Imagine WebAR Curved Tracker

Version:	1.0.1
WebGL Demo:	https://webar.imaginerealities.com.au/ct/demo
Contact Support:	https://imaginerealities.com.au/contact-support
Publisher's Website:	https://imaginerealities.com.au
Unity Forums Thread:	Coming Soon
Discord Community:	https://discord.gg/ypNARJJEbB

Introduction

The web is one of the most promising platforms for augmented reality, because it allows users to experience AR without the need to download a standalone application. And most, if not all, WebAR plugins for Unity require developers to pay on a subscription and/or per view basis.

Imagine WebAR Curved Tracker is an augmented reality plugin for Unity WebGL which allows developers to recognize and track curved images wrapped around bottles, cans, and cups, in AR experiences for the web. This plugin also allows developers to host their own AR experiences like any other Unity WebGL build.

WebGL applications built using this plugin are able to run on both desktop and mobile browsers. And since AR is mostly experienced through mobile devices, we are giving mobile browsers a higher priority.

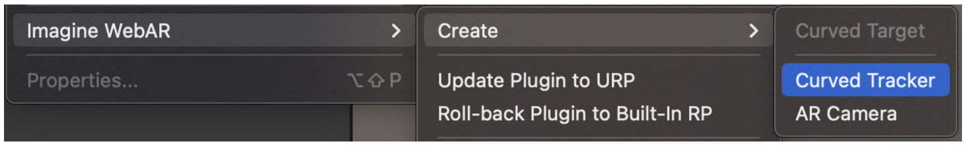
Render Pipelines supported are Built-In RP and URP. Though, some rendering features are not supported (see **Current Limitations**) or still experimental.

The plugin consists mainly of two modules: [1] a computer-vision (CV) module, written in Javascript, which uses Natural-Feature tracking. This method disregards the need of any markers, and allows developers to anchor 3D objects directly into any image (Given that this image has enough “features”. See **WebAR Best Practices**). [2] A Unity Editor module which provides the tools necessary to create “Curved Image targets” and set up your AR Scene in Unity.

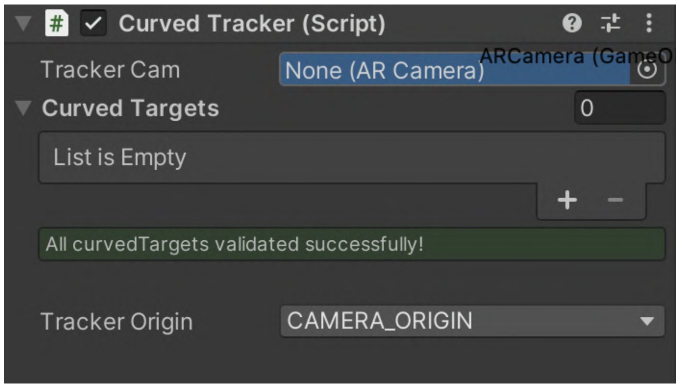
Setting up your AR Scene in Unity

A simple AR scene can be set up in a few minutes. Tutorial video can be found here
To set up your first scene:

1.) Import the plugin and create a new Scene in Unity.
2.) Create a **CurvedTracker** from **Assets>Imagine WebAR>Create>Curved Tracker**



3.) Delete the Main Camera gameobject, and create an **AR Camera** from **Assets>Imagine WebAR>Create>AR Camera.** Drag this object into the Tracker Cam property of the Curved Tracker



