Unreal Engine Testbed for Computer Vision of Tall Lunar Tower Assembly

Brian Notosubagyo,[1] Matthew K. Mahlin,[2] and Jacob T. Cassady[3]
NASA Langley Research Center, Hampton, VA 23681

The Tall Lunar Tower project at the NASA Langley Research Center is focused on the design, modeling, fabrication, and testing of a supervised autonomously assembled engineering development unit for tall lunar towers. The lunar south pole environment poses many challenges for robotic assembly of the tall tower, particularly to computer vision camera systems due to a high-contrast lighting environment. This paper will present an Unreal Engine 5 video game engine Lunar South Pole Lighting Testbed to simulate realistic lunar lighting conditions for synthetic image generation. The fidelity of the simulation environment is investigated by comparing the accuracy of computer vision models trained using synthetic image data and trained from real image data collected in a lunar analog environment.

I. Introduction

The Tall Lunar Tower (TLT) project Engineering Development Unit (EDU) is being developed at the NASA Langley Research Center (LaRC) to demonstrate supervised autonomous robotic tower assembly for energy collection and communication relays at the lunar south pole. At the extreme latitudes near the lunar south pole, the solar disk does not rise above 3° over the horizon and lunar night can last 14 Earth days or longer. However, the duration of lunar night can be reduced to just two days at higher elevations [1]. The high-contrast lighting conditions at the lunar south pole pose a significant challenge to computer vision camera systems needed for rover navigation and similarly for robotic assembly operations [2]. Development of physical lunar analog environments to emulate lunar lighting conditions requires large test areas, room modification for accurate computer vision feature detection, and landscape creation [3]. To avoid the expense and overhead of developing these environments, previous simulations have been developed in simulation platforms such as Gazebo to generate high-fidelity synthetic images for computer vision systems [2].

This paper presents an alternate lunar simulation environment that utilizes the commercially available Unreal Engine 5 (UE5) [4][*] video game engine to create an in-house virtual test bed called the Lunar South Pole Lighting Testbed (Lunar-SPLIT). The simulation environment's purpose is to generate high-fidelity synthetic images for computer vision systems with a robotic assembly use case. Synthetic images were generated for training data sets to be used in an You Only Look Once (YOLO) object-detection model [5] that detects if a rivet fastener has been installed during the inspection stage of the TLT strut robotic assembly. A YOLO version 5 (YOLOv5) [6] object-detection model was trained with synthetic image datasets collected via a UE5 virtual camera game asset with camera properties assigned to a Lens File. The Lens File provides the same specifications of a physical Intel RealSense D405 stereo camera resulting in a virtual camera digital twin (VCDT). In this paper, the Intel RealSense D405 stereo camera is denoted by "D405 camera."

This paper begins with Section II which describes the development of the UE5 Lunar-SPLIT with lunar south pole terrain and lighting generation from NASA elevation maps and ephemeris data. Next, Section III presents the development of a lunar analog environment (LAE) developed at the University of Southern California that uses a physical D405 camera to collect images to train a YOLOv5 object-detection model. Afterwards, Section IV describes

[1] Pathways Student Trainee, Structural Mechanics and Concepts Branch, AIAA Member
[2] Research Aerospace Engineer, Structural Mechanics and Concepts Branch, AIAA Member
[3] Computer Engineer, Flight Software Systems Branch, AIAA Member
[*] The use of trademarks or names of manufacturers in this report is for accurate reporting and does not constitute an official endorsement, either expressed or implied, of such products or manufacturers by the National Aeronautics and Space Administration

the image data collection process in the LAE, the synthetic image data collection process in the UE5 Lunar-SPLIT, and the YOLOv5 object detection model deployment in the LAE. Section V describes the UE5 Lunar-SPLIT scene finetuning for synthetic image training to match the visual fidelity of the LAE. This paper concludes in Section VI with a comparison and discussion of confidence scores and detection of inspection cases of the YOLOv5 object detection model trained with synthetic image data and real image data collected in the LAE.

II. Unreal Engine 5 Lunar South Pole Lighting Testbed

The virtual environment was developed in Unreal Engine version 5.0.3 featuring the lunar south pole terrain and leveraging several software tools that enable the realistic lighting of large environments. The UE5 video game engine provides two important features: Nanite virtualized geometry (NVG) and Lumen global illumination and reflections (LGIR). NVG enables the use of a new internal mesh format and rendering technology to render pixel-scale detail and high object counts and renders only details that can be perceived [7]. LGIR is the UE5 fully dynamic global illumination and reflections system that renders diffuse interreflection with infinite bounces and indirect specular reflections in large, detailed environments at scales ranging from millimeters to kilometers [8].

This paper describes two methods used by the Digital Lunar Exploration Sites Unreal Simulation Tool (DUST) team at NASA Johnson Space Center to develop a photorealistic lunar environment. The first method imported detailed lunar south pole terrain data generated from laser altimeters [9]. The second method positioned the directional light game asset that emulates the Sun using data from the NASA Jet Propulsion Laboratory (JPL) Spacecraft, Planet, Instrument, Camera-matrix, Events (SPICE) ephemeris data kernels [9]. A screen capture from the resulting UE5 Lunar-SPLIT is shown in Fig. 1, which depicts the high-contrast lighting environment where synthetic image training data sets were generated.

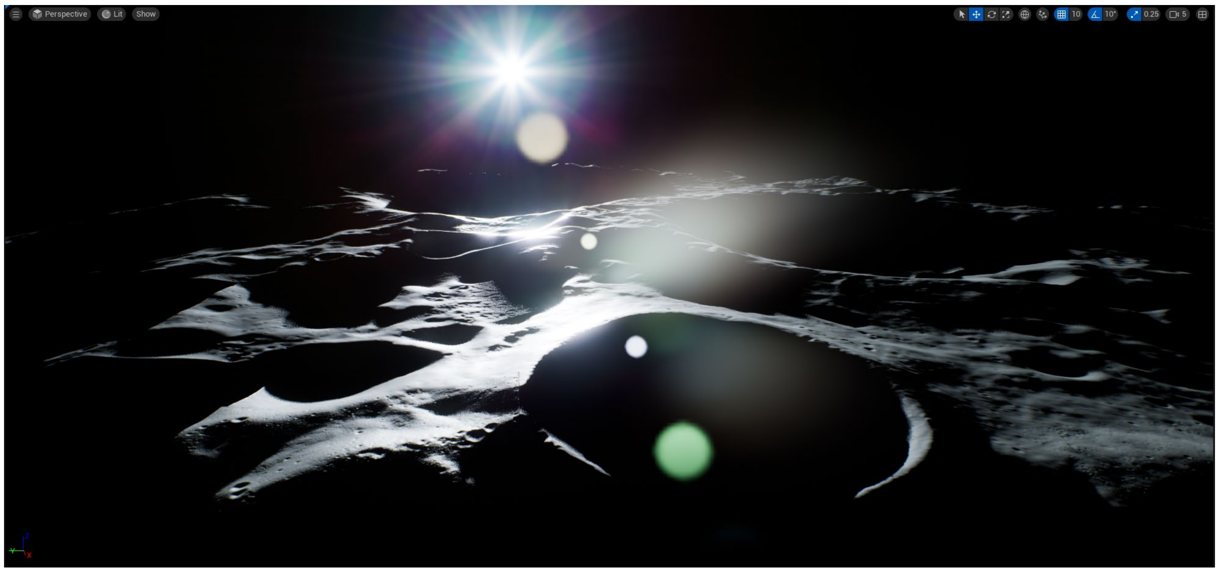


Fig. 1 UE5 Lunar-SPLIT: view of Shackleton Crater and Connecting Ridge.

A. Lunar Terrain Generation

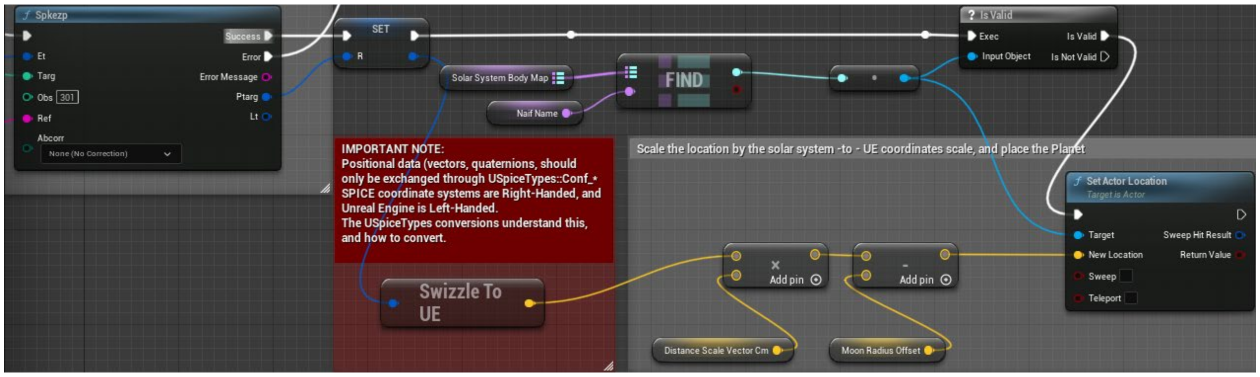
To develop accurate lunar terrain, two terrain generation methods were investigated: terrain generated from the native UE5 Landscape plugin and static meshes generated directly from digital elevation models (DEM) acquired by the Lunar Orbiter Laser Altimeter aboard the Lunar Reconnaissance Orbiter (LRO). Both methods generated terrain from the same DEM data. DEM data translation varied between each method where terrain generated by the UE5 Landscape Plugin produced terrain elevations with integer values and static meshes generated directly from DEM data produced terrain elevations with double precision values. The UE5 Lunar-SPLIT employed the static mesh method due to higher precision when translating the DEM data. In addition, static meshes allowed for realistic interaction with a directional light asset that simulates the Sun because the mesh can cast shadows via an assigned two-sided material. Accurate lunar terrain elevation directly affects the lighting environment with long shadows cast from terrain kilometers away and shadows in the immediate TLT assembly area. Lower resolution terrain static meshes farther away from the Connecting Ridge were used to generate long shadows that may interact with the TLT assembly area.

A higher resolution terrain static mesh at the Connecting Ridge was used to simulate smaller scale accurate shadows closer to the assembly area.

Lunar south pole terrain static meshes were obtained from the DUST team from their previous work with lunar terrain generation from laser altimetry data in the DEM format that they converted to wavefront object files [9]. Several static meshes of 5 meter/pixel resolution from DEM files were imported for the greater lunar south pole terrain. The Connecting Ridge static meshes were generated from 20 centimeter/pixel resolution DEM files and provided higher resolutions because TLT assembly operations were planned to occur at this site. The static meshes were imported into the UE5 with the NVG feature enabled for improved performance. The terrain meshes were positioned with a shared world origin at the lunar south pole in the simulation scene.

