# Robot Pose Estimation in Camera Frame

IMUSE: Project Documentation

By: Akshaya Agrawal

# Installation Guide

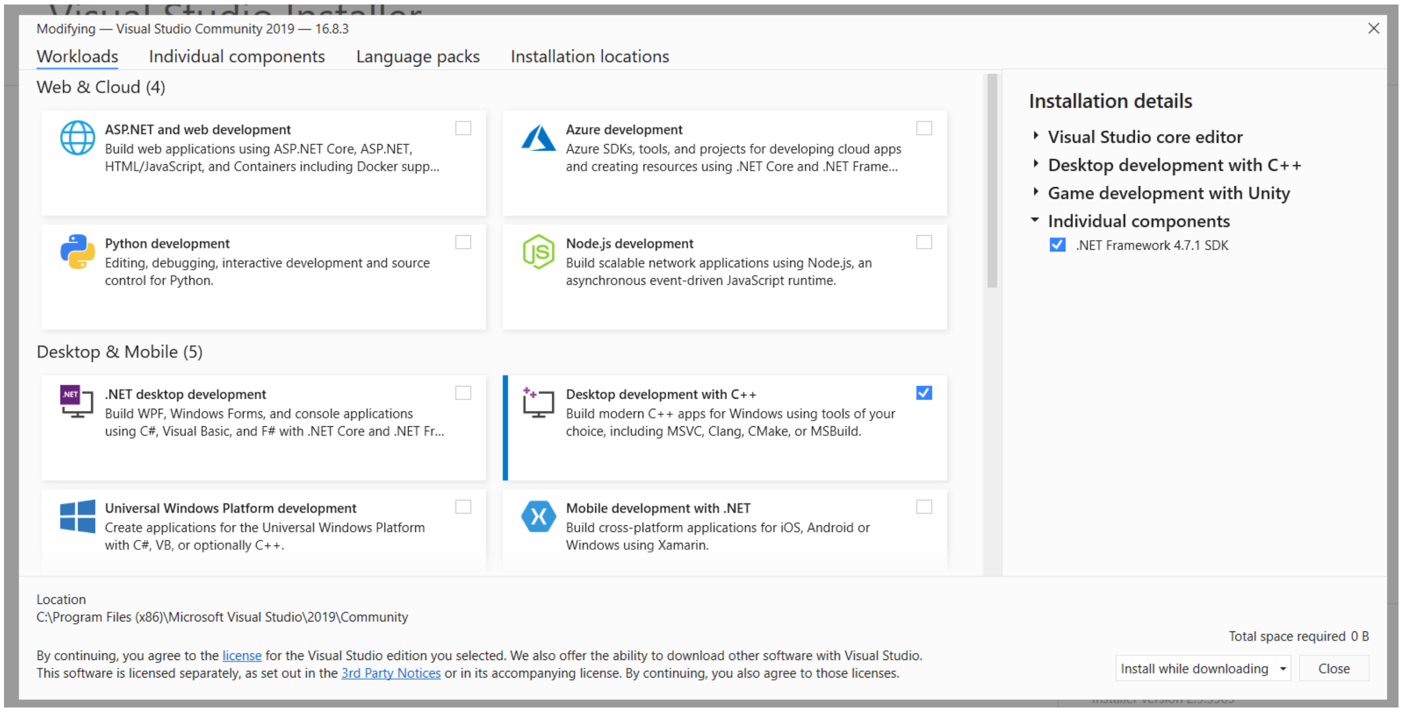
## Synthetic Data Generation

### Setup the Project

Installing Dataset Synthesizer Project on Unreal Engine.

#### Windows Specific

1. Modify the installed Visual Studio
    a. Open Visual Studio Installer -> Click Modify -> Select Desktop development using C++ -> Install



Modifying — Visual Studio Community 2019 — 16.8.3

Web & Cloud (4)

- ASP.NET and web development — Build web applications using ASP.NET Core, ASP.NET, HTML/JavaScript, and Containers including Docker supp...
- Azure development — Azure SDKs, tools, and projects for developing cloud apps and creating resources using .NET Core and .NET Frame...
- Python development — Editing, debugging, interactive development and source control for Python.
- Node.js development — Build scalable network applications using Node.js, an asynchronous event-driven JavaScript runtime.