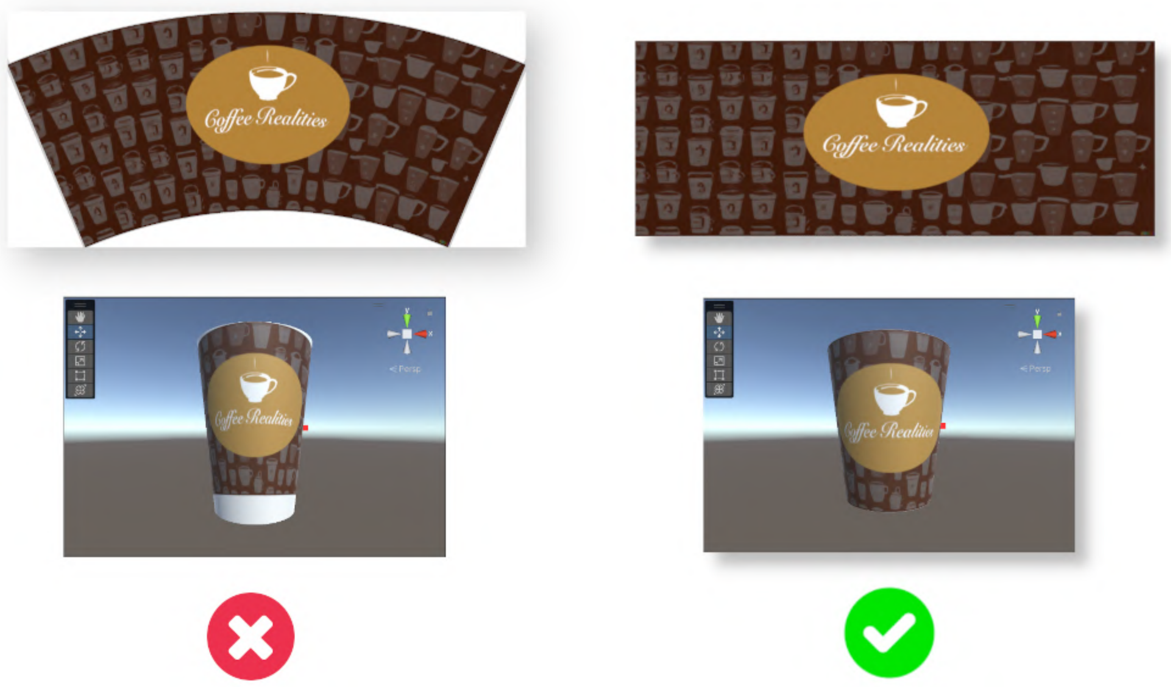
Tracker Cam: None (AR Camera) ARCamera (GameO
Curved Targets: 0
List is Empty
All curvedTargets validated successfully!
Tracker Origin: CAMERA_ORIGIN

4.) Now we are ready to add **Curved Targets** to our tracker.

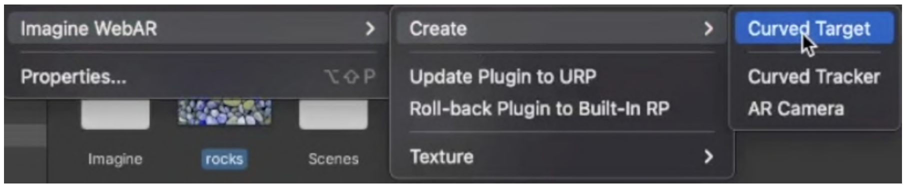
Adding Curved Targets to your Tracker

CurvedTargets are images that are wrapped into cylindrical or conical shapes. These get detected by the tracker, and overlaid with 3D objects. Any image can be used as a curved target however, tracking performance will greatly depend on the amount of “features” in the image. Images with high pixel contrasts are strong targets, while images with smooth gradients perform poorly or not at all. Please see **WebAR Best Practices** for more information.

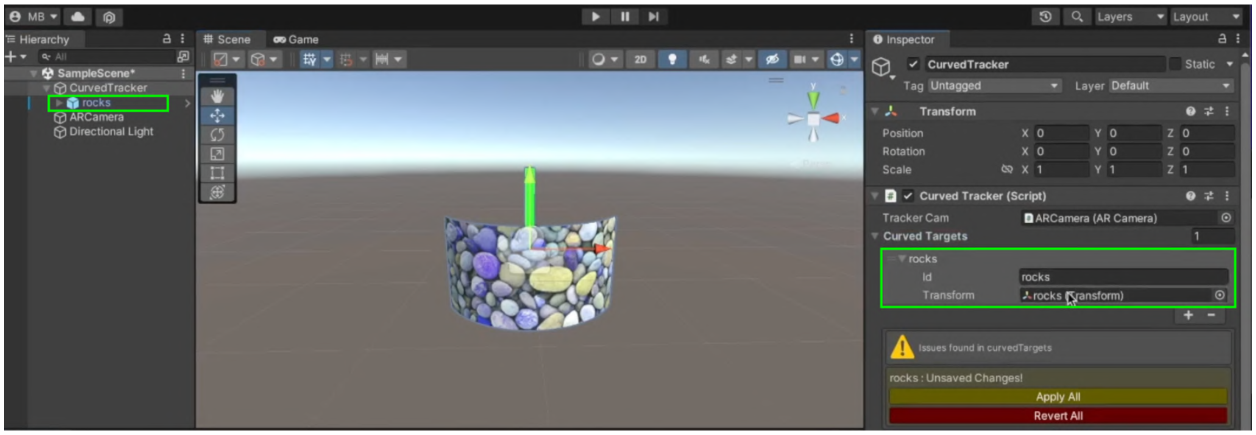
Important: For conical curved targets - make sure that you are using undistorted images. If you have a conically warped image pattern, make sure to undistort it first (See **Undistorting Warped Images**) before making it into a Curved Target.



