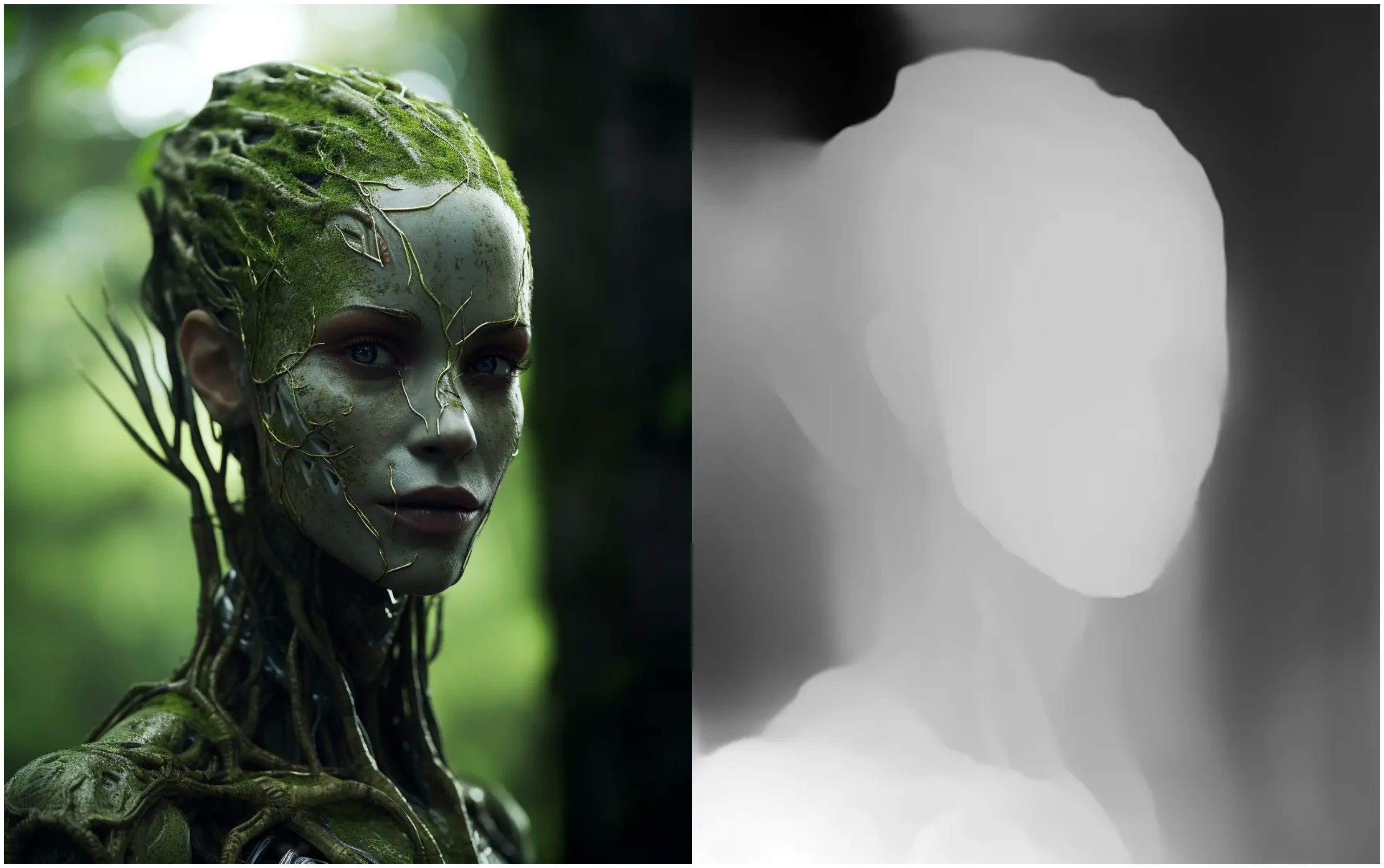
Table of Contents

Midas

Midas unlocks real-time monocular depth estimation with MiDaS models running in Unity Sentis.

To get started, please see the Getting Started page. The package also provides a number of samples that can be imported via the Package Manager.



Getting started

This page will cover the basics of using the Midas package.

Basic Usage

To start, create a new 'Midas' instance:

```
var midas = new Midas();
```

Then you can pass in a Texture2D to @Doji.AI.Depth.Midas.EstimateDepth(Texture,System.Boolean)

```
// Estimate depth from an input texture
midas.EstimateDepth(inputImage);
RenderTexture predictedDepth = midas.Result;

// ... use predictedDepth as needed
```

Finally, when you're done you should call 'Dispose()' to properly free up native memory resources.

```
midas.Dispose();
```

You can reuse the same 'Midas' instance for multiple inferences. But if you only need it once, you might want to use the 'using' statement, so you don't need to worry about disposing it:

```
using (Midas midas = new Midas()) {
    // your code
}
```

A simple example on how to use the library can also be found in the 'Basic Sample' that can be imported via the Package Manager.