B. **Lunar Lighting Generation**

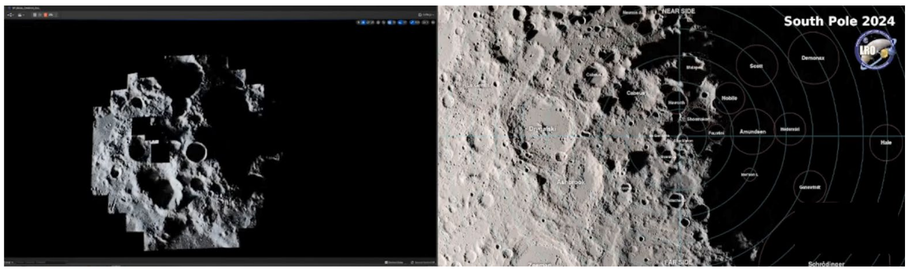
The Gamergenic MaxQ Spaceflight Toolkit UE5 plugin was used to position and orient UE5 celestial body assets such as the Sun and Earth according to the SPICE ephemeris data stored in kernels [10]. To pair ephemeris data for the desired celestial bodies with their respective game assets, UE5 blueprint functions were used to call SPICE commands to retrieve time, position, and orientation data from leap seconds kernels used in conversions between ephemeris time and coordinated universal time and planetary constants kernels that provide physical constants data for Solar System bodies [11]. The SPICE command "spkezr" was used to retrieve position and velocity data for the Sun and Earth from Spacecraft and Planet (S/P) kernels that contain spacecraft and celestial body position and velocity data [12]. The use of the "spkezr" command in a UE5 blueprint function is shown in Fig. 2 where the celestial bodies of the Sun and Earth are inputs and the position and velocity data are returned, transformed to the UE5 coordinate system and lunar south pole-centered frame, and assigned to their respective game assets.


IMPORTANT NOTE:
Positional data (vectors, quaternions, should only be exchanged through USpiceTypes::Conf_*
SPICE coordinate systems are Right-Handed, and Unreal Engine is Left-Handed.
The USpiceTypes conversions understand this, and how to convert.

Scale the location by the solar system -to - UE coordinates scale, and place the Planet

Fig. 2 Lunar south pole-centered blueprint function using MaxQ/SPICE calls for positioning of celestial body assets in UE5 Lunar-SPLIT simulation scene.

With accurate positions of the Sun and directional light assets linked to SPICE kernel data, the UE5 Lunar-SPLIT can show shadows and lighting on the lunar south pole at desired times of the year. LGIR was enabled after the directional light asset was parented to the Sun game asset. A comparison between the UE5 Lunar-SPLIT that uses "predict" versions of SPICE kernels for future mission planning and illumination at the lunar south pole from LRO data extrapolated to the year 2024 is shown in Fig. 3. Also, in Fig. 3 the shadows of the UE5 Lunar-SPLIT are shown in the same general position as the illumination from the LRO data extrapolated to the year 2024 at an arbitrary date that can be used to observe lighting conditions over a specified mission window. Overall visual differences include color grade, length of shadows, and darkness of shadows, which can be configured to match that of the LRO data but are reasonable for this preliminary use of UE5 Lunar-SPLIT as a testbed for the YOLOv5 object detection model.



a) b)

Fig. 3 Lunar shadow comparison: a) UE5 Lunar-SPLIT simulation scene and b) LRO illumination.

C. Tall Lunar Tower Engineering Development Unit Fidelity Model Integration

The TLT EDU being developed at NASA LaRC will demonstrate the supervised autonomous assembly of truss bays comprised of aluminum rivet joint corners and horizontal, vertical, and diagonal carbon fiber struts with aluminum end fittings. The Construction Robot System (CRS) is where the truss bays are positioned, assembled, and lifted incrementally for subsequent truss bay assembly. The Assistant Robot System (ARS) consists of a Universal Robot 10e (UR10e) manipulator that retrieves carbon fiber struts via end-effectors, positions strut members relative to the secured CRS strut members and rivets the strut members to the rivet joints. Two ARSs are used for strut assembly with each having an end-effector that has two D405 cameras for alignment of strut end fittings with the rivet joint corners.

Creo Parametric parts of the CRS and ARS were converted into wavefront object files and imported into Blender [13] for Filmbox (FBX) file format conversion. The FBX files were imported into the UE5 Lunar-SPLIT scene. Custom material assets for the CRS and ARS components that were in the field of view (FOV) of the D405 camera VCDT were then created. Visual fidelity matching of 3D printed parts that were fabricated for the LAE and game assets in the UE5 Lunar-SPLITs was achieved by adjusting base color, specular, and roughness parameters of material assets. Visual fidelity matching of the UE5 Lunar-SPLIT and LAE occurs in the rivet joint corner where the ARS is joining strut members to the rivet joint.

The TLT CRS and ARS game assets were placed in the UE5 Lunar-SPLIT scene near the Connecting Ridge where there would be no shadowing from the lunar terrain to maintain the same visual fidelity as the LAE. Due to the physical limitations of the LAE working area, shadowing of the physical TLT CRS and ARS were from changes in position of a light emitting diode (LED) light source that emulates the Sun and not from lunar terrain. The same shadow conditions on the TLT CRS and ARSs in the LAE were achieved in the UE5 Lunar-SPLIT. Matching shadow conditions allows synthetic image to match visually with images taken in the LAE. Visual matching enables accurate comparison of performance of both synthetic and real images used to train the YOLOv5 object detection model. Figure 4 depicts the TLT EDU CRS and ARS in the UE5 Lunar-SPLIT on a mock lander platform with a close-up view of the truss bay rivet joint corner.

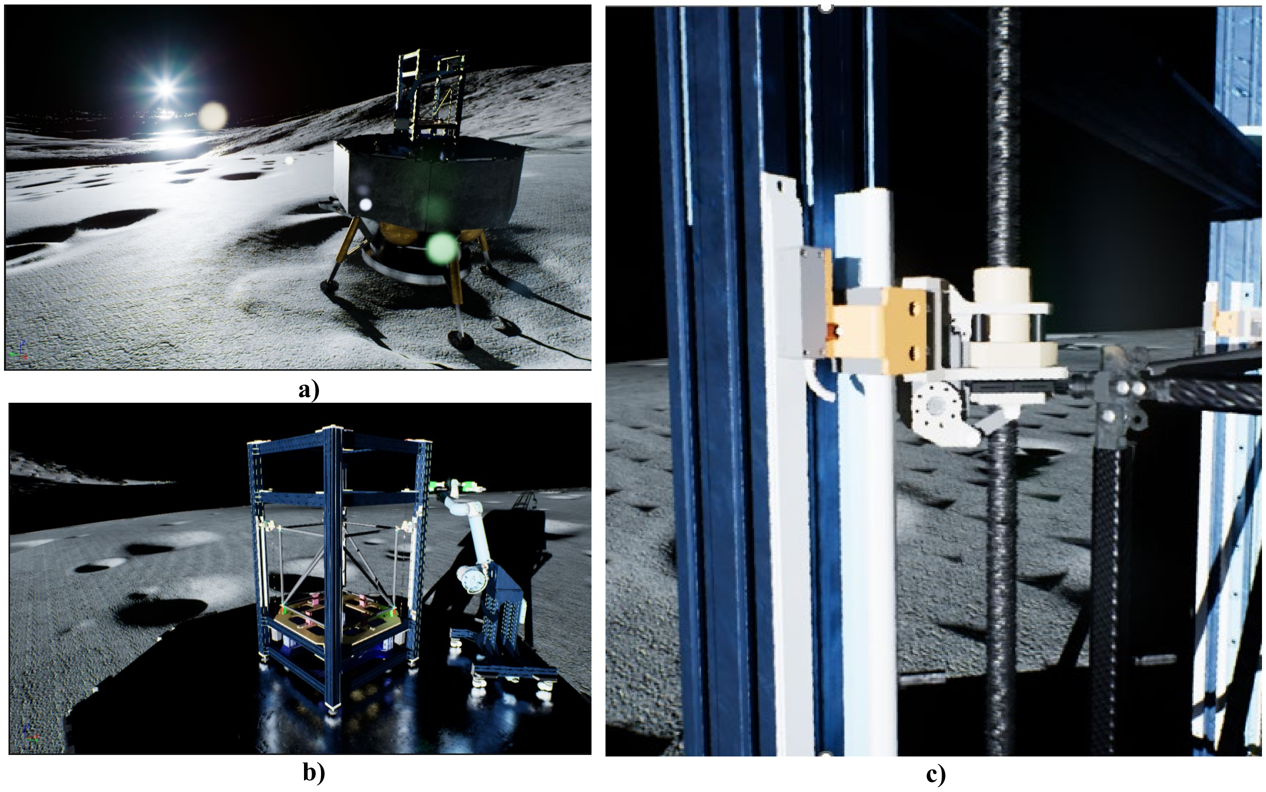


Fig. 4 TLT CRS and ARS Jig imported into UE5 Lunar-SPLIT: a) TLT EDU on top of mock lander, b) TLT CRS and ARS, and c) TLT Truss bay rivet joint corner.

D. **D405 Camera Virtual Camera Digital Twin**

Real images for the YOLOv5 object detection model were collected via a D405 camera as shown in Fig. 5. The D405 camera uses an image signal processor to enhance red, green, blue (RGB) data with a depth sensor, The D405 camera has an 87° horizontal FOV, a 58° vertical FOV and a 1280-pixel by 720-pixel RGB frame resolution. The camera data stream can be assessed using the associated software, the RealSense Viewer, and by importing a pyrealsense2 library in a Python environment. The D405 camera specifications along with the focal length, optical centers, and lens distortion coefficients values of the D405 camera used in the LAE were obtained from a camera calibration script using an OpenCV Python library [14]. These values were translated into the UE5 Lunar-SPLIT via a VirtualCamera2Actor blueprint game asset with a specified Lens File game asset to create a D405 camera VCDT.

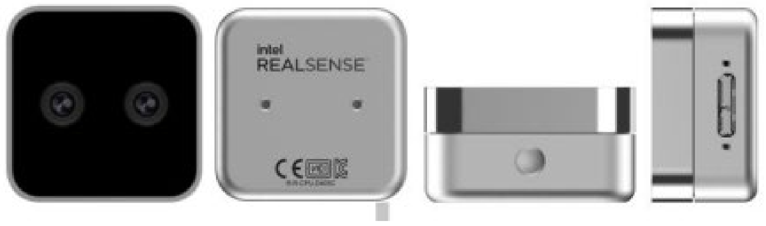


Fig. 5 D405 camera

1. OpenCV Camera Calibration

To import D405 camera specifications used for image collection in the LAE into a D405 camera VCDT, a camera calibration Python script produced a camera matrix that had the focal length, optic centers, and lens distortion coefficients of the physical camera. The camera calibration script imports an OpenCV library that outputs the values after processing an AVI video file that depicts a checkerboard positioned at various locations in the FOV of the D405 camera, Fig. 6. The AVI video file was created via a Python script that accesses the depth and RGB data streams from the D405 camera depth sensor via the pyrealsense2 library [15].