1.) Drag/import your image in Unity.
2.) Select this image in the Unity project window, and go to **Assets>Imagine WebAR>Create>Curved Target**



3.) Note: Make sure to do this while you have your AR scene open, this CurvedTarget is automatically added to your hierarchy and to your CurvedTracker.



CurvedTracker
Tag Untagged Layer Default

Transform

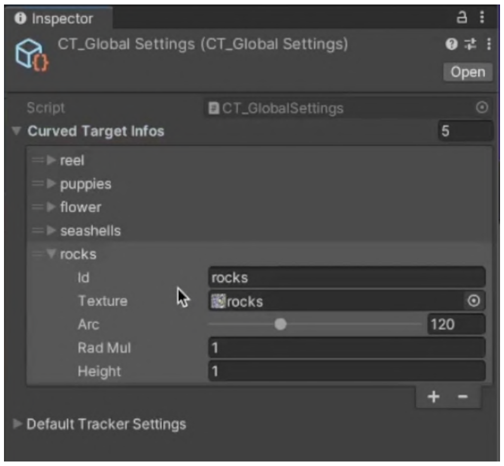
	X	Y	Z
Position	0	0	0
Rotation	0	0	0
Scale	1	1	1

Curved Tracker (Script)

Tracker Cam ARCamera (AR Camera)
Curved Targets 1
rocks
Id rocks
Transform rocks (Transform)

Issues found in curvedTargets
rocks : Unsaved Changes!
Apply All
Revert All

4.) Important: To check if your CurvedTarget is set up correctly, go to **Assets/Imagine/CurvedTracker/Resources/CT_GlobalSettings.asset** and you should see your new Image Target in **ImageTargetInfos**. And the Id should match with the one you have in your Curved Tracker.

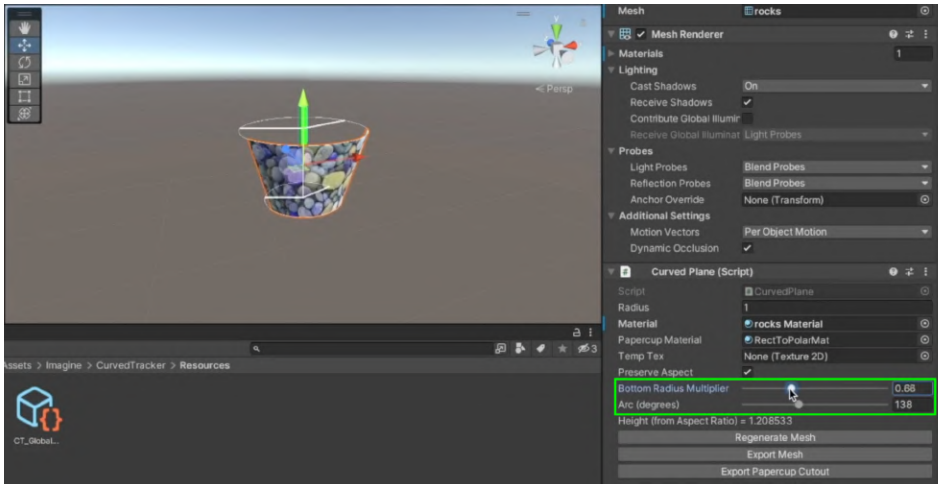


Inspector
CT_Global Settings (CT_Global Settings)
Open

Script CT_GlobalSettings
Curved Target Infos 5
reel
puppies
flower
seashells
rocks
Id rocks
Texture rocks
Arc 120
Rad Mul 1
Height 1

Default Tracker Settings

5.) Next, set your CurvedTracker's **Arc (Degrees)** and **Bottom Radius Multiplier** by dragging slider properties. See **Getting Your Curved Target's Measurements**

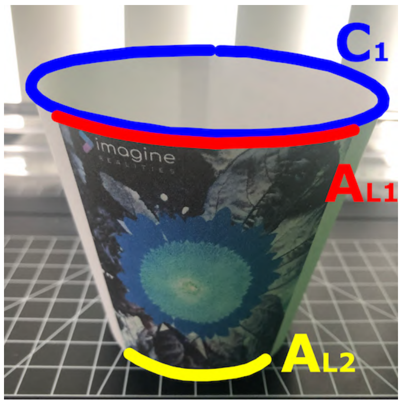


```
Mesh                          rocks
Mesh Renderer
Materials                     1
Lighting
  Cast Shadows                On
  Receive Shadows             ✓
  Contribute Global Illumir
  Receive Global Illuminat    Light Probes
Probes
  Light Probes                Blend Probes
  Reflection Probes           Blend Probes
  Anchor Override             None (Transform)
Additional Settings
  Motion Vectors              Per Object Motion
  Dynamic Occlusion           ✓
Curved Plane (Script)
Script                        CurvedPlane
Radius                        1
Material                      rocks Material
Papercup Material             RectToPolarMat
Temp Tex                      None (Texture 2D)
Preserve Aspect               ✓
Bottom Radius Multiplier      0.68
Arc (degrees)                 138
Height (from Aspect Ratio) = 1.208533
```

6.) And finally, drag in your game objects in your new Image Target object.