Choosing a Model Type

The default model is midas_v21_small_256. You can specify the model to be used in the Midas constructor.

```
var midas = new Midas(ModelType.dpt_beit_large_384);
```

You can also change the model on an existing 'Midas' instance through the ModelType property. Changing the model automatically disposes of the existing model and initializes the new one.

```
midas.ModelType = ModelType.dpt_beit_large_384;
```

Note, that different models can have different output sizes, so changing the model might change the dimensions of the predicted depth map.

For more information on available models and when to use which, see the Models page.

Choosing a Backend

Use the Backend property to set the desired backend for model execution (GPUCompute by default). Changing the backend automatically disposes of the existing model and initializes the new one.

```
midas.Backend = BackendType.CPU;
```

Normalizing Depth

Set the NormalizeDepth property to true if you want to normalize the estimated depth values between 0 and 1. This is mainly useful for visualization.

MiDaS Models

There are variety of different MiDaS models available. To be able to use them with Unity Sentis, the official models were converted to ONNX using this colab notebook. You'll find links to the pretrained models in ONNX format below.

Overview

Model Type	Size	MiDaS Version
midas_v21_small_256	63 MB	2.1
midas_v21_384	397 MB	2.1
dpt_beit_large_512	1.34 GB	3.1
dpt_beit_large_384	1.34 GB	3.1
dpt_beit_base_384	450 MB	3.1
dpt_swin2_large_384	832 MB	3.1
dpt_swin2_base_384	410 MB	3.1
dpt_swin2_tiny_256	157 MB	3.1
dpt_swin_large_384	854 MB	3.1
dpt_next_vit_large_384	267 MB	3.1
dpt_levit_224	136 MB	3.0
dpt_large_384	1.27 GB	3.0

Usage

To keep the package size reasonable, only the midas_v21_small_256 model is included with the package when downloading from the Asset Store. To use other models you have to downloaded them first.

When you create an instance of the Midas class you can pass the ModelType in the constructor to choose which model to use. In the Unity Editor if you're about to use a model that is not yet present, you will automatically be prompted to allow the file download.

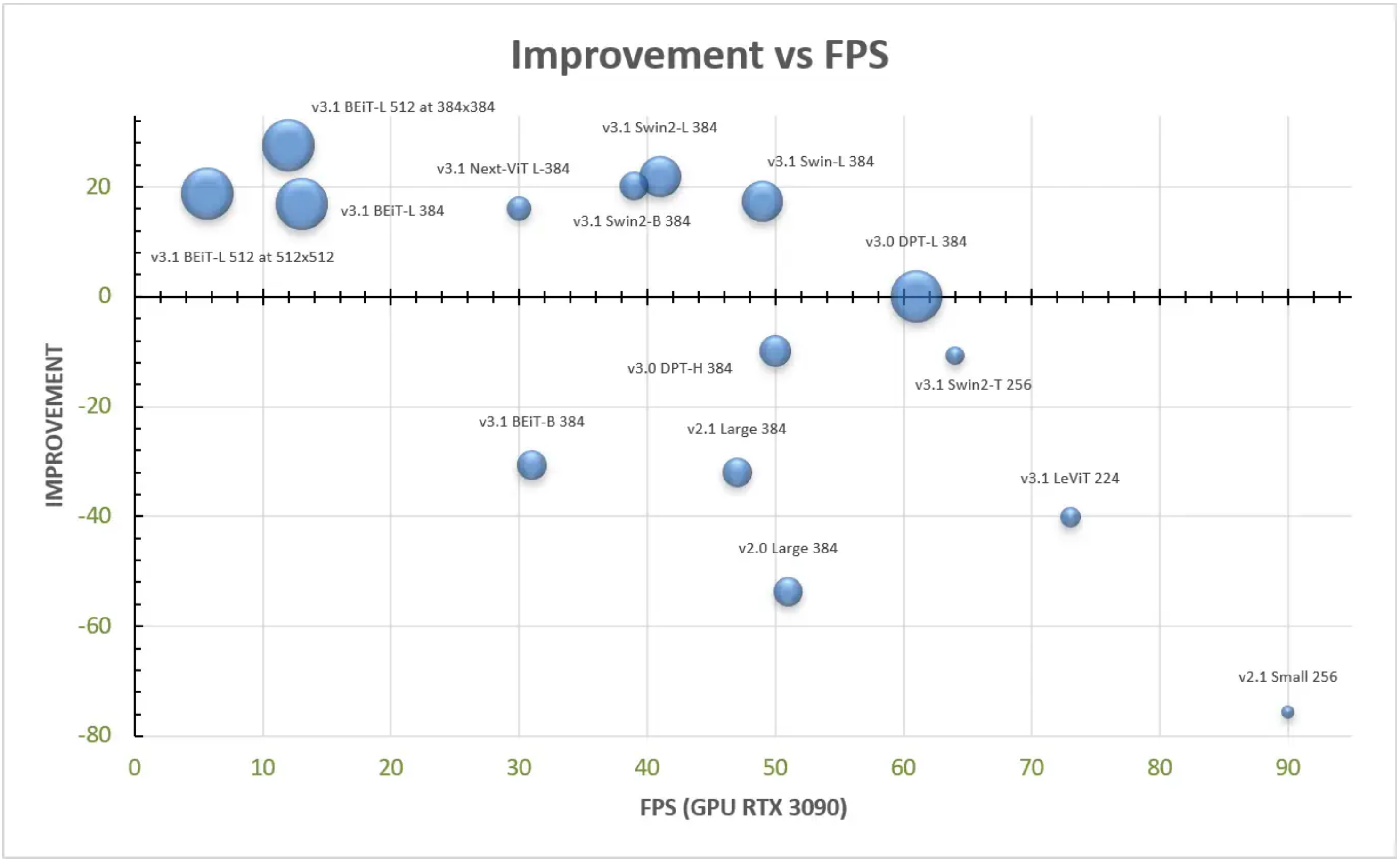
Otherwise you can always manually download the ONNX models from the links above and place them inside the Resources/ONNX folder.

