MANOMOTION

SDK PRO 2.1.1

Technical Documentation

Table of contents

- Table of contents
- Introduction
  - ManoMotion SDK PRO
  - 2.1 Patch notes
  - 2.0 Patch notes
  - Main components
  - Prerequisites
  - Glossary
- SDK PRO
  - HandInfo
    - Hand Tracking (TrackingInfo)
      - Bounding Box
      - Depth Estimation
      - Skeleton Tracking
      - Joint Orientation
      - Hand contour
      - Two hands
      - Try-On Features
        - Finger Info
        - Wrist Info
    - 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

- One Euro Filter
- AddOn
- Features management
  - Skeleton
  - Gesture
  - Fast Mode
  - Finger Information
  - Wrist Information

Unity Package
- Prefabs
  - ManoMotionManager

AR Foundation
- Using ARFoundation with SDK Pro
- ARFoundation Input Manager
- ARFoundation AddOn

Additional resources
- Tutorials
- Unity (Supported versions, documentation)

Minimum requirements
- Platforms supported
- Tested devices
  - Android
  - iOS

Performance information
- Processing time
- FPS

Limitations
- Processing Time
- Finger information
- Left and Right hand
- Hand Rotation
- Try-on features Rotation
- Power Save Mode
- False positive detection
- Low end devices
- RGB Input only
- Dual screen rendering
- Hand Occlusion

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. Moreover, it explains how to get started with Unity and how to integrate ManoMotion's SDK PRO into an existing Unity project.

2.1 Patch notes

Reintroduced Hand Occlusion
The hand occlusion functionality is now in the ManoMotion SDK. To give you an example how this works we have added the Occlusion scene under the ManoMotionScenes prefab which you can find in any of the Unity scenes in ManoMotion > Scenes.

Hand reminder and placement
For the example scenes we have added a hand placement zone to guide the user where to put your hand so it's at a good distance from the camera. We also added a hand template that breathes in and out when there is no hand found.

Depth radar
In the Interaction scene we added a kind of radar that shows the user how close they are with their hand to the correct depth to be able to grab the sphere. This disappears after the sphere has been grabbed for the first time.

Android minimum API version
There was code in an editor script which would set the minimum API version to 24, this has been removed.

2.0 Patch notes

Removed licensing
The licensing module has been removed. This makes it so that users do not need their own personal license keys which were connected to specific package names anymore.

Image processing threading
Option to run image processing on a separate thread than Unity's main thread (see ProcessingType on the ManoMotionManager, set to Async by default). This lets Unity run at a much higher FPS without interruptions (especially on Smartphones) in return for a short delay in tracking.

Full package cleanup

- Prefabs and example scenes: Adjustable (Windows, Android and iOS), ARFoundation (Android and iOS) and Video input.
- Updated naming conventions, components and folder structure.
- Added ManoMotion namespace.
- Removed deprecated values, scripts, assets and scenes.

  New: handInfos[0].gestureInfo
  Old: hand_infos[0].hand_info.gesture_info

Preview features

- TryOn features for rings, bracelets and watches by using FingerInfo and WristInfo.
- 3D hand mesh controlled by hand tracking.

Main components

ManoMotionManager
This component handles the communication between camera input and image processing through extern methods, the results of the processing is stored in the HandInfos array. The ManoMotionManager also contains a **Session** which has some settings for SDK smoothing and which features to be enabled like gestures, wristInfo and fingerInfo.

It also contains the hand information from processing that contains hand- and gesture tracking.

InputManager
These components read the camera image and send them for processing through events. There are different types depending on what you want to do with your project:

- **InputManagerAdjustable:** The SDK takes control of the camera. This is used for Windows and mobile apps without AR functionality.
- **InputManagerARFoundation:** AR Foundation handles the camera input and displays the background. Images are retrieved from ARCameraBackground which should be on the Main Camera.
- **InputManagerVideo:** Runs a VideoPlayer and sends each frame for processing. Can be used for debugging purposes or setting up test cases.

ManoVisualization
When running without AR Foundation this component displays the camera background in front of the camera at a specified distance. The background is scaled to fit the device screen with methods from ManoUtils. Although there is a setting for if the background should cover the entire screen or fit completely inside the screen.

- **HandOcclusion**
  Displays the hand cutout on a mesh in world space on top of the camera background.
  Depth is distance from the camera to the hand wrist.
  Updates scale with methods from ManoUtils.
- **HandOcclusionARFoundation**
  Displays the hand cutout on a UI image.
  Depth is distance from the camera to the hand wrist, set on the Canvas Plane Distance.

  NOTE:
  “Screen Space - Overlay” makes the hand cutout align well but doesn’t give it any depth.
  “Screen Space - Camera” with the Render Camera set makes the hand cutout be slightly misaligned but gives it depth.