Getting Your Curved Target Measurements

To get the values for your **Arc (degrees)**, and **Bottom Radius Multiplier**, you will need to get three measurements of your cup.
As shown in the figure below, using a tape measure, get the **upper circumference (C_1)**, **upper arc length (A_{L1})**, and **lower arc length (A_{L2})**.



Then use the formulas below to get the values of your **Arc (degrees)**, and **Bottom Radius Multiplier**.

$$Arc(degrees) \ = \ \frac{A_{L1}}{C_1} * 360$$

$$Bottom\ Radius\ Multiplier \ = \ \frac{A_{L2}}{A_{L1}}$$

For example if we get the following measurements:

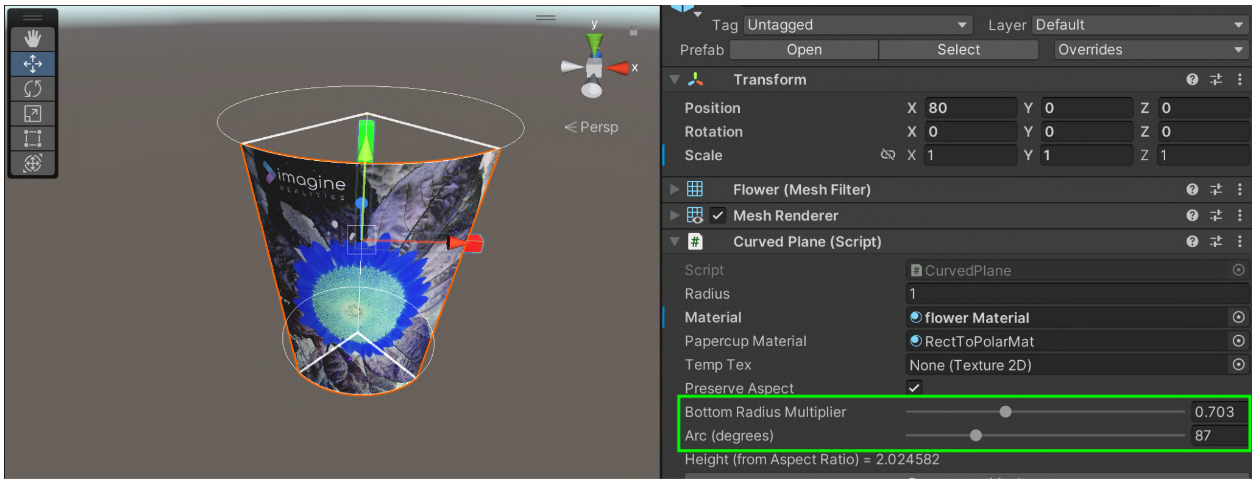
$$C_1 \ = \ 26.4\ cm$$
$$A_{L1} \ = \ 6.4\ cm$$
$$A_{L2} \ = \ 4.5\ cm$$

Then we can compute for the **Arc (degrees)**, and **Bottom Radius Multiplier** as follows:

$$Arc(degrees) \ = \ \frac{6.4}{26.4} * 360 \ = \ 87\ degrees$$

$$Bottom\ Radius\ Multiplier \ = \ \frac{4.5}{6.4} \ = \ 0.703$$

And using these values, we can now recreate our curved target in Unity.



Tag Untagged Layer Default
Prefab Open Select Overrides
Transform
Position X 80 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
Flower (Mesh Filter)
✓ Mesh Renderer
Curved Plane (Script)
Script CurvedPlane
Radius 1
Material flower Material
Papercup Material RectToPolarMat
Temp Tex None (Texture 2D)
Preserve Aspect ✓
Bottom Radius Multiplier 0.703
Arc (degrees) 87
Height (from Aspect Ratio) = 2.024582

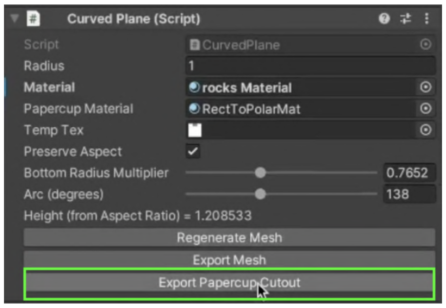
Notes:

- Cylindrical targets have a Bottom Radius Multiplier of 1
- Large Arc (degrees) values (eg. 180-360) can be less effective in terms of detection and tracking quality, so consider using lower values eg. (90-120).
- The tracker is optimized to work on normal-shaped cups, small Bottom Radius Multiplier values (eg. lower than 0.5) can get hard to detect.

