a hand.

Tracking does not work for two hands

The current version of ManoMotion SDK PRO only supports use with one hand at the time.

Low end devices

We recommend using high end devices, examples of phones from Samsung Galaxy 9 (Android) or above or from iPhone X (iOS) devices.

RGB Input only

We only support RGB images as input for the SDK PRO.
So it will not work on, for example, grayscale images.

Dual screen rendering

Using SDK PRO in VR applications with stereo rendering may not give the same result as rendering one screen.