Desktop & Mobile (5)

- .NET desktop development — Build WPF, Windows Forms, and console applications using C#, Visual Basic, and F# with .NET Core and .NET Fr...
- [✓] Desktop development with C++ — Build modern C++ apps for Windows using tools of your choice, including MSVC, Clang, CMake, or MSBuild.
- Universal Windows Platform development — Create applications for the Universal Windows Platform with C#, VB, or optionally C++.
- Mobile development with .NET — Build cross-platform applications for iOS, Android or Windows using Xamarin.

Installation details

- Visual Studio core editor
- Desktop development with C++
- Game development with Unity
- Individual components
    - [✓] .NET Framework 4.7.1 SDK

Location
C:\Program Files (x86)\Microsoft Visual Studio\2019\Community

By continuing, you agree to the license for the Visual Studio edition you selected. We also offer the ability to download other software with Visual Studio. This software is licensed separately, as set out in the 3rd Party Notices or in its accompanying license. By continuing, you also agree to those licenses.

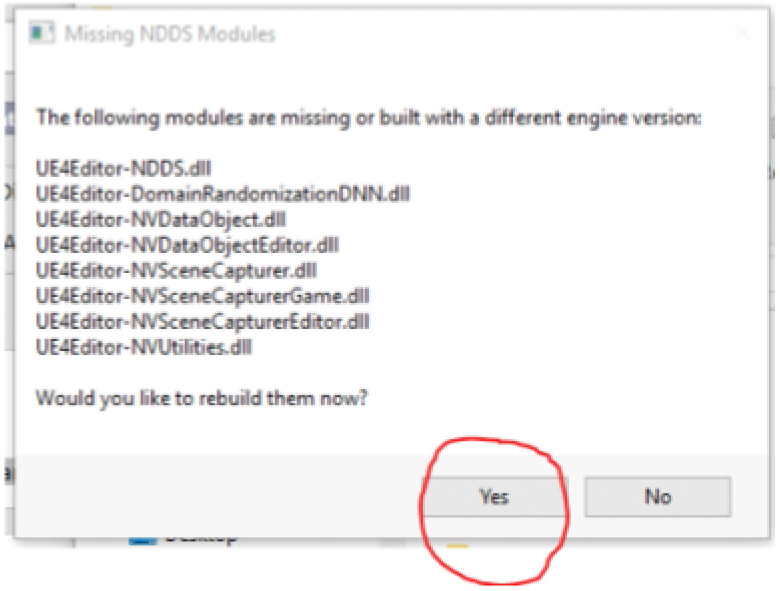
Total space required 0 B

2. Install 4.22.3 version of Unreal Engine.

## Importing Project

Link to UE4 Project:
https://drive.google.com/drive/folders/1CUETHB9jxFArTvCSdqj7bxqgkEVIoCUn?usp=sharing

1. Download the project from the above link.
2. Find NDDS.uproject file inside the source folder in the Dataset_Synthesizer folder.
3. Run it by double clicking.
4. Select yes on the popup for rebuilding the binaries.



Missing NDDS Modules

The following modules are missing or built with a different engine version:

UE4Editor-NDDS.dll
UE4Editor-DomainRandomizationDNN.dll
UE4Editor-NVDataObject.dll
UE4Editor-NVDataObjectEditor.dll
UE4Editor-NVSceneCapturer.dll
UE4Editor-NVSceneCapturerGame.dll
UE4Editor-NVSceneCapturerEditor.dll
UE4Editor-NVUtilities.dll

Would you like to rebuild them now?

[✓ Yes] No

## Run the Project

Once the Project is loaded, open the Prototype-2Level Level.