Creating a Conical Cup for Testing

The plugin also allows you to export paper cup cutouts that you can use to build physical paper cups to test your applications. You can watch the video tutorial here.

1. Once you're finished setting up your curved target. Select it from the hierarchy, go to CurvedPlane component and click **Export Papercup Cutout**.

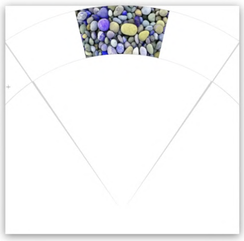


Curved Plane (Script)

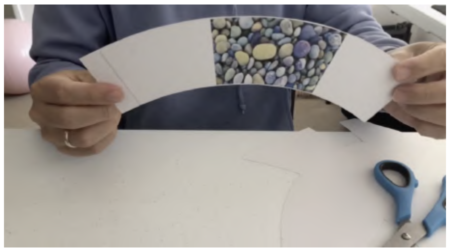
Script	CurvedPlane
Radius	1
Material	rocks Material
Papercup Material	RectToPolarMat
Temp Tex	
Preserve Aspect	✓
Bottom Radius Multiplier	0.7652
Arc (degrees)	138

Height (from Aspect Ratio) = 1.208533

2. Save the resulting image file to your preferred directory. Crop the unnecessary areas and print the pattern in a card stock



3. Follow the guidelines and cut the pattern. You can leave a flap on one side where you can attach the other end.



4. Finally, connect the two ends together with tape.

You can now test your curved targets with this physical papercup. We recommend measuring and recalculating your papercup's **Arc** and **Bottom Radius Multiplier** as there can be slight variations that can decrease the tracking quality.

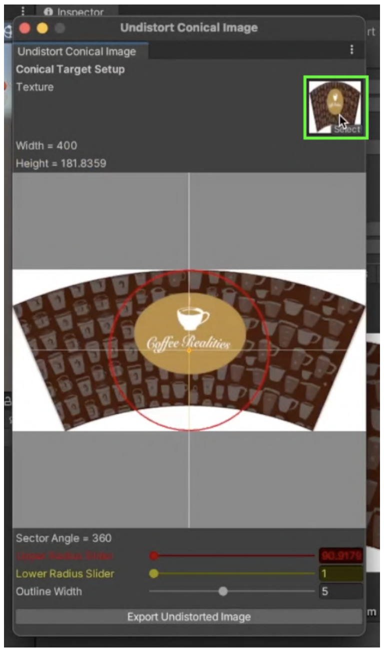
Undistorting a Warped Image

In most cases, you will be provided with distorted conical images for your curved targets. You will not be able to use these images directly and they will need to be "undistorted" first. Follow the steps below to undistort your warped images.

Coffee Realities

1. Import your image to you project. Then go to **Assets>Imagine WebAR>Texture>Undistort Conical Image**. A new popup window will open.

2. Drag your image in the texture property of this window.



Undistort Conical Image

Conical Target Setup

Texture

Width = 400

Height = 181.8359

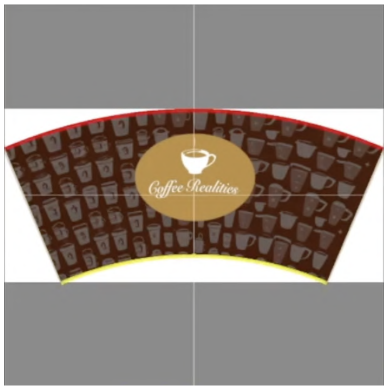
Sector Angle = 360

Upper Radius Slider 90.9179

Lower Radius Slider 1

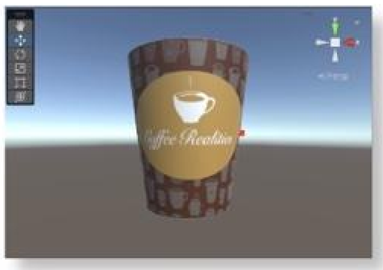
Outline Width 5

3. Then drag the **Upper Radius Slider** to align the red outline with the upper curvature of your image.
4. Next drag the **Lower Radius Slider** to align the yellow outline with the lower curvature of your image. You can increase the **Outline Width** to properly see if the lines match the curvatures. At this stage, your red and yellow lines should be tracing your image arc like shown below.



5. Finally, click **Export Undistorted Image** and save the resulting file in your Assets folder. You can now use this undistorted image as a proper curved target.