Which Model To Use

You should choose the appropriate model type based on your requirements, considering factors such as accuracy, model size and performance.

The available models usually have a tradeoff between memory & performance and the accuracy/quality of the depth estimation.

The official MiDaS documentation does a fairly good job of showcasing the different capabilities, so I'll just copy the performance overview here:



Generally, if you have a hard realtime requirement, e.g. when you want to do depth estimation at runtime or on mobile devices, you may want to use smaller models like midas_v21_small_256 or dpt_swin2_tiny_256. If you need best quality (editor tools) or depth estimation needs to happen just once, you might be able to use models like dpt_swin2_large_384 or dpt_beit_large_384. Keep in mind the larger memory requirements for these model though.

Namespace Doji.AI.Depth

Classes

Midas

A class that allows to run Midas models to do monocular depth estimation.

Enums

ModelType

All the supported MiDaS models.

Class Midas

Namespace: Doji.AI.Depth

Assembly: Doji.Midas.dll

A class that allows to run Midas models to do monocular depth estimation.

```
public class Midas : IDisposable
```

Inheritance

object ← Midas

Implements

IDisposable

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

Midas(ModelType)

Initializes a new instance of MiDaS.

```
public Midas(ModelType modelType = ModelType.midas_v21_small_256)
```

Parameters

`modelType` ModelType

Properties

Backend

Which Unity.Sentis.BackendType to run the model with.

```
public BackendType Backend { get; set; }
```

Property Value

BackendType

ModelType

Which of the MiDaS models to run.

```
public ModelType ModelType { get; set; }
```

Property Value

ModelType

NormalizeDepth

Whether to normalize the estimated depth.

```
public bool NormalizeDepth { get; set; }
```

Property Value

bool

Remarks

MiDaS predicts depth values as inverse relative depth. (small values for far away objects, large values for near objects) If NormalizeDepth is enabled, these values are mapped to the (0, 1) range, which is mostly useful for visualization.

Result

A RenderTexture that contains the estimated depth.

```
public RenderTexture Result { get; set; }
```

Property Value

RenderTexture

Methods

Dispose()

```
public void Dispose()
```

EstimateDepth(Texture, bool)

```
public void EstimateDepth(Texture input, bool autoResize = true)
```

Parameters

`input` Texture

`autoResize` bool

Enum ModelType

Namespace: Doji.AI.Depth

Assembly: Doji.Midas.dll

All the supported MiDaS models.

```
public enum ModelType
```

Fields

`Unknown = -1`

`midas_v21_small_256 = 0`

63 MB MiDaS v2.1 model.

`midas_v21_384 = 1`

397 MB MiDaS v2.1 model.

`dpt_beit_large_512 = 2`

1.34 GB MiDaS v3.1 model with a beitl16_512 backbone.

`dpt_beit_large_384 = 3`

1.34 GB MiDaS v3.1 model with a beitl16_384 backbone.

`dpt_beit_base_384 = 4`

450 MB MiDaS v3.1 model with a beitb16_384 backbone.

`dpt_swin2_large_384 = 5`

832 MB MiDaS v3.1 model with a swin2l24_384 backbone.

`dpt_swin2_base_384 = 6`

410 MiDaS v3.1 model with a swin2b24_384 backbone.

`dpt_swin2_tiny_256 = 7`

157 MiDaS v3.1 model with a swin2t16_256 backbone.

`dpt_swin_large_384 = 8`

854 MB MiDaS v3.1 model with a swinl12_384 backbone.

`dpt_next_vit_large_384 = 9`

267 MB MiDaS v3.1 model with a next_vit_large_6m backbone.

`dpt_levit_224 = 10`

136 MB MiDaS v3.0 model with a levit_384 backbone.

`dpt_large_384 = 11`

1.27 GB MiDaS v3.0 model with a vitl16_384 backbone.