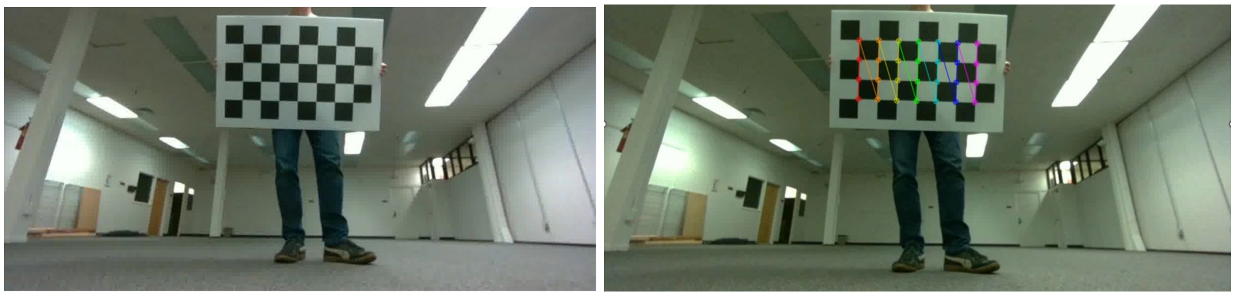
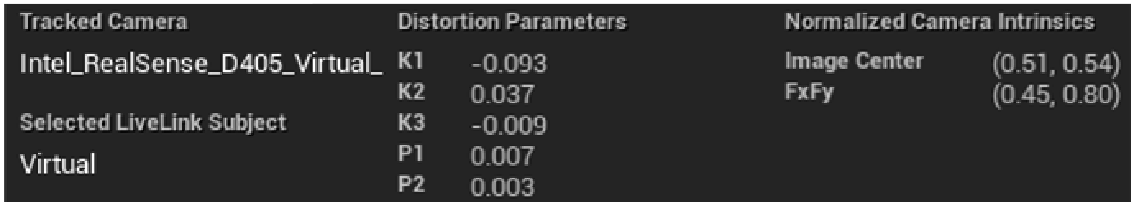

a) **b)**

Fig. 6 Checkerboard calibration method: a) AVI video file frame and b) camera calibration output from OpenCV python script.

2.. Unreal Engine 5 Lens File

The Camera Calibration plugin was introduced to Unreal Engine 4.27 and provides a pipeline to create accurate compositions from computer graphic renders and live video using a virtual camera in Unreal Engine that accurately simulates the physical camera used to capture real-world video footage for virtual production [16]. The camera calibration plugin and associated plugins VirtualCamera, Live Link, LiveLinkLens, OpenCV, and OpenCV Lens Distortion are enabled to utilize a UE5 Lens File that receives the input of the physical D405 camera distortion coefficient, camera matrix optic center and focal length values. Figure 7 shows the D405 Camera Lens file with distortion parameters set to the distortion coefficients, the camera image center set to the camera matrix optic center, and focal lengths set to focal length values obtained from the OpenCV camera calibration Python script.

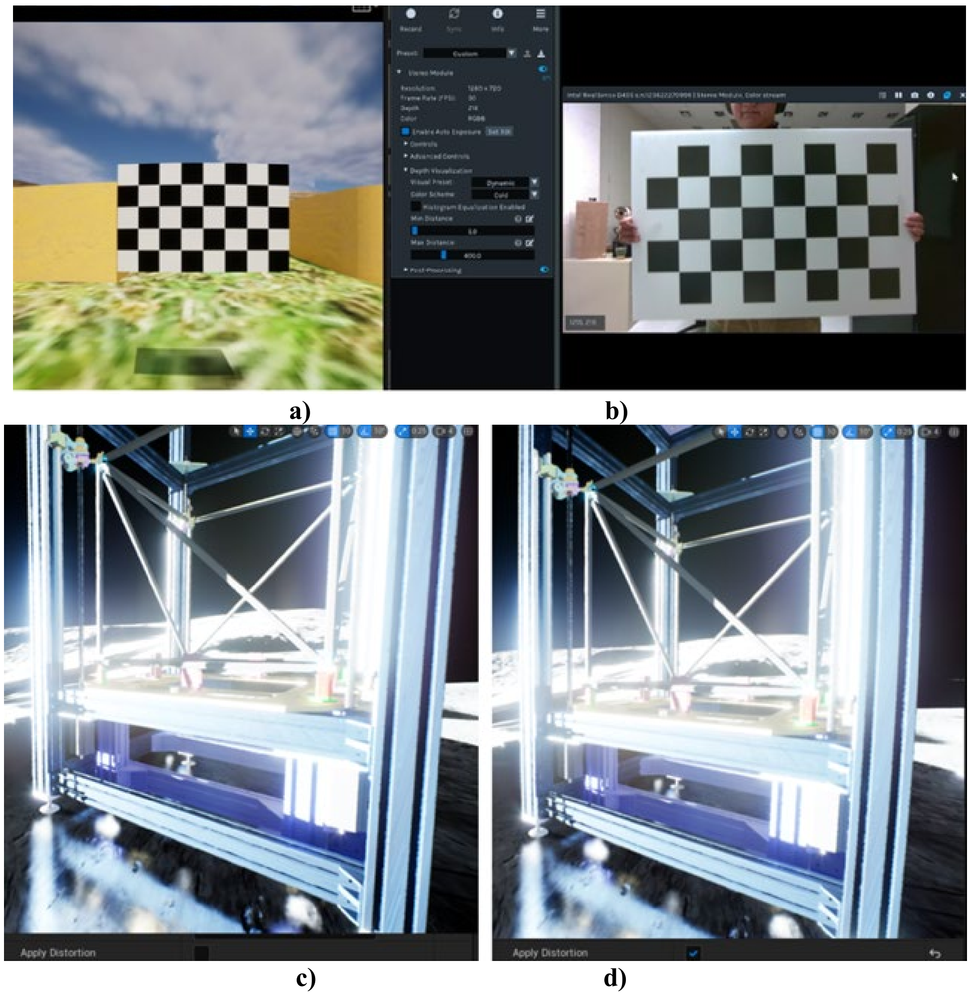

Tracked Camera
Intel_RealSense_D405_Virtual_
Selected LiveLink Subject
Virtual

Distortion Parameters	
K1	-0.093
K2	0.037
K3	-0.009
P1	0.007
P2	0.003

Normalized Camera Intrinsics	
Image Center	(0.51, 0.54)
FxFy	(0.45, 0.80)

Fig. 7 D405 camera Lens File with camera matrix and distortion coefficients from OpenCV camera calibration Python script

With the D405 Camera Lens File set to the outputs of the OpenCV camera calibration python script, a VirtualCamera2Actor game asset was created in UE5's Outliner with LiveLinkComponentController and LensDistortion components added to the game asset. With the D405 Camera Lens File selected, lens distortion is applied to the camera and the VirtualCamera2Actor game asset becomes the D405 camera VCDT . Figure 8 shows the view of the D405 camera VCDT in a test UE5 environment compared to the physical D405 camera RGB data stream in the RealSense Viewer with the checkerboard as reference. At the bottom of Fig. 8, the D405 camera VCDT is imported into the UE5 Lunar-SPLIT, and views of the TLT CRS with lens distortion disabled and enabled are shown. The lens distortion in Fig. 8 is slight. The D405 camera does not integrate fully with the Camera Calibration plugin for virtual production overlaying of the checkerboards in the real-world and UE5 Virtual environment. Thus, verification of similar visual fidelity was not supported. Visual matching of the D405 camera VCDT and the physical D405 camera feed was done by positioning the checkerboard at the same position in both environments. Since lens distortion is very slight in the D405 camera VCDT view as well as the physical D405 camera, the VCDT view is reasonable for this preliminary use of UE5 as a testbed for the YOLOv5 object detection model.



Preset: Custom
Stereo Module
Resolution: 1280 x 720
Frame Rate (FPS): 30
Depth: Z16
Color: RGB8
Enable Auto Exposure
Controls
Advanced Controls
Depth Visualization
Visual Preset: Dynamic
Color Scheme: Cold
Histogram Equalization Enabled
Min Distance: 0.0
Max Distance: 400.0
Post-Processing

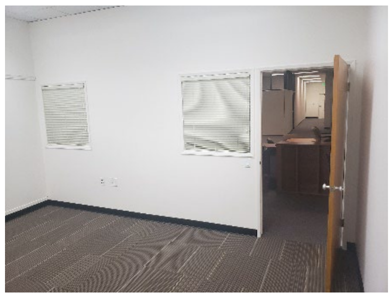
Fig. 8 Lens File camera lens distortion: a) test UE5 Environment with checkerboard and virtual camera with lens distortion from Lens File compared, b) physical D405 camera feed in the RealSense Viewer, c) UE5 Lunar-SPLIT lens distortion disabled, and d) UE5 Lunar-SPLIT lens distortion enabled.

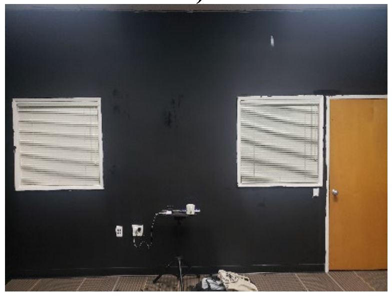
III.   Lunar Analog Environment

The LAE was constructed in the Research Annex building located outside of the campus of the University of Southern California. The LAE space had a working area of 4 meters by 5 meters with two windows and a single door which were sources of outside light leakage. The primary objective of the LAE was to have a physical TLT CRS and ARS assembly workspace that was in the FOV of the D405 camera with matching visual fidelity as the lighting conditions of UE5 Lunar-SPLIT virtual environment. With similar visual fidelities, images from the UE5 Lunar-SPLIT and the LAE were collected to train the YOLOv5 object detection model, and performances based on synthetic and real images were compared. Previous developments of indoor lunar analog environment facilities were investigated on their approach on dark room outfitting in areas of room construction, surface material, and lighting [3].