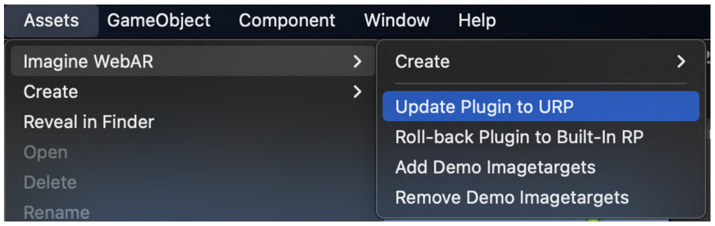
Coffee Realities



Updating the plugin to URP

Imagine WebAR also supports the Universal Render Pipeline for WebGL. To update the plugin to URP, follow the steps below:

1.) Create a new Unity project using the **3D(URP)** template.
2.) Import the plugin.
3.) Go to **Assets>Imagine WebAR>Update Plugin to URP**



4.) Wait for the reimport to finish.

Note: The plugin does not fully support all URP features - Camera HDR and Post-Processing are disabled by default. This is because the plugin is rendering the video background in javascript while the 3D objects are rendered by Unity WebGL with a transparent background. Turning these features on breaks the transparency of the canvas and the background is rendered black.
This can be partially resolved by enabling **Use Webcam Texture (Experimental)** in your TrackerCamera.

WebAR Best Practices

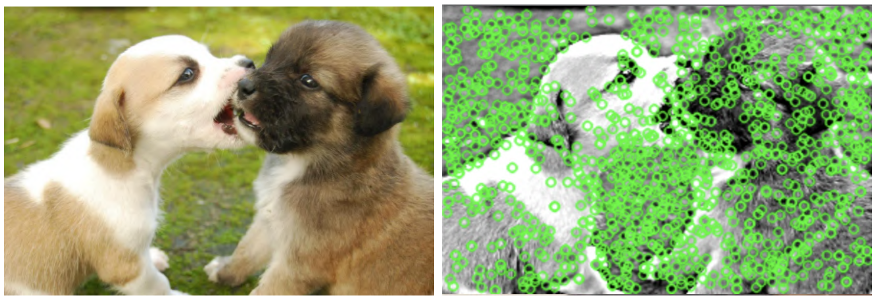
Selecting the right Image Target

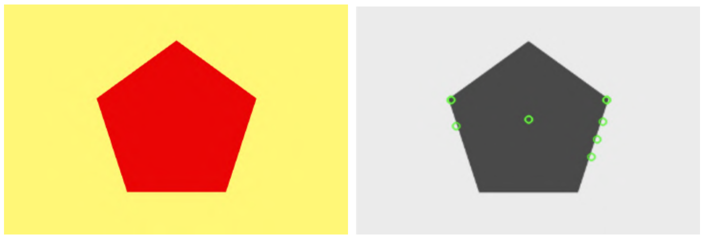
Several factors can affect the detection and tracking performance of your Image Target. An ideal Image Target is described by the following:

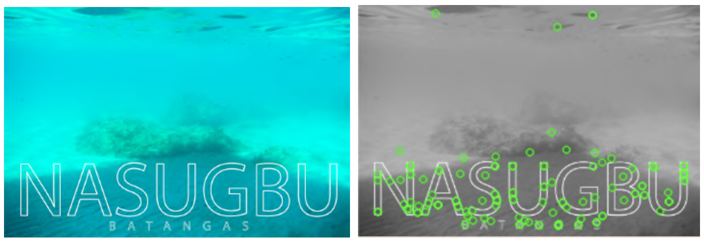
- Rich and well-distributed details across the entire image
- Excellent contrast between the bright and dark regions of the image
- No repetitive or symmetrical patterns

What are Features?

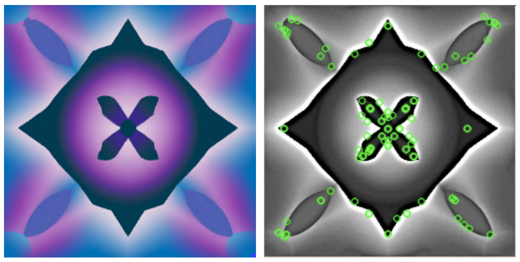
Features are sharp details such as corners and contrasting pixels in textures. Tracking images with more features are more robust to noise and jitter.


A great example of a feature-rich Image Target


A poor Image Target with very few trackable features



A poor Image Target due to lack of contrast


An Image Target with symmetrical or repeating patterns can “confuse” the tracker

<u>Physical factors affecting tracker quality</u>

Physical factors such as print media and lighting also play a vital role in the effectiveness of the tracker.

- Images printed in glossy or reflective surfaces will have tracker quality issues on some angles. It is best to print your images in matte, whenever possible.
- Environment should be well lit. The lighting conditions can easily affect how the camera sees the image thus affecting tracker quality.
- Image printout should be rigid. Creased or bent targets degrade the quality or the tracker.