SkeletonManager
Uses the hand tracking information to control hand skeletons by updating joint positions and rotations with extra smoothing filters (OneEuroFilters). Has the option to hide the skeletons.

Contour
Used for hand occlusion. EnabledFeatures contains an int used to configure the SDK if the hand contour should be processed.

Number of contour points
TrackingInfo contains the int field numberOfContourPoints which lets the contour know how many points are used for each frame for the contour. This value is used in the ContourGizmo component to update the hand outline each frame.

Prerequisites

- General knowledge about the Unity 3D engine.
- Being able to deploy applications for Windows, Android or iOS using Unity.
- A device of preference (Windows, Android or iOS) 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 is detected or not.
Gizmos	Visualization of ManoMotion features.

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 their hand using just an RGB image as input. The library contains ManoMotion's Technology delivered as a Unity package that can be deployed for Windows, Android & iOS devices.

The SDK can be used together 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

The 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 process a new frame. The information is separated into three main categories:

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

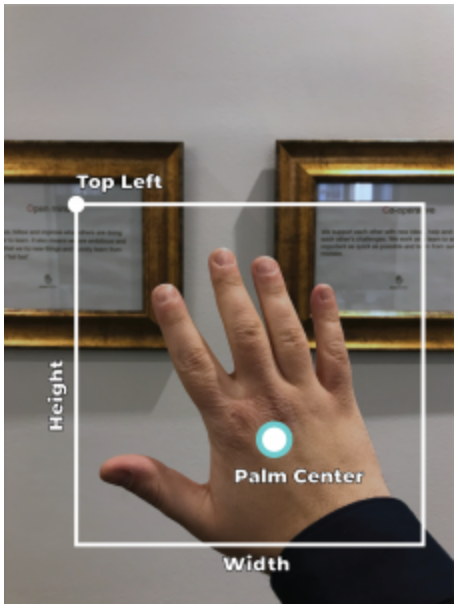
Each of them will be explained in more detail in this document.

Hand Tracking (TrackingInfo)

Hand tracking is provided in the structure called TrackingInfo. It provides deep information about the hand such as estimated depth value (distance of the hand from the camera) and individual joint points (normalized values between 0 and 1). The normalized joint position values must be denormalized and adapted to the device screen which can easiest be done with ManoUtils.CalculateNewPositionWithDepth.

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), the width and height. The Bounding Box can easily be displayed by adding the BoundingBoxManager prefab.



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 the ManoMotion SDK as the distance of the hand from the camera by a value between 0 (close) and 1 (far).

Skeleton Tracking

Skeleton Tracking is given the structure SkeletonInfo which contains 21 normalized joint positions and the confidence of the skeleton. The confidence is a float that goes from 0 (not detected) to 1 (detected).

Joint Orientation

The joint orientations have been changed to be calculated based on a target joint instead of having the orientations in the HandInfo struct. The joint up vector will go toward the next joint (for example joint 1 to joint 2) and the forward vector will be the camera's forward direction when the hands backside is showing and towards the camera when the palmside is showing. There are also rotation values for each hand which are stored in the SkeletonManager and updated in UpdateJointsOrientation() and can be used for things like grabbing objects.

Hand contour

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. The amount of contour points used for each frame is defined from the numberOfContourPoints.

Two hands

The ManoMotion SDK supports processing for two hands at the same time. This makes it possible to build “smartphone headset” experiences where the user only interacts with their hands. The ManoMotionManager contains an array of two HandInfos which contains processed hand information.

Try-On Features

These features enable developers to create their own try-on experiences and applications.

Finger Info

The finger information structure contains two normalized positions for points on the left and right side of the finger and an int for any warnings (fingerWarning). The SDK calculates the finger information for the finger specified in the sessions features

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

The wrist information structure contains two normalized positions 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.

Error Code	Description
2000	Wrist outside of frame

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's 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 behaviors and can be mapped 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. More information about each gesture can be found with visuals here.

The explanation of each category is found below.

ManoClass

The ManoClass represents the raw detection of the hand in every frame and does not depend on previous gesture information. The result can be any of the following classes: Grab, Pinch, Point and NoHand. The ManoClasses are the fundamental component of the Gesture Analysis.

Continuous Gestures

Continuous Gestures are aimed to understand and categorize if the user is continuously performing a given Gesture. The Continuous Gesture is calculated based on present and past information which 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).

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.

Swipes

There are four different swipes that the SDK will detect: **left, right, up** and **down swipes**. The swipes work with any gesture. 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. 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 which side of the hand that is facing the camera. Can be back or palm when ManoClass is Grab, with other hand gestures the side will always be back.

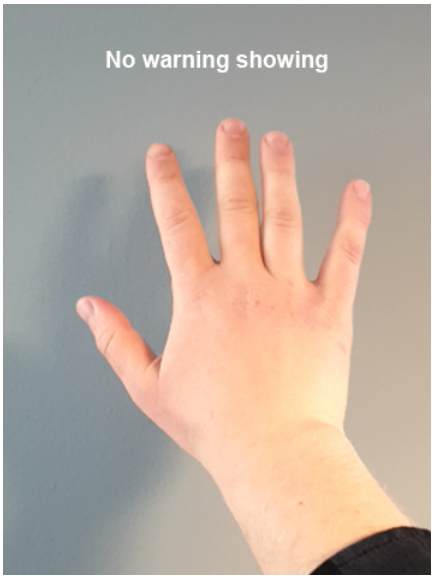
Left or Right hand

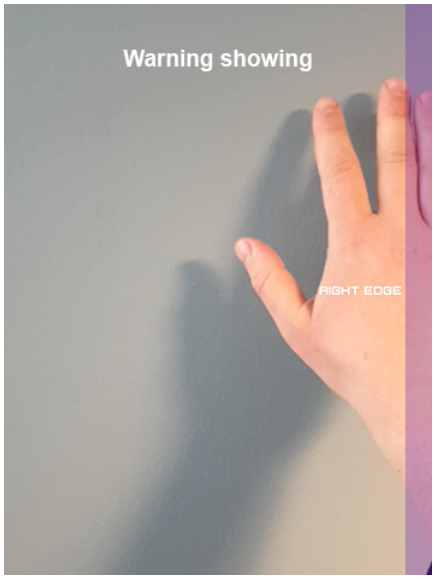
See Limitations.

The left or right hand information comes as a LeftOrRightHand enum with either the 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 can 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. Also gives information if the Power Saver Mode is enabled when using an Android device.

Orientation

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

Front facing camera orientation

The SDK supports front facing cameras with some limitations. To change to the front facing camera enable the isFrontFacing value on the InputManager or call the SetFrontFacing method during runtime.

Gesture smoothing

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.

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 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. 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 SkeletonManager also applies OneEuroFilters to the skeleton joints.

One Euro Filter

We recommend using the already implemented One Euro Filter class, which is more dynamic and stable than the standard “tracking smoothing” feature from the SDK. It is already implemented in the SkeletonManager class. It can also be applied to any GameObject in the scene, making it very useful for apps that rely on several tracked references. Give it a try!

Source code credit to Dario Mazzanti: https://github.com/DarioMazzanti/OneEuroFilterUnity

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 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.

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 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.

Unity Package

Prefabs

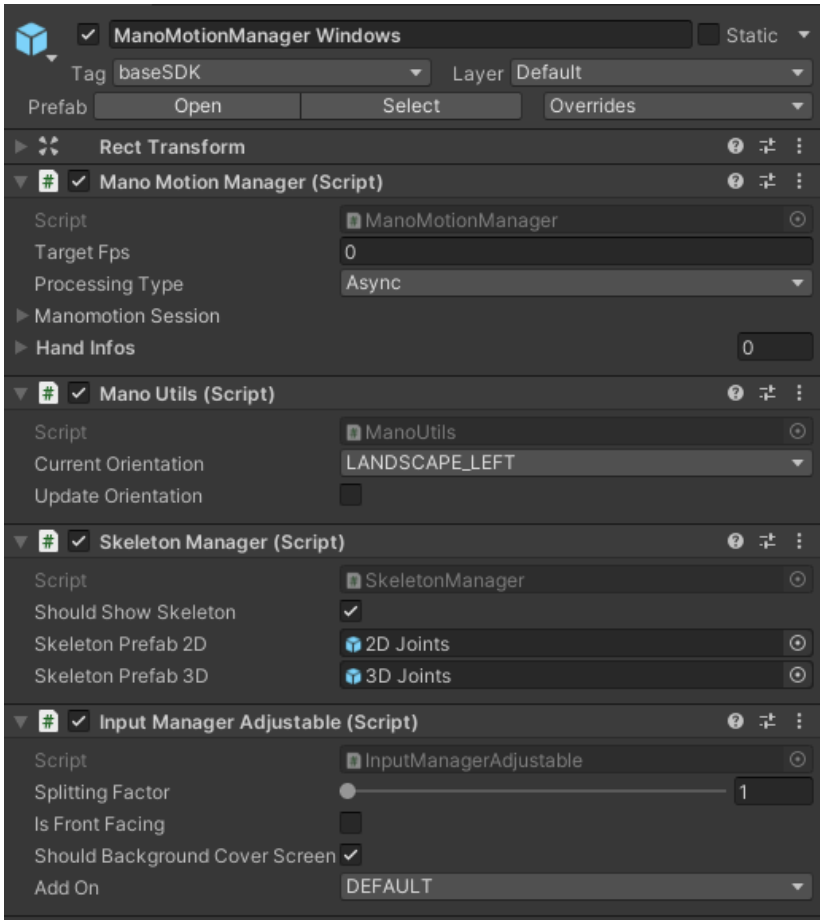
Inside the Unity package you can find a number of prefabs to make it easier to put the ManoMotion SDK into your project. Here is an explanation of the most important ones.

ManoMotionManager

The ManoMotionManager prefabs hold some key components for the ManoMotion SDK and is where the communication with the ManoMotion Library happens. From the ManoMotionManager visualization info you can access the camera image (rgbImage).

ManoUtils contains calculations for visualization of our features in relation to the screen.

The Input Manager collects camera image information and has methods to swap between back and front facing cameras.



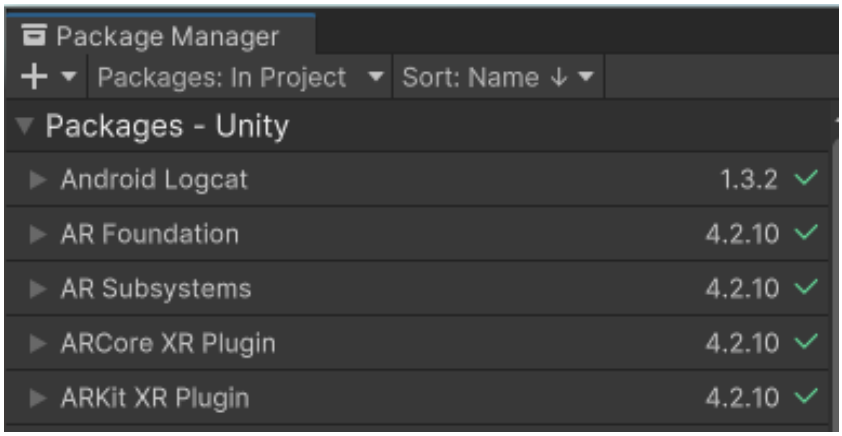
AR Foundation

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

ManoMotion SDK PRO is compatible with Unity's ARFoundation package, tested with the latest Unity LTS versions 2021.3 (4.2.10) and 2022.3 (5.1.5) Using ARFoundation, you can create lots of innovative AR experiences by combining hand tracking/interaction and ARFoundation's AR and spatial understanding capabilities.

Using ARFoundation with SDK Pro

You will need to install the AR Foundation Unity package from the Unity Package Manager and ARCore (Android) or ARKit (iOS) depending on the target device. In Project Settings > XR Plug-in Management ARCore and ARKit should be enabled in the Android/iOS tabs to be initialized at startup.



ARFoundation Input Manager

The main difference between the Adjustable and ARFoundation Input Manager is the way the SDK receives the images. In SDK PRO, ManoMotion takes control over the device camera and sends the images to the SDK. When using ARFoundation, it will take control over the camera and clone the frames from the AR Camera Background to send the images to the SDK.

When the ManoMotion SDK controls the camera it requests an image with a certain size and gets the closest possible resolution. The requested image resolution is set to be small in order to get better performance. It is possible to set the LOWEST_RESOLUTION_VALUE variable (that is used to set the scale of the image cloned) in the InputManagerARFoundation component. If a smaller image is used the detection quality might be reduced, but it will gain some speed. We recommend using a value of 480.

ARFoundation AddOn

When using SDK PRO with ARFoundation, the SDK needs to use the ARFoundation's AddOn from the session settings so the SDK knows how to handle the images it receives internally. This is done automatically in InputManagerARFoundation.Start():

```
OnAddonSet?.Invoke(AddOn.ARFoundation);
```

Additional resources

Tutorials

How to develop your own Ring try-on app

ManoMotion SDK CE YouTube tutorial playlist

Unity (Supported versions, documentation)

We recommend using the official latest LTS versions of Unity 2021.3 and 2022.3.

Unity - Scripting API

Minimum requirements

Platforms supported

ManoMotion SDK PRO runs on Windows, Android & iOS devices.

Tested devices

This is a non-comprehensive list of devices in which we have tested our technology on. It may change in the future as the libraries and packages requirements change as well.

Android

Samsung S8
Samsung S9 / S9+
Samsung S10
Samsung S20+
Samsung S22+
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
iPhone 11
iPhone 12
iPhone 12 mini
iPhone 13 Pro

Performance information

Processing time

This shows the time it takes the SDK to process the images, in milliseconds.

FPS

The FPS counter shows the current frame rate of the application.

Limitations

Internet connection

The package requires an internet connection when starting an application with it.

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.

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.

Try-on features Rotation

Try-on features (finger, wrist) are only 2D.

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.

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.
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.

Hand Occlusion

When running the hand occlusion with processingType set to Async the occlusion image will not get synced with the latest camera frame but rather the last processed image. Therefore it’s recommended to set processingType to Sync when using the camera background and not a 3D environment.

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