A. Lunar Analog Environment Dark Room Outfitting

Commercial-off-the-shelf materials were used to outfit the LAE. To remove stray light sources and reduce diffuse lighting emanating in the LAE, black vinyl fabric was attached to a wall to drape over the two windows and single door as well as on the ceiling to minimize light bounce from the light panels. Several coats of flat black paint were applied to the walls of the LAE to minimize light bounce. Emulation of the lunar terrain was accomplished with gray fabric with similar visual properties to lunar basalt rock. Figure 9 shows the dark room outfitting of the LAE.


a)

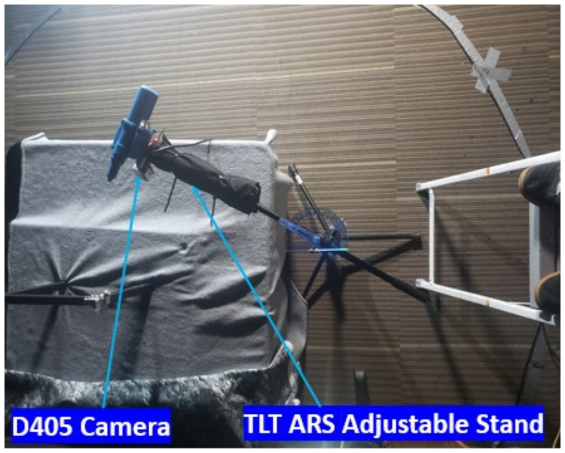

b)

c)

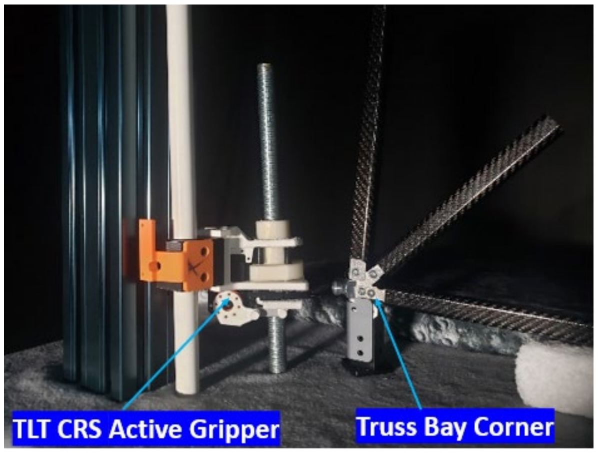
Fig. 9 LAE Environment darkroom outfitting: a) LAE prior to flat black paint application, b) flat black paint applied to walls of LAE, and c) black vinyl fabric attached to ceiling to inhibit diffuse lighting.

B. Tall Lunar Tower Test Jig

A low fidelity corner section of the TLT CRS that secures a rivet joint corner for strut assembly was 3D printed and a rivet joint corner which includes carbon fiber strut components with aluminum end fittings and a rivet joint was assembled as shown in Fig. 10. These TLT parts would be in the FOV of the D405 camera with the camera view focused on the TLT rivet joint assembly area. For ease of transitioning between a "Rivet In" and a "No Rivet" inspection case, a Phillips-head screw was used instead of a rivet when fastening the aluminum end fitting of a vertical strut to a rivet joint. The D405 camera was fastened to a camera mount attached to a 3D printed rivet gun tool that would be located at the end of the end-effector of the TLT ARS shown in Fig. 10. The D405 camera was attached to an adjustable arm to obtain varying viewing angles of the TLT rivet joint assembly area for real image dataset collection in the LAE.



a)

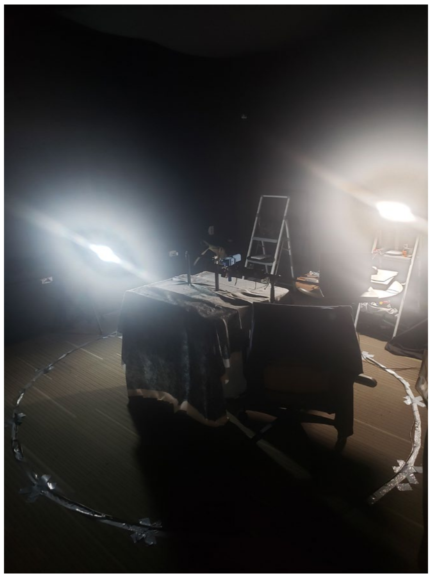


b)

Fig. 10 TLT test jig setup: a) 3D printed rivet gun tool with D405 camera mounted and attached to adjustable arm, and b) visual fidelity model of TLT CRS active gripper and truss bay rivet joint corner in the LAE.

C. LED Light Source and Lighting Circle Angles Setup

A single LED light source with a color temperature of 6000 K within the Sun's color temperature range was chosen to emulate the desired high-contrast lighting that was shown in the UE5 Lunar-SPLIT. The LED light source was attached to an adjustable arm and positioned 140 centimeters away from the TLT rivet joint and the center of the LED bulb was set to a height of 7 centimeters relative to the TLT CRS and ARS working area table to emulate the 3° above the horizon of the Sun angle at the lunar south pole. The LED light source position was maintained at these distances as the light source was rotated about the TLT rivet joint assembly area at 12 lighting circle angles (LCA) at 30° increments shown in Fig. 11. The D405 camera view was held constant with the camera facing the TLT rivet joint assembly area. Image dataset collection in the LAE and synthetic image dataset collection in the UE5 Lunar-SPLIT were taken at these 12 LCA. The LCA diagram shown in Fig. 11 shows a black and white shadow color wheel that indicates the positions of the LED light source that casts shadows on the TLT rivet joint assembly area and positions where the working area is fully illuminated with a gray gradient in positions directly left or right of the TLT rivet joint assembly area.


a)

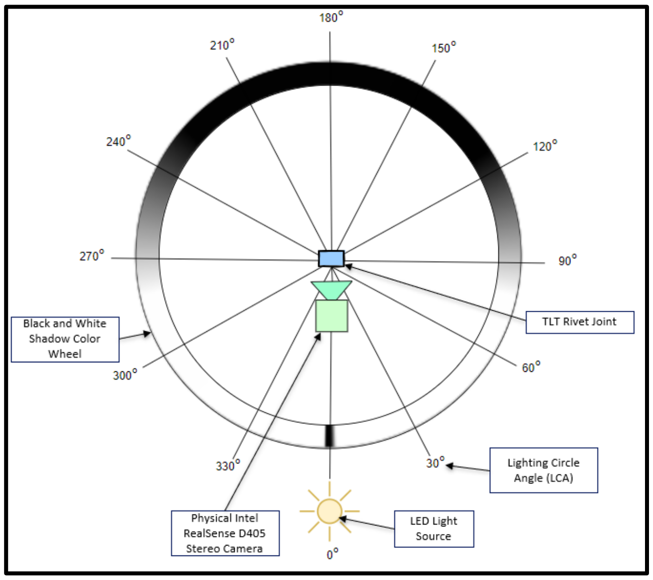
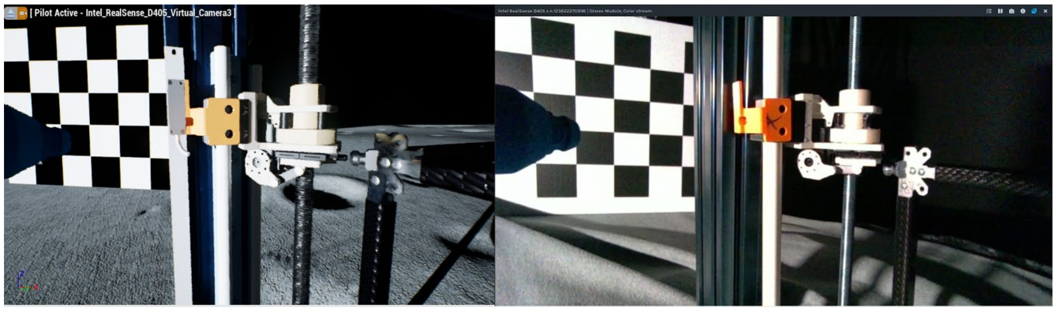

180°
210°
150°
240°
120°
270°
90°
Black and White Shadow Color Wheel
TLT Rivet Joint
300°
60°
330°
30°
Lighting Circle Angle (LCA)
Physical Intel RealSense D405 Stereo Camera
LED Light Source
0°
b)

Fig. 11 Lighting setup for placement of LED light source: a) LCA markers in LAE and b) LCA diagram with black and white shadow color wheel to depict when the TLT rivet joint assembly was fully illuminated or completely in shadow.

With the physical working area of the D405 camera established and the lunar lighting conditions of the UE Lunar-SPLIT adjusted to match the visual fidelity of the LAE, image datasets for YOLOv5 object detection model training could be generated. Figure 12 shows a side-by-side comparison of the UE5 Lunar-SPLIT and the LAE with the TLT rivet joint assembly area in the FOV of the D405 camera VCDT and physical D405 camera respectively.



a) b)

Fig. 12 Visual fidelity comparison of UE5 Lunar-SPLIT with the LAE with light source at 0° LCA: a) TLT rivet joint assembly area in UE5 Lunar-SPLIT and b) TLT rivet joint assembly area in LAE.

IV. You Only Look Once Object Detection Model Deployment in the LAE

A YOLOv5 object detection architecture was used to detect Rivet In and No Rivet inspection cases in the TLT rivet joint assembly area. The PyTorch machine learning library [17] was used to train separate YOLOv5 object detection models that were trained with synthetic images from the UE5 Lunar-SPLIT and real images collected in the LAE from the physical D405 camera. Two different YOLOv5 object detection models trained by synthetic image datasets were generated: the UE5 Trained YOLOv5 object detection model and the UE5 Combination Trained YOLOv5 object detection model. Figure 13 shows a flowchart of the three object detection models and the process in which each were trained and deployed in the LAE.