<u>Lightweight AR experiences and game optimizations</u>

Another thing to consider is that WebAR runs in web browsers with very limited processing power compared to other platforms. So it is important to choose a lightweight experience and well-optimized to run even on low-tier mobile devices across a wide range of browsers and versions. Consider the following simplifications:

- Use low-res sprite sheets and/or low-poly models
- Use simple/unlit shaders whenever possible
- Avoid real-time image effects and post-processing
- Use simple animations instead of physics simulations
- Total memory consumed less than 300MB (including tracker)

<u>FPS and Overheating</u>

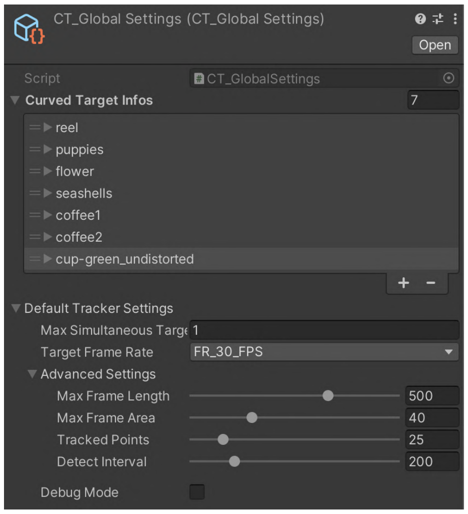
It is also very important to consider the impact of frame rate to the tendency of the device to heat-up. AR experiences, in general, consume more resources (such as camera and calculations) than standard games, thus putting more strain on device hardware and causing them to heat up, especially on high frame rates and longer play sessions. And In response to overheating, devices will cap the framerate to avoid any serious damage.

From our testing, mid-tier devices (eg. iPhone 8) is able to run a simple AR scene at 60 FPS for 3-5 minutes straight, before the frame rate starts to drop due to overheating.

Hence, it is best to keep your frame rate as low as possible (30 FPS or less is recommended for WebAR experiences in mobile). As well as designing your AR experience to be playable in small intervals (2-3 minute sessions) while keeping your game as optimized as possible.

Tracker Settings

The tracker can be customized with a couple of properties. And these default settings can be found in **Assets>Imagine>CurvedTracker>Resources>CT_GlobalSettings.asset.**



CT_Global Settings (CT_Global Settings)

Open

Script CT_GlobalSettings

Curved Target Infos 7

- reel
- puppies
- flower
- seashells
- coffee1
- coffee2
- cup-green_undistorted

Default Tracker Settings

Max Simultaneous Targe 1

Target Frame Rate FR_30_FPS

Advanced Settings

Max Frame Length 500

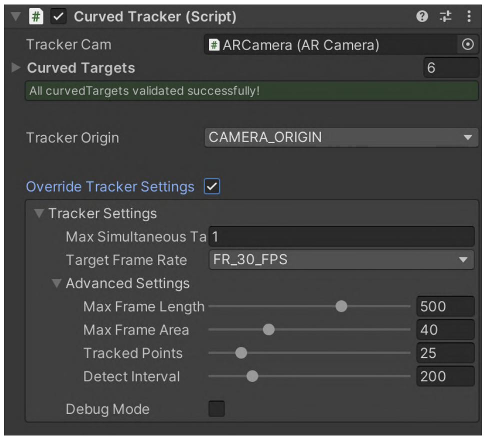
Max Frame Area 40

Tracked Points 25

Detect Interval 200

Debug Mode

They can also be overridden on a per scene basis, by enabling the **Override Tracker Settings** in your Curved Tracker object.


Curved Tracker (Script)
Tracker Cam: ARCamera (AR Camera)
Curved Targets: 6
All curvedTargets validated successfully!
Tracker Origin: CAMERA_ORIGIN
Override Tracker Settings: ✓
Tracker Settings
Max Simultaneous Ta: 1
Target Frame Rate: FR_30_FPS
Advanced Settings
Max Frame Length: 500
Max Frame Area: 40
Tracked Points: 25
Detect Interval: 200
Debug Mode: ☐

<u>Tracker Origin</u>

- Use **CAMERA_ORIGIN** to position your image targets relative to the camera at the origin.
- Use **FIRST_TARGET_ORIGIN** to position your camera relative to the first detected Image Target. This is useful for Image Target setups which are sensitive to orientation such as when using gravity, trail renderers or world particle systems.

<u>Max Simultaneous Targets</u>

- Use this setting to set the number of image targets that are allowed to be tracked simultaneously.
- Note: In theory, there is no limit in the number of Image Targets tracked simultaneously , but doing so can decrease the frame rate of your application.