1. For defining the path for data generation:
   a. Select SceneCapturer_AllFeatureExtractors -> Details -> Scene Data Handler -> Save Path -> Root Captured Directory Path -> <select the folder where you want to generate the data>

2. For changing the number of frames to capture according to requirement:
   a. Select SceneCapturer_AllFeatureExtractors -> Details -> Max Number of Frames to Capture -> <specify number of frames>

Finally Play the Level!! Data will be generated at the defined data generation path.

# Keypoint Detection Deep Learning Neural Network

## Setup the Project

1. Clone the DREAM folder at: https://drive.google.com/drive/folders/1KX4LCYY_sLol0nt-HJnS2EoEqh62-mZl?usp=sharing

2. Install all the dependencies using:

```
pip install . -r requirements.txt
```

## Run the Project

### Training

Below is an example for training a DREAM-vgg-Q model for the Kinova robot:

```
python train_network.py -i data/sample_data -t 0.8 -m manip_configs/kinova.yaml -ar arch_configs/dream_vgg_q.yaml -e 25 -lr 0.00015 -b 128 -w 16 -o <path/to/output_dir/>
```

```
usage: train_network.py [-h] -i INPUT_DATA_PATH [-t TRAINING_DATA_FRACTION] -m
        MANIPULATOR_CONFIG_PATH [-o OUTPUT_DIR] [-f] -ar
        ARCHITECTURE_CONFIG -e EPOCHS -b BATCH_SIZE
        [-z {adam,sgd}] [-lr LEARNING_RATE] [-not-a]
        [-w NUM_WORKERS] [-g GPU_IDS [GPU_IDS ...]]
        [-s RANDOM_SEED] [-v] [-r]
```

### Inference

Run the network on a single image to display detected 2D keypoints.

```
python network_inference.py -i <path/to/network.pth> -m <path/to/image.png>
```

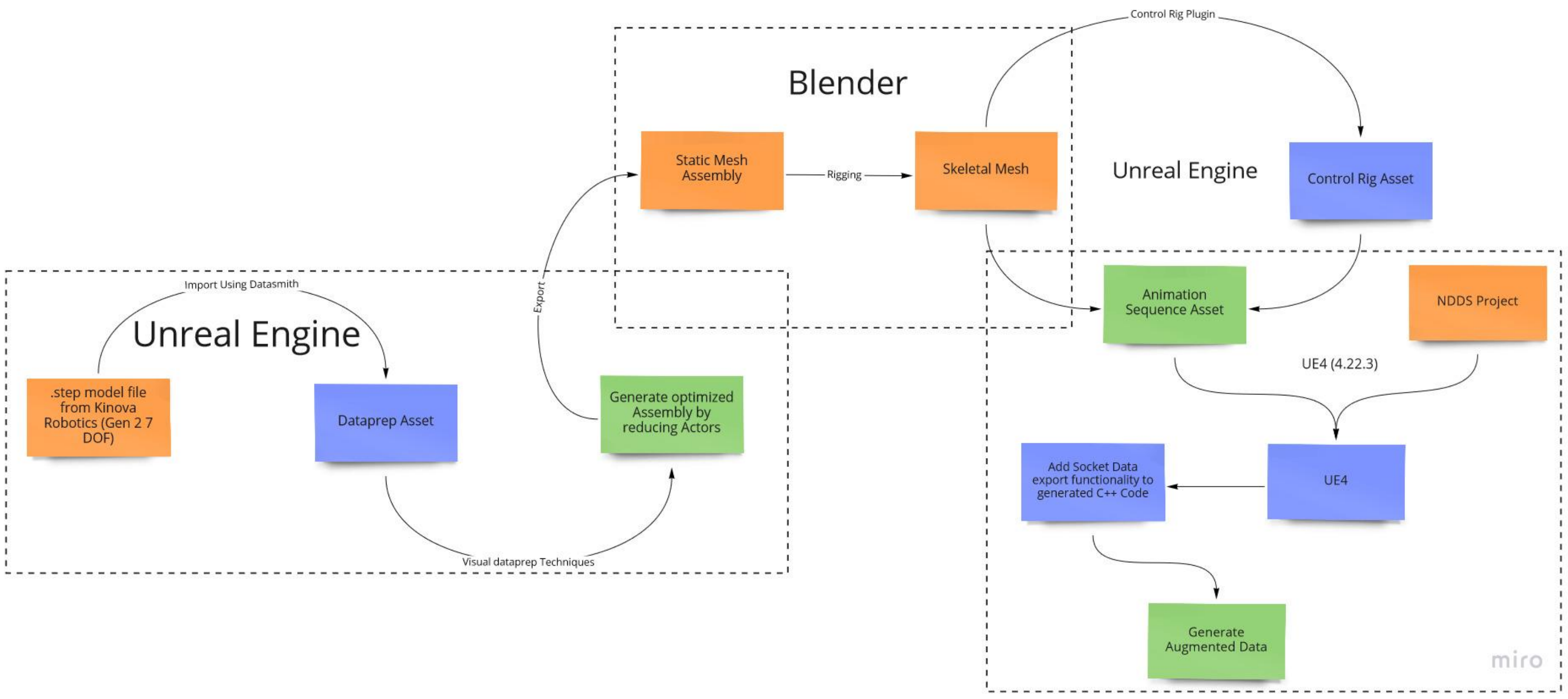
```
usage: network_inference.py [-h] -i INPUT_PARAMS_PATH [-c INPUT_CONFIG_PATH]
        -m IMAGE_PATH [-k KEYPOINTS_PATH]
        [-g GPU_IDS [GPU_IDS ...]]
        [-p IMAGE_PREPROC_OVERRIDE]
```