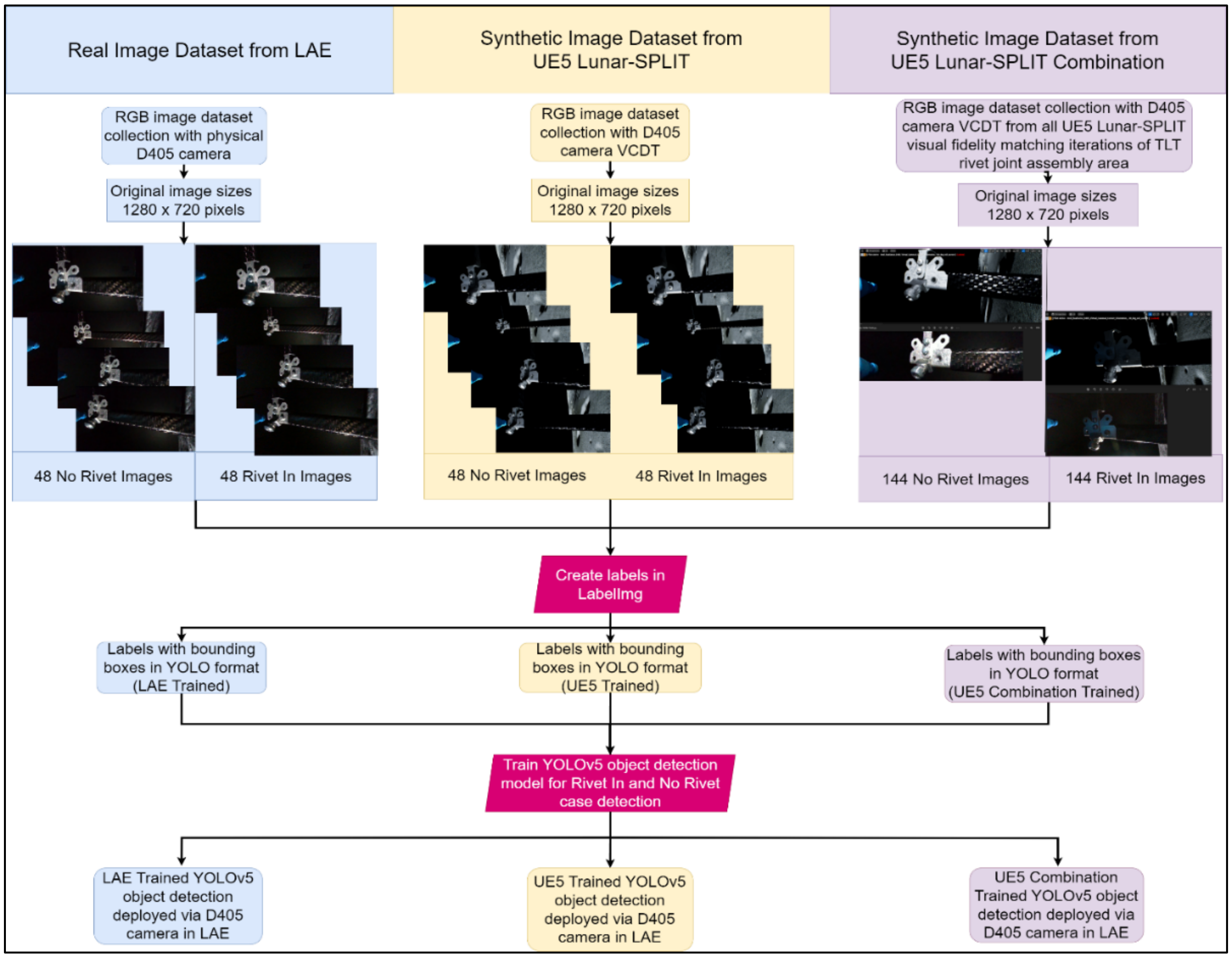

Real Image Dataset from LAE
Synthetic Image Dataset from UE5 Lunar-SPLIT
Synthetic Image Dataset from UE5 Lunar-SPLIT Combination
RGB image dataset collection with physical D405 camera
RGB image dataset collection with D405 camera VCDT
RGB image dataset collection with D405 camera VCDT from all UE5 Lunar-SPLIT visual fidelity matching iterations of TLT rivet joint assembly area
Original image sizes 1280 x 720 pixels
Original image sizes 1280 x 720 pixels
Original image sizes 1280 x 720 pixels
48 No Rivet Images
48 Rivet In Images
48 No Rivet Images
48 Rivet In Images
144 No Rivet Images
144 Rivet In Images
Create labels in LabelImg
Labels with bounding boxes in YOLO format (LAE Trained)
Labels with bounding boxes in YOLO format (UE5 Trained)
Labels with bounding boxes in YOLO format (UE5 Combination Trained)
Train YOLOv5 object detection model for Rivet In and No Rivet case detection
LAE Trained YOLOv5 object detection deployed via D405 camera in LAE
UE5 Trained YOLOv5 object detection deployed via D405 camera in LAE
UE5 Combination Trained YOLOv5 object detection deployed via D405 camera in LAE

Fig. 13 Flowchart detailing process for YOLOv5 object detection model training.

The UE5 Trained YOLOv5 object detection model was trained with 48 Rivet In and 48 No Rivet synthetic images which were the same number of images as the LAE Trained YOLOv5 object detection model. The UE5 Trained YOLOv5 object detection model was used as a one-to-one comparison to the LAE Trained YOLOv5 object detection model. The UE5 Combination Trained YOLOv5 object detection model was trained with 144 Rivet In and 144 No Rivet synthetic images generated during the visual fidelity matching of the LAE TLT rivet joint assembly area. The UE5 Combination Trained YOLOv5 object detection model leverages the ability to generate varied images for training quickly within the UE5 Lunar-SPLIT. Several iterations of the UE5 Lunar-SPLIT TLT rivet joint assembly area were created with changes in game asset materials, meshes and camera angles to better match the visual fidelity of the LAE. This finetuning in visual fidelity is discussed further in Section V of this paper.

Both synthetic and real images were labeled in LabelImg, an open-source tool for graphically labeling images [18]. Realtime deployment of these YOLOv5 object detection models used the physical D405 camera feed in the LAE and was implemented via a YOLOv5 Python script with the pyrealsense2 library.

A. Physical D405 Camera Image Dataset Acquisition and YOLOv5 Implementation

1. Real Image Dataset Collection Camera Angle View Setup in Lunar Analog Environment

Images were collected via a D405 image dataset collection Python script periodically during operation of the D405 camera in the LAE. Different camera views of the TLT rivet joint assembly area were achieved by rotating the adjustable D405 camera mount. Images were collected at four D405 camera angle positions: 0°, 30° pitch, 30° yaw, and 30° roll as shown in Fig. 14, for both the Rivet In and the No Rivet inspection cases. Images were collected with the LED light source positioned at each of the 12 LCA with a total of 96 real images acquired: 48 Rivet In and 48 No Rivet real images.

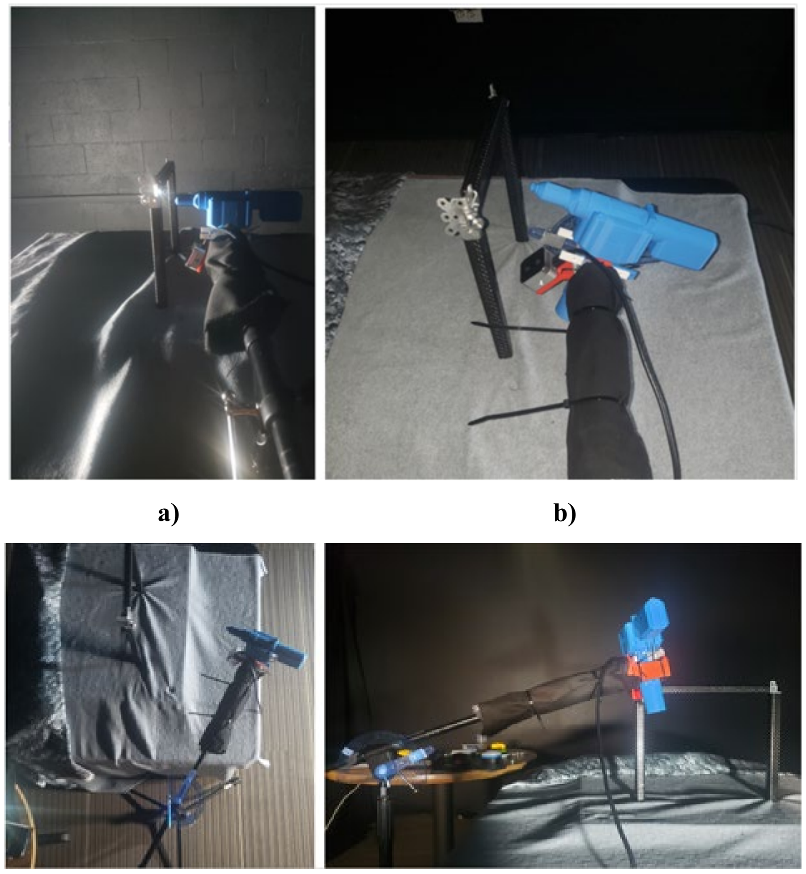


Fig. 14 TLT simplified end effector with D405 camera mounted: a) 0° D405 camera angle, b) 30° pitch D405 camera angle, c) 30° yaw D405 camera angle, and d) 30° roll D405 camera angle.

2. Real Image Dataset Labeling in LabelImg

The real image dataset was labeled in LabelImg with the appropriate class identification which was either "Rivet In" or "No Rivet" via a bounding box. LabelImg outputs the labels and bounding box corner coordinates for each image to train the YOLOv5 object detection model to detect and localize the rivet and determine the inspection case from the D405 camera feed in the LAE. Figure 15 shows the LabelImg graphical user interface classifying and creating bounding boxes for a Rivet In and No Rivet inspection case of the TLT rivet joint assembly area. After images have been labeled with bounding boxes, the labels are exported to YOLO text format and passed through a YOLOv5 training Python script.

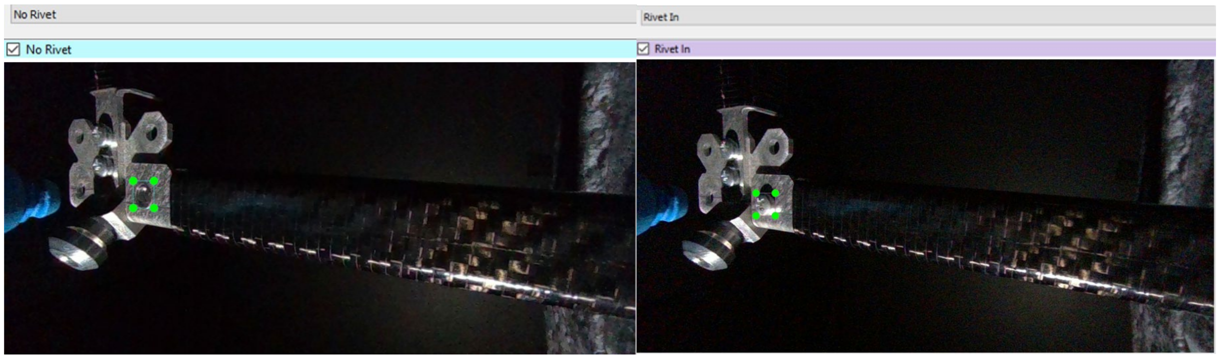


Fig. 15 Example of LabelImg graphical user interface for create bounding boxes and labeling classes for Rivet In and No Rivet inspection cases of real images collected in LAE located at 300° LCA and positioned at 0° D405 camera view.

3. YOLOv5 Training and Deployment with Rivet In and No Rivet Real Image Dataset

The YOLOv5 object detection architecture features real-time object detection by minimizing the number of neural network layers since each section of the input image is only received once [5]. The YOLOv5 object detection model was originally trained on a Common Objects in Context (COCO) 2017 dataset [19] that included 80 classes of common objects. The feature recognition of pre-trained Ultralytic fork of YOLOv5 [6] was leveraged through transfer learning to create a custom object detection model that detects the Rivet In and No Rivet cases of the TLT rivet joint assembly area. The images, labels, and bounding box corner coordinates from LabelImg were passed through the PyTorch machine learning library for training and outputted custom weights for the YOLOv5 object detection model. Figure 16 shows the labeling and bounding boxes created during training on a subset of the Rivet In and No Rivet images. Weights trained with Rivet In and No Rivet images were then called in the YOLOv5 Python inference script that runs the YOLOv5 object detection model over the physical D405 camera feed to detect Rivet In and No Rivet cases in the LAE. The inference is done on real-time RBG color frames of the physical D405 camera feed with a label, bounding box, and confidence score shown where a Rivet In or No Rivet case is detected (Fig. 16). The confidence score is between zero and one and is the probability that an object is present [5].

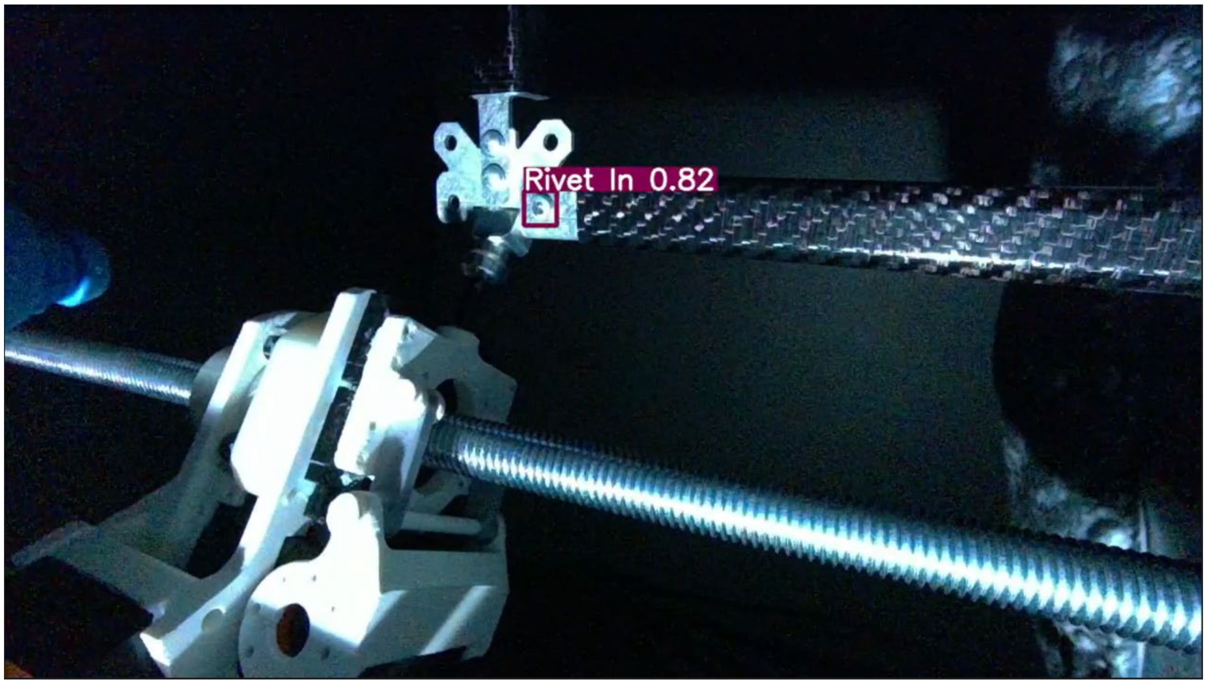
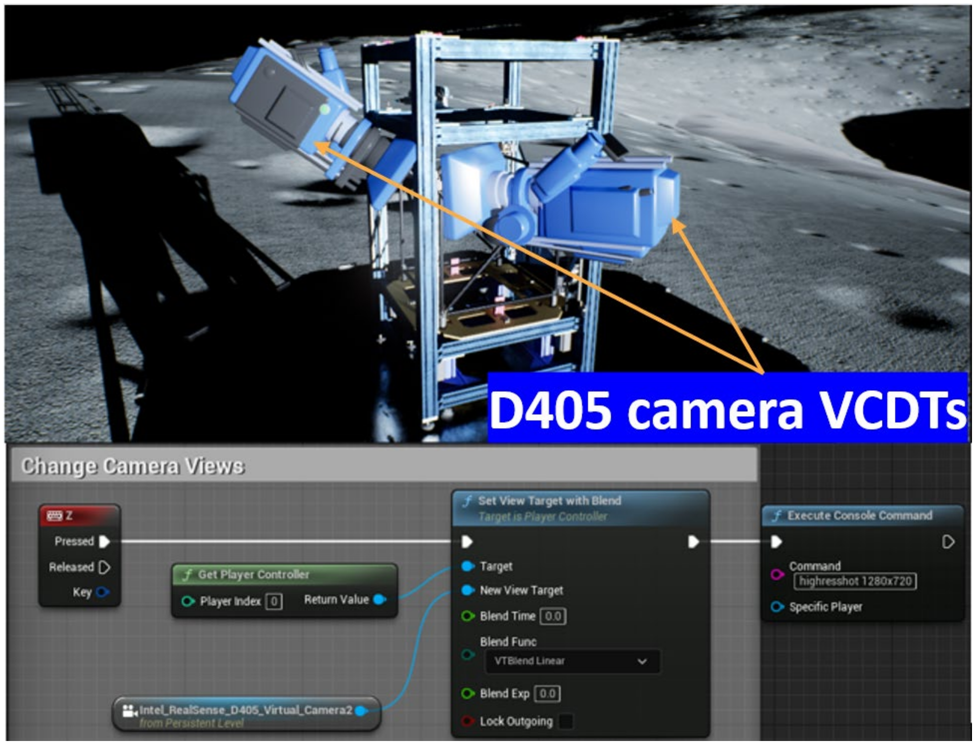


Fig. 16 YOLOv5 Rivet In object detection case using physical D405 Camera

B. UE Lunar-SPLIT D405 Camera Virtual Camera Digital Twin Image Dataset Acquisition and YOLOv5 Implementation

The same labeling and bounding box annotation process in LabelImg for the real image dataset collected in the LAE is applied to the synthetic image dataset collected in the UE Lunar-SPLIT virtual environment. The synthetic images were collected via a level blueprint game asset that screenshots the runtime UE5 scene periodically at varying camera angles by switching between multiple D405 camera VCDT game assets that are positioned at the four D405 camera angle views and positioning the directional light game asset that that emulates the Sun according to the 12 LCA (Fig. 17).



Player Index 0
Blend Time 0.0
Blend Func VTBlend Linear
Blend Exp 0.0
Lock Outgoing
Command highresshot 1280x720

Fig. 17 Synthetic image dataset collection blueprint with depiction of multiple game assets for switching between D405 camera VCDT camera angles

These images were then sorted into their respective Rivet In and No Rivet states and then annotated in LabelImg for labeling and bounding box creation (Fig. 18). The same process of passing the images, labels, and bounding box corner coordinates from LabelImg to the PyTorch machine learning library for training and outputted custom weights was followed to generate both the UE5 Trained YOLOv5 object detection model and the UE5 Combination Trained YOLOv5 object detection model. The UE5 Trained YOLOv5 and the UE5 Combination Trained YOLOv5 object detection models were deployed in the LAE via the physical D405 camera feed to detect Rivet In and No Rivet cases.

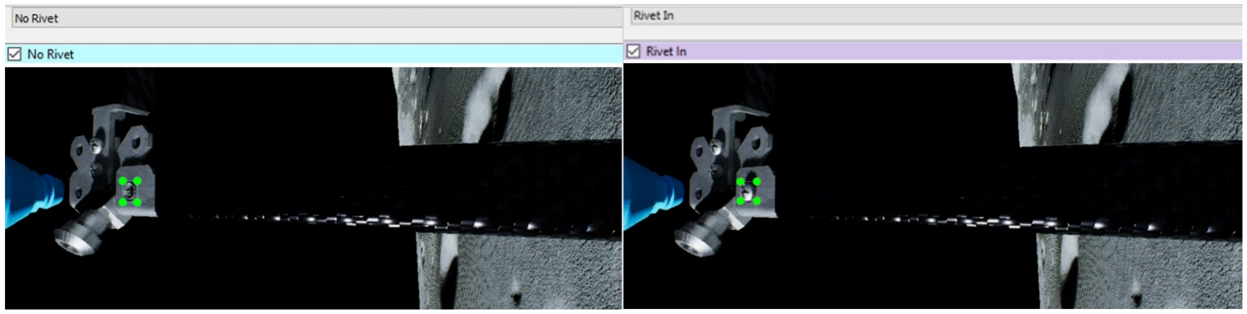


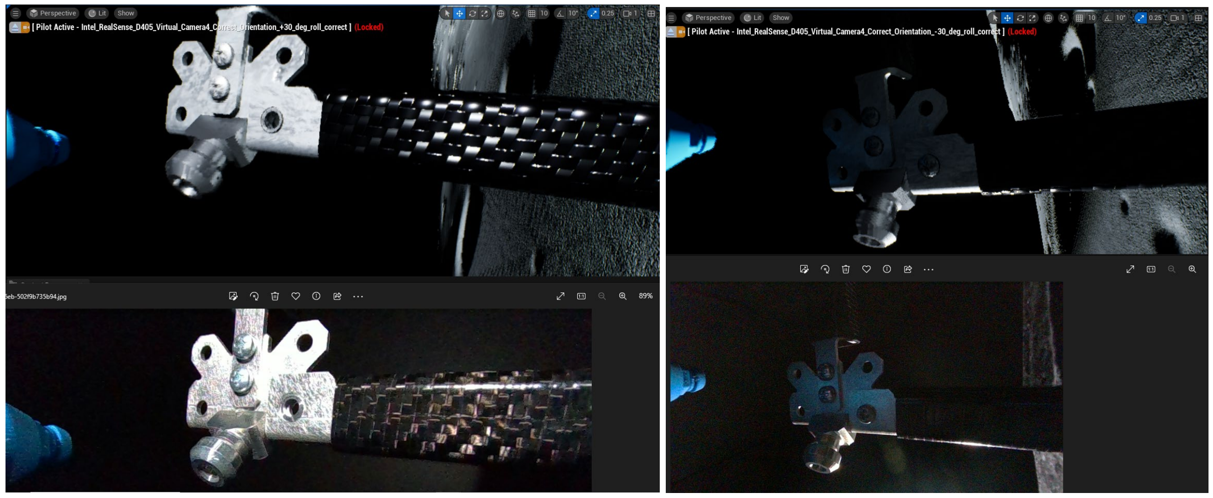
Fig. 18 Example of LabelImg graphical user interface for create bounding boxes and labeling classes for Rivet In and No Rivet inspection cases of real images collected in UE5 Lunar-SPLIT located at 300° LCA and positioned at 0° D405 camera view.

V. Synthetic Image Dataset Finetuning for Computer Vision Model Training

As mentioned previously in Section IV, the UE5 Combination Trained YOLOv5 object detection model was trained with 144 Rivet In and 144 No Rivet synthetic images. These additional synthetic images were generated during the iterative process of visual fidelity matching of the UE5 Lunar-SPLIT and the LAE. The first pass of visual fidelity matching resulted in the aluminum material assigned to the rivet joint and the vertical strut end fitting game assets having less reflectivity than the LAE counterparts. The second pass changed the camera angles to match the view of the D405 camera in the LAE. With the change in camera angle, the differences in shadowing in the UE5 Lunar-SPLIT virtual environment and the physical LAE environment became apparent. When the LED light source in the LAE was positioned at LCA that were behind the TLT rivet joint assembly area, the images collected showed visibility of the Rivet In and No Rivet cases. When the directional light game asset in the UE5 Lunar-SPLIT was positioned at the same LCA, the TLT rivet joint assembly area was in complete shadow and had no visibility of the Rivet In and No Rivet cases. The UE5 Lunar-SPLIT depicts a more accurate lighting condition that would be on the lunar surface because of the physical limitations of the darkroom outfitting in the LAE. For example, the LAE is subjected to light bouncing off the walls even with the coats of flat black paint.

The final visual fidelity matching of UE5 Lunar-SPLIT to the LAE incorporated a point light game asset to provide artificial lighting in the virtual lunar environment. This allowed for visibility of the Rivet In and No Rivet cases when originally the TLT rivet joint assembly area was in shadow. When training the YOLOv5 object detection models with synthetic images, confidence scores during validation of a subset of the real image dataset significantly improved because shadowed Rivet In and No Rivet cases would skew feature recognition. The No Rivet case visual fidelity matching called for the addition of an imperfection around the fastener hole of the vertical end fitting. Previous visual fidelity matching disregarded the imperfection which resulted in lower confidence scores during validation. Figure 19 shows the visual differences between the UE5 Lunar-SPLIT and LAE TLT rivet joint assembly area and the additional game asset modifications for closer visual fidelity.

With three visual fidelity matching passes, the UE5 Combination Trained YOLOv5 object detection model was trained with synthetic images with varied lighting conditions, rivet joint and vertical strut end fitting materials, material imperfections, and camera angle positions. The synthetic image dataset collection for this object detection model showcases how quickly varied synthetic image data can be generated and used for training an object detection model.


Perspective
Lit
Show
[Pilot Active - Intel_RealSense_D405_Virtual_Camera4_Correct_Orientation_+30_deg_roll_correct] (Locked)
[Pilot Active - Intel_RealSense_D405_Virtual_Camera4_Correct_Orientation_-30_deg_roll_correct] (Locked)
89%

a) **b)**

Fig. 19 Illustrates the iterative process of matching the visual fidelity of the UE5 Lunar-SPLIT to the LAE: a) UE5 Lunar-SPLIT (top) versus LAE (bottom) No Rivet Case 60 ° LCA at 30 ° roll D405 camera angle with addition of vertical node end fitting imperfection and b) UE5 Lunar-SPLIT (top) versus LAE (bottom) Rivet In Case 240 ° LCA at 30 ° roll D405 camera angle with addition of point light to avoid complete shadowing of rivet.

VI. You Only Look Once Object Detection: Real Image Versus Synthetic Image Dataset Comparison and Discussion

A. Confidence Score Data Collection and Processing for Comparison Plot

Confidence score data collection was conducted in the LAE to compare the three YOLOv5 object detection models. Realtime inference of the object detection models were done at the four D405 camera angles with the LED light source positioned at each of the 12 LCA for the Rivet In and No Rivet inspection cases. Confidence score data were output to a common separated values (CSV) file after running the object detection model for ten seconds. The confidence score data CSV file were further processed by taking the mean of the confidence scores if a Rivet In and No Rivet case was detected, outputting a zero confidence score if there was no detection, and outputting a negative confidence score for a false negative or false positive. For the Rivet In cases, a true positive is correct detection of a rivet when the rivet is present, and a false negative is an incorrect detection of a No Rivet case when the rivet is present. For the No Rivet cases, a true negative is correct detection of when the rivet is not present, and a false positive is incorrect detection of a Rivet In case detection when the rivet is not present.

B. Rivet In YOLOv5 Object Detection Comparison Plot Discussion

Figure 20 shows a comparison plot of the three object detection models for the Rivet In object detection case. The bottom of the plot shows the previous Black and White Shadow Color Wheel shown in Fig. 11 in the form of a gradient to give context of the shadowing and illumination of the TLT rivet joint assembly area. Best fit curves for each of the object detection models show the trend of the confidence scores. A confidence bound for each best fit curve encloses 80 percent of the dataset to show the overall confidence score trends.

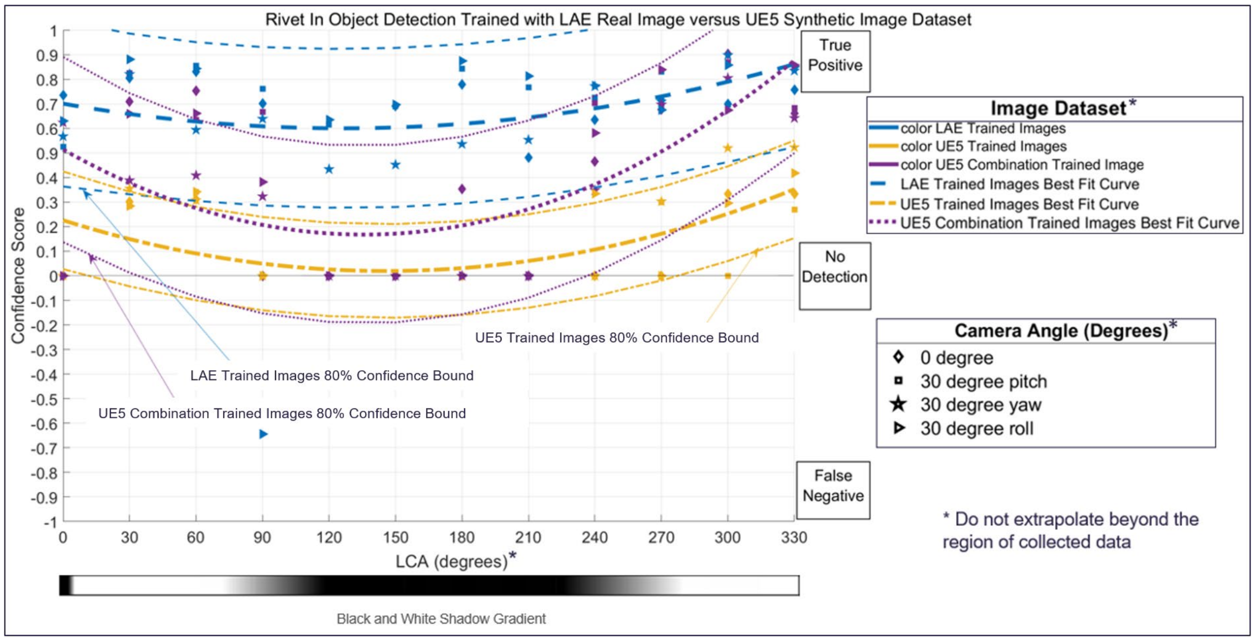


Fig. 20 Rivet In YOLOv5 Object Detection Comparison Plot

The LAE Trained Images YOLOv5 object detection model shows the highest confidence scores averaging around 0.80 when compared to the UE5 Trained and the UE5 Combination Trained Synthetic Images YOLOv5 object detection models. This is to be expected due to the real image dataset training in the same location as where the object detection inferencing is taking place. The LAE Trained Images YOLOv5 object detection has one case of a false negative or No Rivet detection.

The UE5 Trained Images YOLOv5 object detection model shows the lowest confidence scores averaging around 0.30 and has the most no detection cases even with full illumination of the TLT rivet joint assembly working area. Since this object detection model was trained with the same number of images as that trained the LAE Trained Images YOLOv5 object detection model, the visual fidelity of each of the synthetic images would need to be almost identical to the images collected in the LAE to produce higher confidence scores.

The UE5 Combination Trained object detection model has higher confidence scores than the UE5 Trained Images YOLOv5 object detection model. When the TLT rivet joint assembly working area is fully illuminated, confidence scores can reach above 0.80 and follow closely to the LAE Trained Images YOLOv5 object detection model.

When the TLT rivet joint assembly area is fully shadowed in the 120° to 240° LCA range, the synthetic images trained object detection models have no detection and the LAE Trained Images YOLOv5 object detection model confidence scores trend downwards.

C. No Rivet YOLOv5 Object Detection Comparison Plot Discussion

Figure 21 shows a comparison plot of the three object detection models for the No Rivet object detection case. Confidence scores varied greatly throughout the positioning of the LED light source at different LCA.

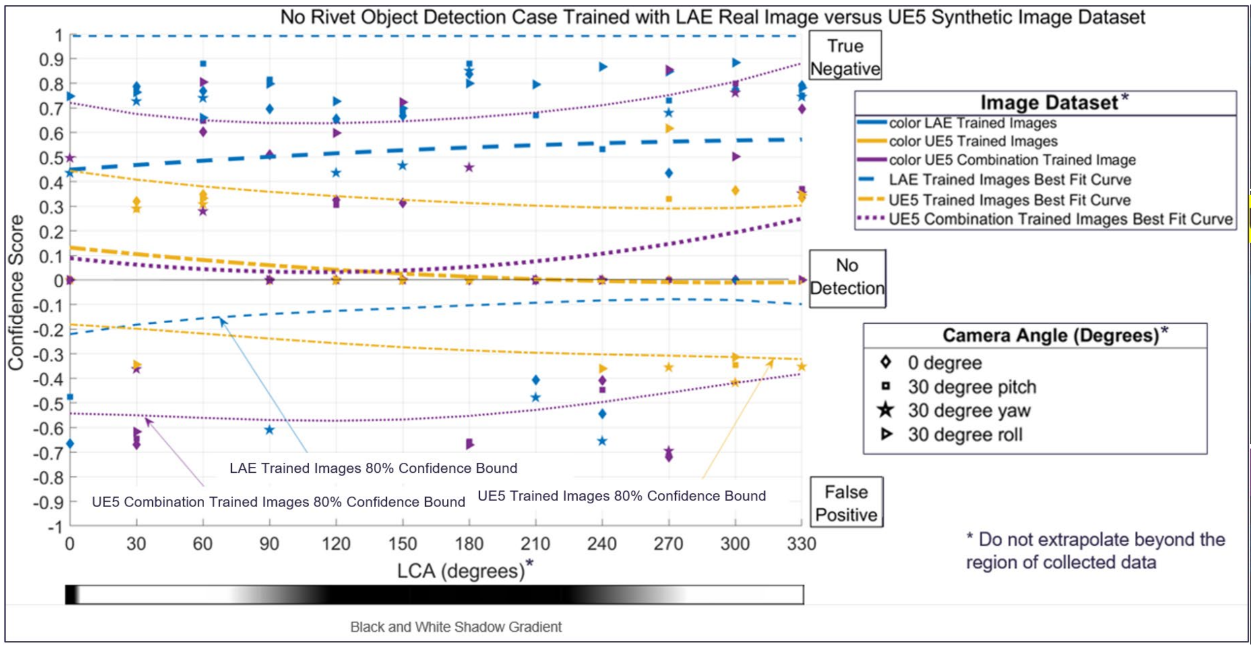


Fig. 21 No Rivet YOLOv5 Object Detection Comparison Plot

All three object detection models have several cases where there are false positives or Rivet In detections which can be attributed to similar visual features of the fastener hole of when the rivet is not present to a rivet. For example, a No Rivet case can still have a metal curvature from a hole where the aluminum material shines like a rivet.

The two synthetic image trained object detection models continue the no detection trend when the TLT rivet joint assembly area is fully shadowed in the 120° to 240° LCA range. The UE5 Trained Images YOLOv5 object detection model has a general low confidence score trend. The UE5 Combination Trained Images YOLOv5 object detection model has a similar data range as the LAE Trained Images YOLOv5 object detection model with similar true negative and false positive detections but with lower confidence scores.

VII. Concluding Remarks and Future Work

A. Concluding Remarks from Real Image Dataset and Synthetic Image Dataset Training of YOLOv5 Object Detection

In this work, an Unreal Engine 5 (UE5) video game engine Lunar South Pole Lighting Testbed (Lunar-SPLIT) to simulate realistic lunar lighting conditions was developed and used to simulate lighting on riveted joints of a TLT. The fidelity of the simulation environment was investigated by comparing the accuracy of computer vision models trained using synthetic image data and trained from real image data collected in a lunar analog environment (LAE). The UE5 Lunar-SPLIT terrain, lighting conditions, and camera were developed from imported static meshes generated from digital elevation maps (DEM), a directional light game asset positioned by celestial body ephemeris data, and a UE5 Lens File set to physical D405 camera specifications. The LAE was developed to emulate the lighting conditions of the UE5 Lunar-SPLIT and to provide a TLT test jig for YOLOv5 object detection model synthetic and real image data training comparisons. Three YOLOv5 object detection models trained from synthetic and real image datasets were compared that included a model with multiple visual fidelity matching passes that varied in lighting conditions, rivet joint and vertical strut end fitting materials, material imperfections, and camera angle positions that produced higher overall confidence scores.

The results shown in the comparison plots show the importance of the use of artificial lighting in completely shadowed areas where object detection models are used to inspect completion of assembly. Object detection models trained with synthetic image datasets result in no detection when images of the objects to be detected have muted features from shadowing. Many synthetic images can be generated quickly by leveraging blueprint scripting in the UE Lunar-SPLIT and can produce high confidence scores when the TLT rivet joint assembly area is fully illuminated in LAE. Object detection cases should have distinguishing features from one another and should have minimal similar features particularly with detecting metal parts in high contrast lighting areas due to reflectivity.

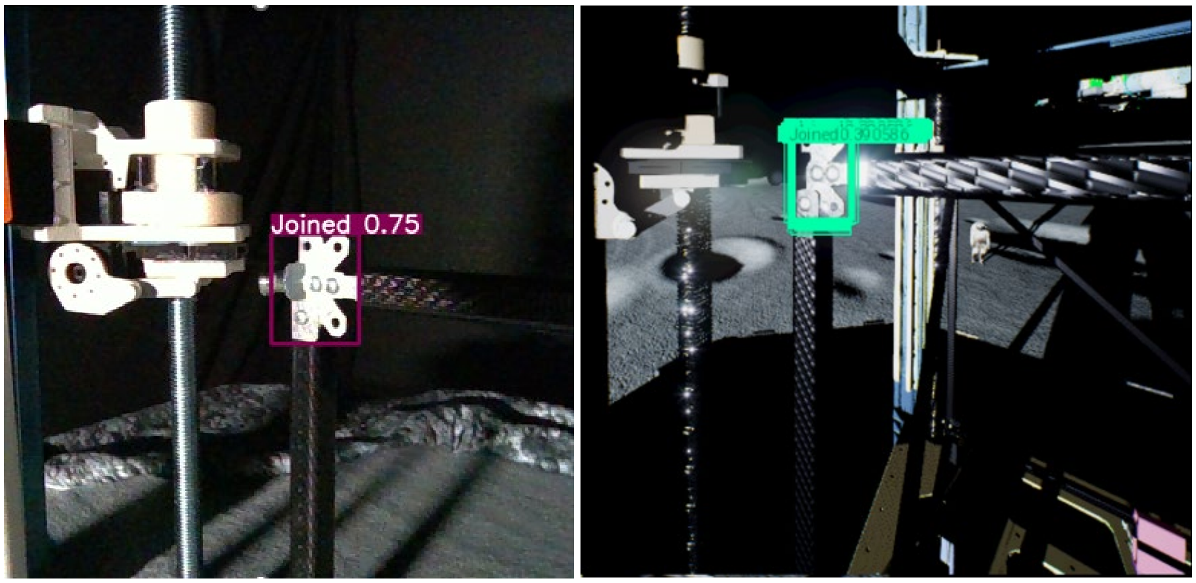
B. Future Work with UE5 Lunar-SPLIT

1. Synthetic Image Dataset Generation Investigations

An investigation of synthetic images generated with artificial lighting of the TLT rivet joint assembly area in the UE5 Lunar-SPLIT could result in consistent and higher confidence scores with correct detections that show the same trend as object detection models trained with real image dataset with similar artificial lighting conditions. An investigation of the opposition effect when the optical axis of a camera and direction of illumination coincide, and apparent contrast disappears would be instructive. Further, an investigation would be useful on the use of data augmentation in the PyTorch machine learning library to increase the number of varied synthetic images by creating new synthetic images by rotating, flipping, and zooming existing synthetic images. Data augmentation could result in higher confidence scores.

2. Integration of Neural Network Models in UE5 Lunar-SPLIT

Neural network model inferencing can be done at runtime in UE5 when models are converted to an Open Neural Network Exchange (ONNX) model and are imported into the scene using a UE5 Neural Network Inference (NNI) plugin [20]. Historia-Inc developed an example project which uses a NNI plugin that can run a YOLOv7 object detection model in a UE5 scene at runtime [21]. Figure 22 shows this example project integrated with the UE5 Lunar-SPLIT with a custom YOLOv5 object detection model trained in PyTorch to detect “Joined” or “Detached” vertical strut assemblies with rivet joints. Integration of different neural network models such as pose detection in UE5 Lunar-SPLIT could be investigated.



a) b)

Fig. 22 Confidence score comparison YOLOv5 object detection model inference in LAE versus inference in UE5 Lunar-SPLIT: a) example of Joined detection in LAE in real-time and b) example of Joined detection in UE5 Lunar-SPLIT at runtime.

Acknowledgments

We would like to acknowledge the NASA Johnson Space Center Digital Lunar Exploration Sites Unreal Simulation Tool (DUST) Team that includes Lee Bingham, Jack Kincaid, and Benjamin Weno for their contributions to this investigation by providing guidance on developing a lunar virtual environment in UE5. We would like to also acknowledge Dr. Dan Erwin at the University of Southern California for providing a space for the LAE where we were free to paint, nail, and duct tape. Finally, we acknowledge Jessica Friz from the NASA Langley Research Center Simulation Development and Analysis Branch for invaluable guidance on modeling and simulation efforts for this project and David Mercer from the NASA Langley Research Center Structural Mechanics and Concepts Branch for providing access to all the software used to complete this project.

References

[1] Doggett, W. R., Heppler, J., Mahlin, M. K., Pappa, R. S., Teter, J., Song, K., White, B., Wong, I.., and Mikulas, M., “Towers: Critical Initial Infrastructure for the Moon,” AIAA SciTech 2023 Forum, 2023.

[2] Allan, M., Wong, U., Furlong, M. P., Rogg, A., McMichael, S., Welsh, T., Chen, I., Peters, S., Gerkey, B., Quigley, M., Shirley, M., Deans, M., Cannon, H., and Fong, T., “Planetary Rover Simulation for Lunar Exploration Missions,” IEEE Aerospace Conference, 2019.

[3] Ludivig, P., Calzada-Diaz, A., Mendez, M., Voos, H., Lamamy, J., “Building a Piece of the Moon: Construction of Two Indoor Lunar Analogue Environments,” 71st International Astronautical Congress (IAC) -The CyberSpace Edition, 12-14 October 2020.

[4] "Unreal Engine 5 Documentation," Epic Games, Inc. 2023. [Online]. Available: https://docs.unrealengine.com/5.0/en-US/. [Accessed January 1, 2023].

[5] Redmon, J., Divvala, S., Girshick, R., Farhadi, A., “You Only Look Once: Unified, Real-Time Object Detection,” Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 779-788.

[6] “YOLOv5,” Jocher, 2020. [Online]. Available: https://github.com/ultralytics/yolov5. [Accessed May 12, 2023].

[7] "Nanite Virtualized Geometry," Epic Games, Inc. 2023. [Online]. Available: https://docs.unrealengine.com/5.0/en-US/nanite-virtualized-geometry-in-unreal-engine/. [Accessed February 23, 2023].

[8] "Lumen Global Illumination and Reflections," Epic Games, Inc. 2023. [Online]. Available: https://docs.unrealengine.com/5.0/en-US/lumen-global-illumination-and-reflections-in-unreal-engine/. [Accessed February 23, 2023].

[9] Bingham, L., Kincaid, J., Weno, B., Davis, N., Paddock, E., Foreman, C., “Digital Lunar Exploration Sites Unreal Simulation Tool (DUST),” 2023 IEEE Aerospace Conference, Big Sky, MT.

[10] "MaxQ: Spaceflight Toolkit For Unreal Engine 5," Gamergenic, 25 12 2021. [Online]. Available: https://www.gamergenic.com/project/maxq/. [Accessed February 16, 2023].

[11] “SPICE Kernel Required Reading,” NAIF, JPL, 2021. [Online]. Available: https://naif.jpl.nasa.gov/pub/naif/toolkit_docs/C/req/kernel.html#Kernel%20Types [Accessed February 16, 2023].

[12] “SPKEZR,” JPL, 2021. [Online]. Available: https://naif.jpl.nasa.gov/pub/naif/toolkit_docs/FORTRAN/spicelib/spkezr.html [Accessed February 16, 2023].

[13] Blender Development Team. (2021). Blender (Version 2.93.4). [Online]. Available: https://www.blender.org

[14] “calib3d,” OpenCV, 2023. [Online]. Available: https://github.com/opencv/opencv/tree/4.x/modules/calib3d. [Accessed March 2, 2023].

[15] “Intel RealSense SDK 2.0,” Intel RealSense. 2023. [Online]. Available: https://github.com/IntelRealSense/librealsense. [Accessed March 2, 2023].

[16] “Camera Lens Calibration Quick Start Guide,” Epic Games, Inc. 2023. [Online]. Available: https://docs.unrealengine.com/5.0/en-US/camera-lens-calibration-quick-start-for-unreal-engine/ [Accessed May 4, 2023].

[17] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S., “PyTorch: An Imperative Style, High-Performance Deep Learning Library,", 2019. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[18] “LabelImg,” Label Studio Community. 2018. [Online]. Available: https://github.com/HumanSignal/labelImg. [Accessed June 8, 2023].

[19] Lin, T., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, L., and Dollár, P., “Microsoft COCO: Common Objects in Context,”, 2015. https://doi.org/10.48550/arXiv.1405.0312. [Online]. Available: https://arxiv.org/abs/1405.0312.
[20] "OnnxRuntime-Unreal Engine," Microsoft Corporation, 2022. [Online]. Available: https://github.com/microsoft/OnnxRuntime-UnrealEngine. [Accessed January 27, 2023].
[21] “NNI Sample,” Historia-Inc, 2022. [Online]. Available: https://github-com.translate.goog/historia-Inc/NNI_Sample?_x_tr_sl=auto&_x_tr_tl=en&_x_tr_hl=en&_x_tr_pto=wapp. [Accessed July 6, 2023].