<u>Target Frame Rate</u>

- Use this setting to let the tracker know your desired framerate (Normally this is between 30fps for mobile experiences).
- <u>Note:</u> Actual frame rate is still determined by the processing power of the end user device.

<u>Advanced Settings > Max Frame Length</u>

- Higher values will increase tracking quality and decrease jitters, but decreases frame rate

<u>Advanced Settings > Max Frame Area</u>

- Higher values will increase tracking quality and detectability, but decreases frame rate

<u>Advanced Settings > Detect Interval</u>

- Lower values will speed up detection time, but significantly decreases frame rate

<u>Advanced Settings > Tracked Points</u>

- Higher values will improve stability, but can decrease frame rate

Debug Mode:

- When this flag is enabled, you can press “I” on your keyboard to visualize the detected features in your image targets.

On Image Found:

- Subscribe to this UnityEvent to invoke callbacks when a specific image target is tracked.

On Image Lost:

- Subscribe to this UnityEvent to invoke callbacks when a specific image target is untracked.

Tracker - Scripting API

You can also use these API calls to control the WebGL tracker:

CurvedTracker.StartTracker

Manually start the tracker. This is useful when you need to restart the tracker after stopping it.

CurvedTracker.StopTracker

Manually stop your tracker. This is useful when you want to display non-AR related content in your scene.

ARCamera.PauseCamera

Temporarily pauses the camera.
Note: currently tracked objects will freeze in place if tracker is not stopped before calling this method.

ARCamera.UnpauseCamera

Resumes the camera.

CurvedTracker.IsImageTracked (string id)

Use this method to check if a specific image target is currently being tracked.

Current Limitations

Transparency

Transparent pixels directly in front of the video background are culled by default. This is currently the biggest limitation of the tracker. This will be resolved once Unity properly supports transparent WebGL canvas rendering.

There is currently an experimental workaround by enabling **Use Webcam Texture** flag in ImageTrackerCamera.

Update: This issue seems to have been fixed as of testing on Unity 2021.3.0f1

Frame rate and Performance

Since WebGL supports a wide variety of devices and web browsers, the actual frame rate will be determined by the processing capability of the user device. During testing we have achieved 45-55 fps on newer iPhones (iPhone X, iPhone 14) while 12-24 fps on older devices (such as iPhone 8 and Samsung S8)

CV Source code

Source code of the CV module is not included by default, mainly because we cannot yet guarantee and provide support on its functionality and performance once customized by other developers.

URP

As mentioned above, some URP features are disabled by default. Camera HDR and Post-Processing overrides the depth buffer thus disabling the transparency of our WebGL canvas.

There is currently an experimental workaround by enabling **Use Webcam Texture** flag in ImageTrackerCamera.

Editor Simulation

Unfortunately, we do not yet have any means to test AR functionality in the Unity Editor. However, we plan to implement this feature in future versions.

FAQ and General Questions

Camera does not open when I host in my website

- Make sure you are hosting on your server with https enabled. Otherwise, access to the webcam will be blocked due to security reasons.

Unity loading bar is stuck at 90%

- This is usually caused by your WebGL compression. You can set **Player Settings>Publishing Settings>Compression Format** to **Disabled.** You can also compress your build but you have to ensure that gzip(.gz) or brotli(.br) is enabled in your hosting server.

Uncaught TypeError: Cannot read properties of undefined (reading 'FOV')

- Check your **Player Settings>Resolution and Presentation** and make sure that have selected the **iTracker** WebGL template.

Image not getting detected

- Double check if your build folder includes a folder called **/targets** and that your image files are included. Also check your **index.html** if your image target is included as
  `<imagetarget id=’YOUR_ID’ src=’targets/YOUR_FILE.png’></imagetarget>`

- Double check if the Image Target is registered in your **ImageTracker** as well as in **ImageTrackerGlobalSettings** and that their ids are matching.
- Check if your Image Target follows WebAR best practices - good amount of features, high contrast and non-symmetrical etc. (See **WebAR Best Practices** for more information)
- Make sure your target images does not have any transparent pixels

Works in localhost, but webcam does not open when build is hosted

- Please make sure that you’re hosting with SSL(using https).

Unity doesn’t start - stuck in loading screen/white screen

- If you are seeing this error or something similar
  Uncaught SyntaxError: Invalid or unexpected token (at WebGL.framework.js.br:1:2)
  Try building without compression in PlayerSettings>Publishing Settings

Can I still use the plugin with irregular/circular/non-rectangular image targets/stickers?

- Yes, you can simply replace all your transparent pixels with plain black or white. Just make sure it still follows **WebAR best practices** above

Known Issues:

[Visit the #bug-reports channel in our discord]

Change Notes: