



SDK CE

by ManoMotion

MANOMOTION



1. Overview	3
2. Quickstart	3
3. Setup ManoMotion SDK in Unity	4
4. SDK Community Edition	5
4.1. Input Signal	5
4.2. ManoLicense	6
4.3. SDK Features	7
4.3.1. Hand Tracking	7
4.3.2. Gesture Recognition	8
4.3.3. Create Your Own Gesture	10
4.4. Properties	11
4.4.1. ManoMotion Manager	11
4.4.1.1. Hand Info	12
4.4.1.2. Session	13
4.4.1.3. Visualization Info	13
4.5. Useful Developer Information	13
5. Supported Devices & Requirements	14
5.1. Requirements	15
5.2. Performance	16
6. Common Issues & Solutions in Unity	16



1. Overview

Welcome to the SDK Community Edition Documentation!

The purpose of this product is to track the human hand and analyze the gestures performed by a user. In the following sections you can get familiar with the features provided by SDK CE and how the licensing system works. Moreover, you can get started with the Unity Package and integration of ManoMotion's SDK CE technology to your project. Finally, you can find the documentation of classes and methods available.

In the following sections, how to use ManoMotion's SDK. We specifically define:

- The type of gestures supported by the SDK CE
- How to interpret the supported gestures
- Good practice guidelines on utilizing the SDK

**Gesture Recognition covers a wide scope, and is a relatively new field when it comes to understanding user's intent. Therefore, it is important to note that there is little or no standards in universally defining 3D hand gestures or in methodologies underlying gesture recognition. ManoMotion aims to define these standards, and we assure that the SDK will mature and improve constantly, as we remain committed to developing our core technology.*

2. Quickstart

Follow these steps in order to get a first experience of the SDK CE capabilities:

1. Download the ManoMotion Unity Package HERE(<https://www.manomotion.com/products-download/>)
2. Create a new project, we recommend the latest updated versions HERE(<https://unity3d.com/get-unity/update>), and import all the contents of the package.
3. Go to Build Settings and set the Platform to either iOS or Android. SDK CE works ONLY on a mobile platform.
4. If you are SDK CE with AR Foundation, you need to add the AR Foundation (ARCore / ARKit) from Unity Package manager. Follow this link(<https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@2.2/manual/index.html>) for more details.
5. Make sure that your computer has the necessary files to develop for mobile phones
JDK(<https://www.oracle.com/technetwork/java/javase/downloads/index.html>)
, Android Studio(<https://developer.android.com/studio/>)
, Xcode(<https://developer.apple.com/xcode/>)
6. Our Editor Script, ManoMotionSetup, will make some automatic adjustments in the player settings so please be aware.
 1. [Android] Set Scripting BackEnd, IL2CPP
 2. [Android] Set Target Architecture to ARM64
 3. [Android] Install Location, Prefer External
 4. [Android] Internet Access Require
 5. [Android] Write Permission, External (SDCard)
 6. [iOS] A default Camera Usage Description – You can replace that.
 7. [iOS] Set Architecture to ARM64
 8. [iOS] Scripting Backend IL2CPP
7. If the Project has the default Bundle Identifier/Package name "com.DefaultCompany", that comes as a



default form Unity, our Editor Script will replace it temporarily with “com.manomotion.sdkceexample”. This is done to speed up our licensing system setup. More in depth at Setup ManoMotion SDK in Unity.

8. Navigate into the Assets folder -> ManoMotion -> Scenes -> Open the ManoMotion SDK CE Scene.
9. The ManoMotionManager object is already packed with a license that in combination with the bundle id/package name from step 5 will allow you to compile without the need of another license.

Important notes:

Make sure that your phone has a stable internet connection. It will be used during the initialization step to verify your bundle ID and License Key.

If you are using a bundle ID and license of your own, make sure that your bundle ID and your license key do not include any spaces or other characters.

If you are combining the SDK CE with any other SDK, make sure that the changes in step 5 are not affected by any existing manifest.


In case you are experiencing missing prefabs, upgrade your Unity version through the Unity Help menu.

3. Setup ManoMotion SDK in Unity

If this is the first time making use of SDK CE, we highly recommend to follow the steps described from Quickstart. The application of compiling the “ManoMotion SDK CE” is an app that showcases all the features included in the SDK CE. You can become familiar with the full list of features and decide which ones you want to be actively calculated. As mentioned in the QuickStart section, by default the Bundle Identifier/Package Name will be initialized with a Free ManoMotion license so you can try it out.

On the other hand, if you have an existing project with a Bundle ID/Package Name, then log in to our website and navigate to the License Page(<https://www.manomotion.com/my-account/licenses/>)

. Click the Add License button and fill the information in the pop up window.




ADD APPLICATION LICENSE

NAME OF YOUR APPLICATION (EX: APPNAME)

SELECT PLATFORM


Select a value



BUNDLE ID (EX: COM.COMPANYNAME.APPNAME)

APPLICATION TYPE

Select a value



GENERATE LICENSE KEY



Upon completion, we will generate a License for you that is now paired with the Bundle ID/Package Name. This information, alongside the expiration time and credits for this license, is going to be always available in the License tab.

Back to the Unity Project, you now make use of this newly generated license key, to the project that has the matching Bundle ID/Package Name. To Do so, Navigate to Assets/ManoMotion/Prefabs/ManoMotionManager. This prefab is the core Gameobject that you should have in your scene. It holds among other useful scripts, the ManoMotionManager.cs Script that provides all of the tracking information and gesture analysis. This script has also the exposed variable "License Key" *It's the value that looks like XXXXX-XXXXX-XXXXX-XXXXX. You should paste the License key that corresponds to this Bundle ID/Package Name and drag this Prefab in your scene.

By default, the ManoMotionManager prefab has an InputManager script that opens the phone's camera and provides the ManoMotionManager with RGB Frames. Based on these frames, the ManoMotionManager is processing the information internally and returns the Tracking and Gesture Information through the appropriate data structure. The class of ManoMotionManager is a static Singleton and it can be accessed in any other script via calling the ManoMotionManager.Instance.

4. SDK Community Edition

Licenses, Key features and Properties

This section introduces the concepts and terms relevant to documentation and licensing in order to make use of ManoMotion's technology. Here you will find relevant information in order to familiarise yourself with the key components of ManoMotion SDK as well as the licensing system.

4.1. Input Signal

ManoMotion's SDK requires a standard RGB image to work properly. The image is resolution independent meaning that the software will track and recognize the hand regardless. However, it is important to consider that high resolution images impact processing time. The lowest recommended image resolution is 320 x 240. By default, the ManoMotionManager prefab has an InputManager script that opens the phone's camera and provides the ManoMotionManager with RGB Frames.



Example of an Input Frame

4.2. ManoLicense

Every time a device attempts to make use of ManoMotion SDK CE, it needs to confirm that the licensing state is



in order. This is done via the **Init** method that is by default called once from the ManoMotionManager. This method passes the Bundle ID/Package Name alongside the License provided (See Setup ManoMotion SDK in Unity). This method checks with ManoMotion's licensing server and connects the information from the license with the ManoLicense data structure. This structure communicates to the developer the following information about the current license:

- **license_status**: notifies whether the license is working or not. Usually, it just works out of the box when a specific license is used for the first time. However, there are several reasons for a license to stop working. Some of the common reasons is because the license requires internet, it is expired (the free licenses last 180 days), number of credits has been used the free license allows up to 10.000 unique device installations.
- **credits**: notifies the developer how many credits are left and the need to potentially purchase more.
- **version**: notifies current SDK version.

This information is obtained every time the app starts. It double checks with the servers to get the latest updates regarding the license and other useful information.

It is a good practice to program your application in a way that if something goes wrong in the initialization process you can catch it and notify through the UI what the potential issues can be. A detailed list of LicenseAnswers are available with the corresponding message in the Unity project.

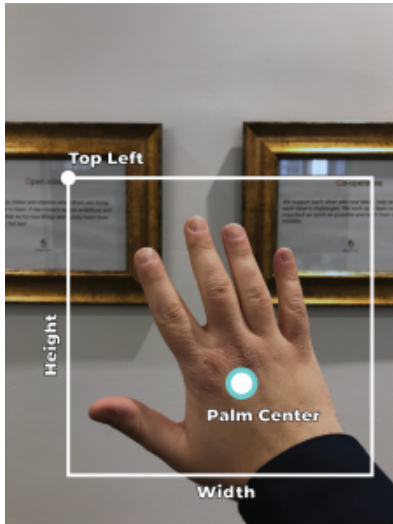
Data Type	Variable
LicenseAnswer (int)	license_status
int	credits
float	version

4.3. SDK Features

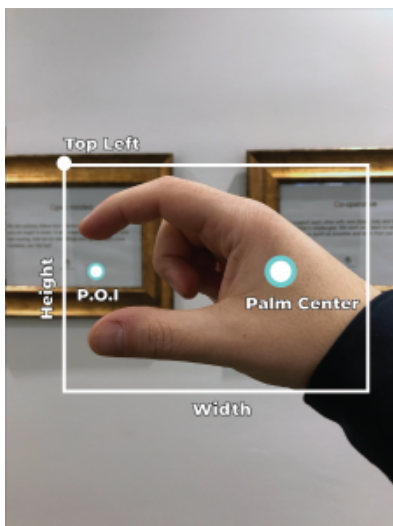
By including the SDK CE in your projects, you get access to a robust product that promises to cover a variety of use cases for the world facing camera. The two main categories of these features are Hand Tracking and Gesture Analysis. These two categories are linked with each other in the sense that ManoMotion's technology is able to detect and track the hand in a given frame and provide additional information of the interaction intent by understanding what the user is doing with their hands.

4.3.1. Hand Tracking

The basic information is provided by the bounding box that surrounds the hand. This bounding box contains the top left location of the rectangle and the size of the box itself. In addition, the SDK provides the palm centre as a reference point for all the hand processes and the P.O.I (Point of Interest) when the user has the Pinch Hand Pose.



Tracking Information – Bounding Box of Open Hand



Tracking Information – Bounding Box of Pinch Class and P.O.I

4.3.2. Gesture Recognition

While our Hand Tracking is able to follow the hand within the frame, our gesture recognition is able to understand what the hand is trying to do. By combining information of previous and current frames, ManoMotion's technology is able to determine what type of gesture the user is performing. In order to do so, the SDK offers a series of information for developers to use in order to map the user input to the functionality they desire.

The quality of Gesture Recognition is dependent on the quality of the hand detection, as such, in order to get the best results please follow the same guidelines and good practices while using the SDK.

Manoclass

For any image captured, Manomotion SDK categorizes each hand pose in one of the three major families (Grab,

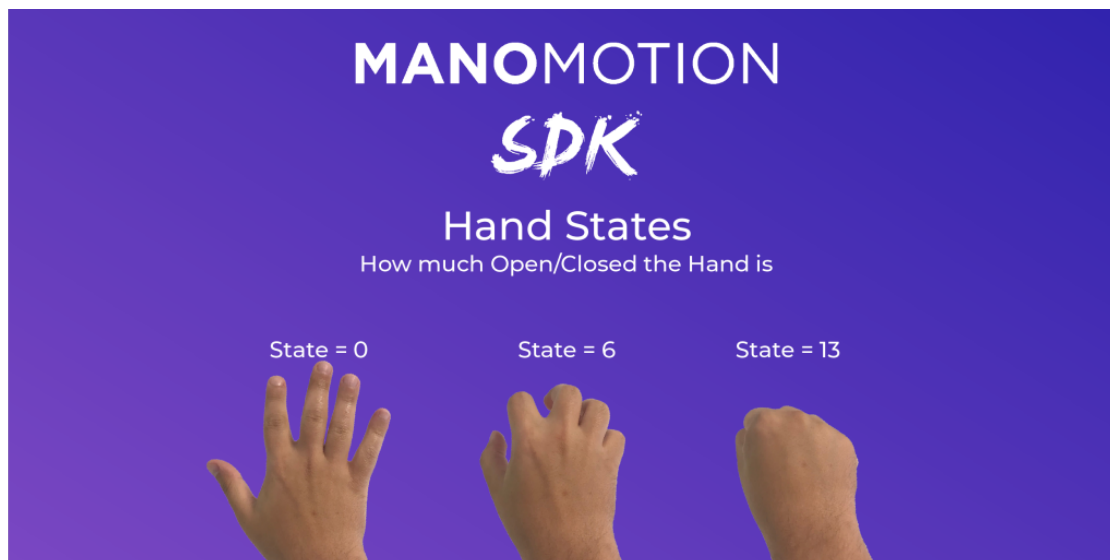


Pinch, Point). Those categories are referred to as Manoclasses and they are one of the most prime components of the Gesture Analysis. The values are updated in every frame and they provide the information of which Manoclass is currently being detected.



Hand State

The main goal of the hand state is to tell how open or close the hand/gesture is at that particular frame. For the ManoClasses Point and Pinch, the number of states provided are two, 0 and 13, which signify the Closed or Open state. For the ManoClass Grab, there are three states: 0, 6, 13, which stands for Closed, Halfway and Open. The value of hand state will be updated for every frame the hand is detected.



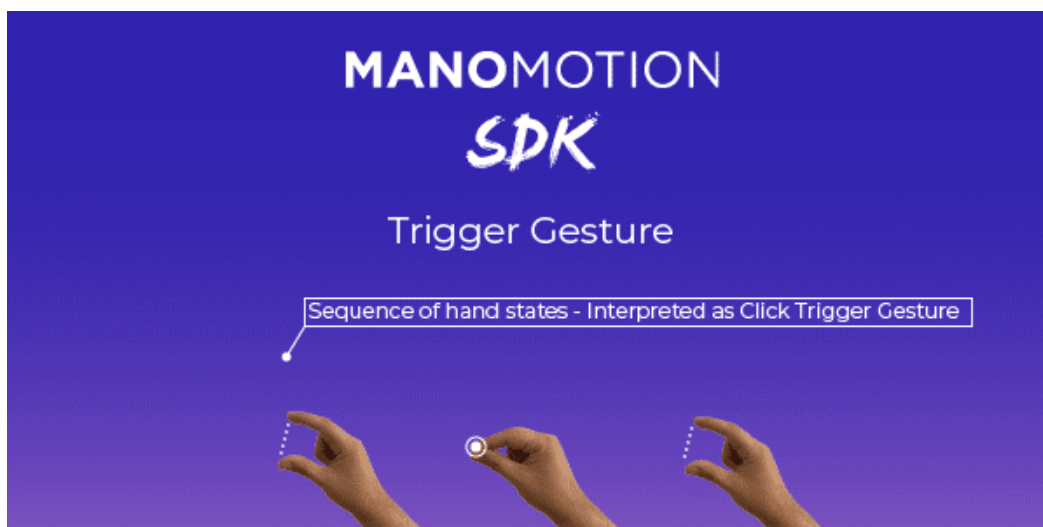
Continuous Gesture

Similar to the ManoClass continuous information, Continuous Gestures are a convenient package of Gesture Analysis aimed at understanding and categorizing whether or not the user is continuously performing a given Gesture. An overview of the ManoClasses, Continuous and Trigger gestures can be found here(<https://www.manomotion.com/supported-gestures/>)



Trigger Gesture

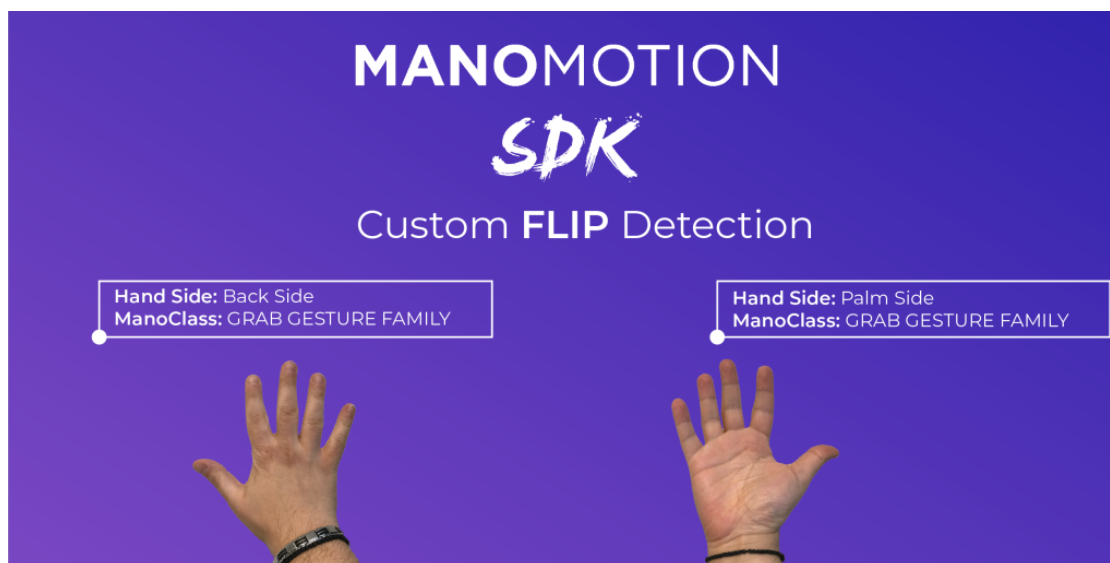
These types of gestures are meant to be used as an Event Type of input, similar to that of a mouse click. In practice, trigger gestures are specific sequences of hand states. If performed sequentially they will trigger an event. In contrast to Continuous Gestures and ManoClasses. An overview of the ManoClasses, Continuous and Trigger gestures can be found [HERE](https://www.manomotion.com/supported-gestures/)(<https://www.manomotion.com/supported-gestures/>)



4.3.3. Create Your Own Gesture

Even though the SDK CE comes with a set of predefined Gestures (Continuous and Triggers), your project may require custom gestures or even adjusted existing gestures. This is totally possible to achieve by combining the previously discussed components such as the ManoClass, HandState, HandSide e.t.c

The modeling of new Gestures or Hand Poses can happen by coding logic that checks for certain conditions to happen. As an example: if the detected HandSide of FrameNumber:0 is Backfacing and the detected HandSide of FrameNumber:1 is PalmFacing, then your code logic can assume that the user has “Flipped” his hand.



4.4. Properties

Variables, Key Methods and Custom Gestures

This section provides the necessary information regarding the variables and methods inside the ManoMotion SDK. You can get familiar with the ManoMotion SDK, try out the examples provided here in order to make the use of Manomotion's technology as easy as possible.

4.4.1. ManoMotion Manager

ManoMotionManager is the main class of the SDK CE and it is necessary in order to provide information regarding Hand Tracking and Gesture Recognition. It handles the process of License Initialization and provides the information of the current session in use. Moreover, it provides performance metrics at realtime, regarding the Frames Per Second (FPS) and processing time of the frame, (value in milliseconds). As seen in the diagram below, ManoMotionManager has a number of properties available for a developer to access.

Data Type	Variable
string	LicenseKey
HandInfoUnity[]	hand_infos
VisualizationInfo	visualization_info
int	Fps
int	Processing_time
int	Width
int	Height
Session	Manomotion_Session
	n



ManoSettings	ManoSettings
ManoLicense	ManoLicense

Example: Retrieving information on every frame

```
void Update()

{

    GestureInfo gestureInfo = ManomotionManager.Instance.Hand_infos[0].hand_info.gesture_info;

    TrackingInfo trackingInfo = ManomotionManager.Instance.Hand_infos[0].hand_info.tracking_info;

    Warning warning = ManomotionManager.Instance.Hand_infos[0].hand_info.warning;

    Session session = ManomotionManager.Instance.Manomotion_Session;

}
```

4.4.1.1. Hand Info

HandInfo provides the detected hand(s)* information. The developer can retrieve Tracking Information, Gesture Information and Warnings about the hand.

Tracking Information

This subclass collects all the information about hand tracking including the Bounding Box that surrounds the hand (top left and size), the interaction points (Palm Center & POI) as well as the depth estimation.

Gesture Information

This subclass collects all the information about the detected gesture. Whether the user is constantly using the same hand pose or has performed a gesture, this information is offered via the variables of ManoClass, ManoGestureContinuous, ManoGestureTrigger, HandSide and State.

Warnings

Warnings are useful information as they let the developer know when the user's hand might be getting too close to the edges or not detected at all. This is important because the SDK requires the hand of the user to be within the camera view in order to accurately understand the hand.

Data Type	Variable
TrackingInfo	tracking_info
GestureInfo	gesture_info
Warning	warning

Example: Retrieving information on every frame

```
void Update()

{
```



```
GestureInfo gestureInfo = ManomotionManager.Instance.Hand_infos[0].hand_info.gesture_info;  
  
TrackingInfo trackingInfo = ManomotionManager.Instance.Hand_infos[0].hand_info.tracking_info;  
  
Warning warning = ManomotionManager.Instance.Hand_infos[0].hand_info.warning;  
  
}
```

Need more examples about how to use the provided features? Check the video series here(<https://www.manomotion.com/guides-v2/>)

4.4.1.2. Session

Session is a struct that the SDK uses to communicate real-time information to the SDK. This entity is being sent and received every frame. It is responsible for input and output operations such as setting the the different smoothing values in order to get raw signals or more stable signals. The gesture smoother will apply changes on the continuous and trigger gestures, the ManoClass variable will continue to output raw information regarding the detection. Moreover it communicates changes in regards to phone orientation, and finally uses this to enable/disable features such as the usage of the Point Of Interaction (POI).

Data Type	Variable
Flags	flag
DeviceOrientation	orientation
AddOn	add_on
float	tracking_smoother
float	gesture_smoother
Features	enabled_features

4.4.1.3. Visualization Info

VisualizationInfo holds the data regarding the visual information of the Frame.You can access it directly from the RGB input that the SDK processes. If you want to combine our SDK with an other process that needs the frame, we highly recommend to make use of this image for consistency reasons.

Data Type	Variable
Texture2D	rgb_image

4.5. Useful Developer Information

These are variables that will prove useful for the development of your app:

- [ManomotionManager.Instance](#) : The static class of ManoMotion that includes all the tracking information



and gesture analysis of the detected hand. Use it to access all of the important variables in your scripts.

- [ManomotionManager.Instance.Hand_infos\[0\].hand_info.gesture_info](#) : The gesture analysis information of the detected hand. It includes the ManoClass of the gesture, the Continuous Gesture, the Trigger Gesture. More over, it includes the state of the hand (how much open or closed it is) and the side of the hand being detected
- [ManomotionManager.Instance.Hand_infos\[0\].hand_info.tracking_info](#) : The tracking information of the hand. It includes the bounding box information, palm centre, depth estimation and a BETA version of the POI (point of interaction) only for the pinch hand pose.
- [ManomotionManager.Instance.Hand_infos\[0\].hand_info.warning](#) : The warning information regarding the detection of the hand. Examples of this are when the hand is approaching towards the edges of the screen thus the detection quality will be affected. One of the key warnings you should be checking is if you are getting a warning of a hand not being detected.

Similar to the ManoMotionManager Singleton, make use of our other Singleton class, ManoUtils. It provides the necessary method for you to use when you want to place something on the hand.

- [ManoUtils.Instance.CalculateScreenPosition\(Vector3 position, float depth\)](#) : Moves an object in Screen Coordinates (pixels). Used primarily for 2D Canvas usecases. Similar to Camera.main.ViewportToScreen.
- [ManoUtils.Instance.CalculateWorldPosition\(Vector3 position, float depth\)](#) : Moves an object in World Coordinates (Unity units). Used primarily for 3D objects. Similar to Camera.main.ViewportToWorld.

5. Supported Devices & Requirements

Types of hardware that can support the ManoMotion technology

Here you can find a list of all of the devices that can support the ManoMotion SDK. We intend to use this list as a reference of minimum and confirmed specs. We assume that the higher tier and newer phones released will have no issue utilizing our tech. However, if you encounter an issue with your device contact us at support@manomotion.com.

Samsung:

- Galaxy S6
- Galaxy S6 Edge
- Galaxy S7
- Galaxy S7 Edge
- Galaxy S8
- Galaxy S8+
- Galaxy A5 (2017)

Apple:

- iPhone 6
- iPhone 6S
- iPhone 7 Plus
- iPhone 8
- iPad Mini 4
- iPhone X

iPhone 5, iPod Touch and other apple devices with 32bit architecture are not supported



Huawei:

- P10 Lite
- Pro Mate 9

Xiaomi:

- MI3
- Redmi Note 3
- POCOPHONE F1
- Mi 6A
- Mi8

OnePlus:

- One plus 3

Asus:

- Zenphone AR

Lenovo:

- Tab4 Ten Plus

Motorola:

- G5

5.1. Requirements

Here you can find the minimum requirements needed from a mobile device in order to support our SDK.

Android

CPU	RAM	GPU	Camera	OS
Octa-core (4x2.1 GHz Cortex-A57 & 4x1.5 GHz Cortex-A53)	3 GB	Not used	Any	Android 5.0.2



iOS

CPU	RAM	GPU	Camera	OS
Dual-core 1.4 GHz Typhoon (ARM v8-based)	1 GB	Not used	Any	iOS 8

Supported Architectures

Armeabi-v7a

5.2. Performance

The performance of the SDK is highly dependent on the device resources and the test environment. Below, there is informative data regarding the performance. Note that this information is merely informative as depending on the device, the camera sensor and the environment conditions the performance of the SDK may change.

- (iOS) Iphone XS Max: Average Processing Time 8ms
- (Android) Huawei p30: Average Processing time 12ms

6. Common Issues & Solutions in Unity

1)When importing the ManoMotion package, the prefabs appear to be missing.

(Solution) Make sure your Unity version is an official release (Beta and Alpha versions are usually not used or tested by ManoMotion). Grab the latest official version from [HERE](https://unity3d.com/get-unity/update)(<https://unity3d.com/get-unity/update>)

2)The application Does Not Work. Through the logs or UI, I am receiving the message "LICENSE_KEY_NOT_FOUND" or "LICENSE_INCORRECT_BUNDLE_ID"

(Solution) Make sure that there are no spaces or additional characters in your license key(See [HERE](#)) field or Bundle Identifier(See [HERE](#)).

3)The application does not work and I am receiving a message of Internet Required.

(Solution) Make sure that the internet speed of your phone is adequate to communicate with our service. This issue is faced when there is either a)no internet connection available in



the device. b) When the option Internet Required is not set in build settings (See Here) c) A very slow connection available in the device.

You can get more help, ideas and feedback by joining our development Discord server. Join HERE(<https://discord.gg/HpRQpG9>)