## Software architecture

The entire project is divided into two sections.

1. Synthetic Data Generation
2. Keypoint Detection Neural Network

## Synthetic Data Generation

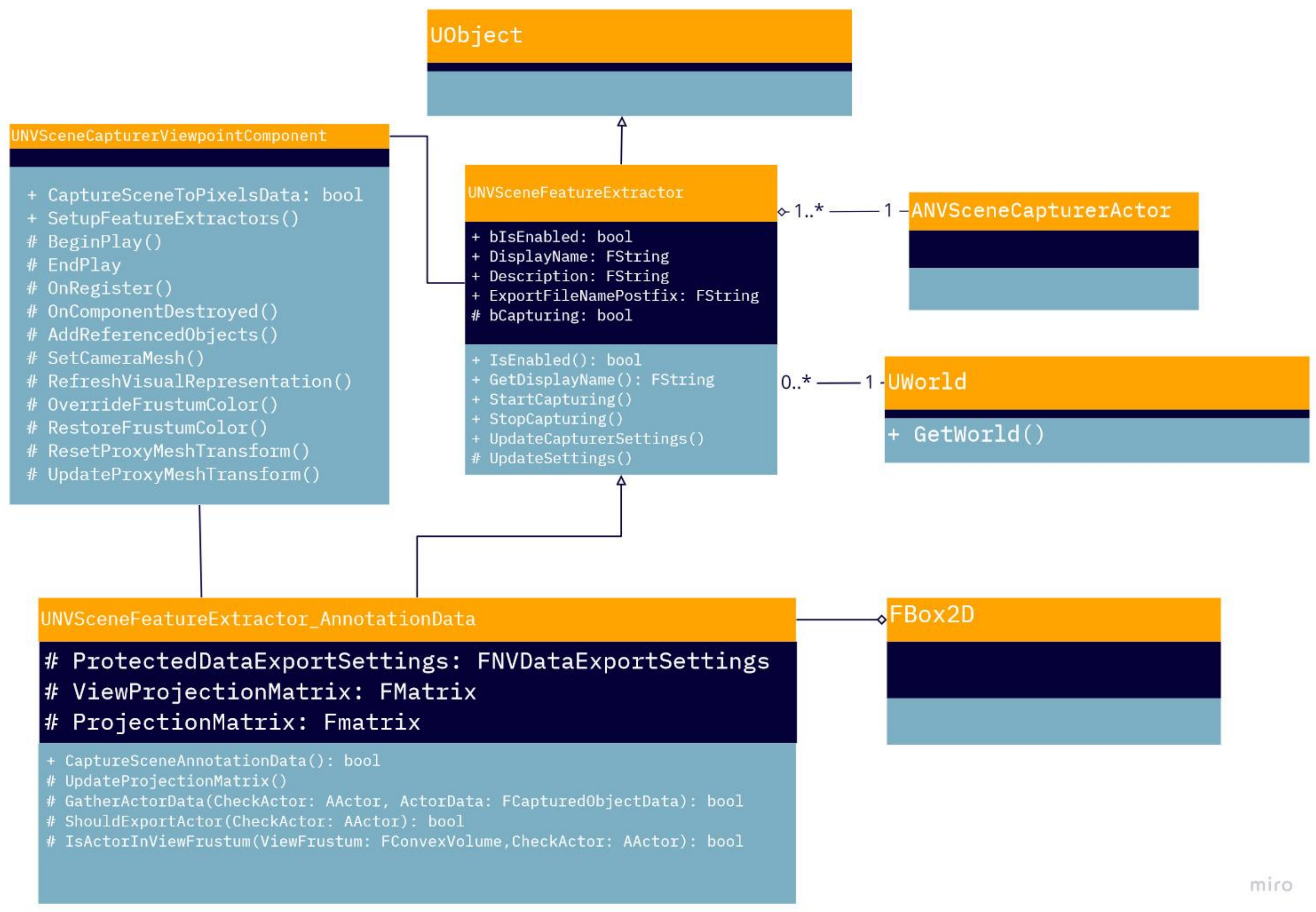


For Synthetic data generation, there are 4 major blocks:

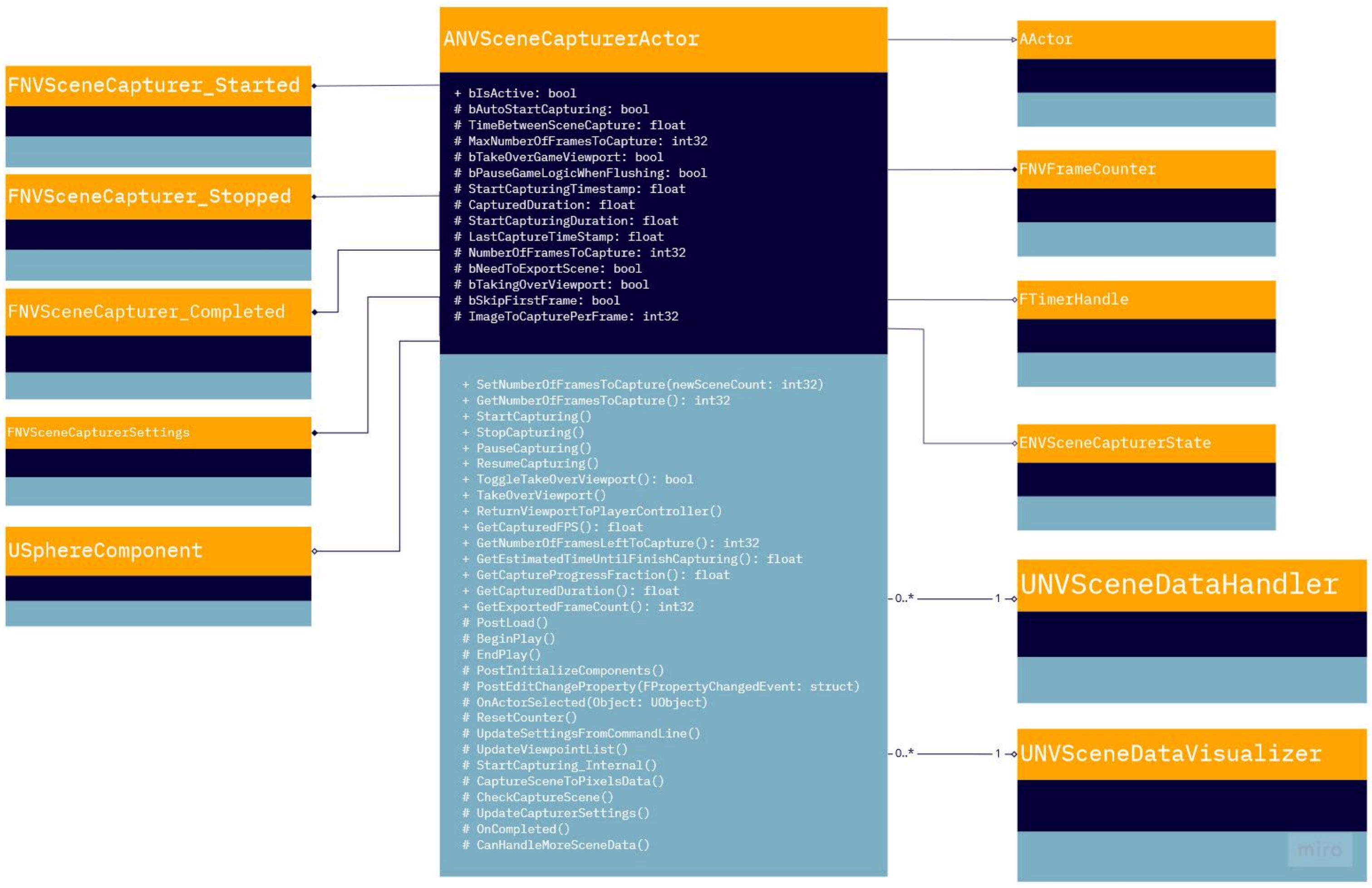
1. Generating the Static Mesh in Unreal Engine
2. Generating the Skeletal Mesh in Blender
3. Generating the Animation Sequence Asset in Unreal Engine using Control Rig and Animation generation tools.
4. Generating Augmented data by modifying the open source Dataset Synthesizer Project by NVIDIA.

To export the keypoints locations (socket Data) corresponding to each of the joints of the robotics arm we have edited the C++ source code of the Dataset Synthesizer Project.

UNVSceneFeatureExtractor_AnnotationData is the major class that exports the feature data to the .json file. Please refer to the below class diagrams to get a better Understanding.


UObject
UNVSceneCapturerViewpointComponent
+ CaptureSceneToPixelsData: bool
+ SetupFeatureExtractors()
# BeginPlay()
# EndPlay
# OnRegister()
# OnComponentDestroyed()
# AddReferencedObjects()
# SetCameraMesh()
# RefreshVisualRepresentation()
# OverrideFrustumColor()
# RestoreFrustumColor()
# ResetProxyMeshTransform()
# UpdateProxyMeshTransform()
UNVSceneFeatureExtractor
+ bIsEnabled: bool
+ DisplayName: FString
+ Description: FString
+ ExportFileNamePostfix: FString
# bCapturing: bool
+ IsEnabled(): bool
+ GetDisplayName(): FString
+ StartCapturing()
+ StopCapturing()
+ UpdateCapturerSettings()
# UpdateSettings()
1..*
1
ANVSceneCapturerActor
0..*
1
UWorld
+ GetWorld()
UNVSceneFeatureExtractor_AnnotationData
# ProtectedDataExportSettings: FNVDataExportSettings
# ViewProjectionMatrix: FMatrix
# ProjectionMatrix: Fmatrix
+ CaptureSceneAnnotationData(): bool
# UpdateProjectionMatrix()
# GatherActorData(CheckActor: AActor, ActorData: FCapturedObjectData): bool
# ShouldExportActor(CheckActor: AActor): bool
# IsActorInViewFrustum(ViewFrustum: FConvexVolume,CheckActor: AActor): bool
FBox2D
miro

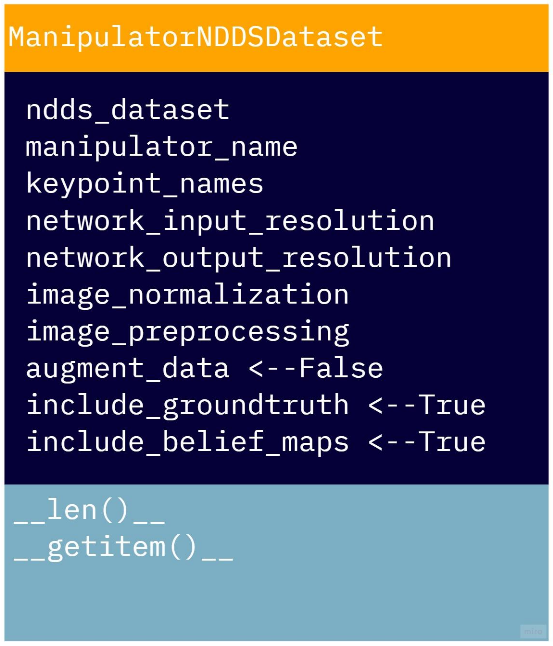
**Class Diagram 1**


ANVSceneCapturerActor
+ bIsActive: bool
# bAutoStartCapturing: bool
# TimeBetweenSceneCapture: float
# MaxNumberOfFramesToCapture: int32
# bTakeOverGameViewport: bool
# bPauseGameLogicWhenFlushing: bool
# StartCapturingTimestamp: float
# CapturedDuration: float
# StartCapturingDuration: float
# LastCaptureTimeStamp: float
# NumberOfFramesToCapture: int32
# bNeedToExportScene: bool
# bTakingOverViewport: bool
# bSkipFirstFrame: bool
# ImageToCapturePerFrame: int32
+ SetNumberOfFramesToCapture(newSceneCount: int32)
+ GetNumberOfFramesToCapture(): int32
+ StartCapturing()
+ StopCapturing()
+ PauseCapturing()
+ ResumeCapturing()
+ ToggleTakeOverViewport(): bool
+ TakeOverViewport()
+ ReturnViewportToPlayerController()
+ GetCapturedFPS(): float
+ GetNumberOfFramesLeftToCapture(): int32
+ GetEstimatedTimeUntilFinishCapturing(): float
+ GetCaptureProgressFraction(): float
+ GetCapturedDuration(): float
+ GetExportedFrameCount(): int32
# PostLoad()
# BeginPlay()
# EndPlay()
# PostInitializeComponents()
# PostEditChangeProperty(FPropertyChangedEvent: struct)
# OnActorSelected(Object: UObject)
# ResetCounter()
# UpdateSettingsFromCommandLine()
# UpdateViewpointList()
# StartCapturing_Internal()
# CaptureSceneToPixelsData()
# CheckCaptureScene()
# UpdateCapturerSettings()
# OnCompleted()
# CanHandleMoreSceneData()
FNVSceneCapturer_Started
FNVSceneCapturer_Stopped
FNVSceneCapturer_Completed
FNVSceneCapturerSettings
USphereComponent
AActor
FNVFrameCounter
FTimerHandle
ENVSceneCapturerState
0..*
1
UNVSceneDataHandler
0..*
1
UNVSceneDataVisualizer
miro

**Class Diagram 2**

## Keypoint Detection Neural Network

We have edited the open source DREAM project code that is available on Github for training the Neural Network. Below is the Class Structure for the ManipulatorNDDSDataset class that enables us to create a pair of input output tensors for the training Network.



```
ManipulatorNDDSDataset

ndds_dataset
manipulator_name
keypoint_names
network_input_resolution
network_output_resolution
image_normalization
image_preprocessing
augment_data <--False
include_groundtruth <--True
include_belief_maps <--True

__len()__
__getitem()__
```

Class Structure

Further a training network and inference scripts are written in python using pyTorch library.