

## Vi3W

### Task-1

#### Documentation: Steps, Challenges, and Optimizations in Setting up Stable Fast 3D

##### i) Setting Up the Environment

###### 1. Create a Virtual Environment

- Created an isolated environment using: **python -m venv sf3d\_env**.
- Activated the environment:
  - **Windows: sf3d\_env\\Scripts\\activate**

###### 2. Update PATH

- Checked the Python path with **where python** to make sure the interpreter in the right order, which is **sf3d\_env**.
  - Successfully updated the value of PATH to point to the virtual environment's directory, **Scripts**.
- 

##### ii) Installing Dependencies

###### 1. Requirements Installation

- Ran **pip install -r requirements.txt** for installation of dependencies.
- Modified the **requirements.txt** to exclude GPU dependencies:
  - Removed **rembg[gpu]** and instead added: **rembg==2.0.57; sys\_platform == 'darwin'**.

###### 2. Install PyTorch Compatible with CPU

- PyTorch, TorchVision, and Torchaudio were installed using the following command:
    - **pip install torch==2.5.1+cpu torchvision==0.20.1+cpu torchaudio==2.5.1+cpu --index-url https://download.pytorch.org/whl/cpu**
-

### iii) Troubleshooting uv\_unwrapper

#### 1. Error Encountered

- The script reported:
  - ImportError: cannot import name 'Unwrapper' from 'uv\_unwrapper'.

#### 2. Steps to Resolve

- Verified that setup.py exists in the directory uv\_unwrapper.
  - Installed uv\_unwrapper by:
  - pip install ./uv\_unwrapper/
- If torch was missing after installation, reinstalled the torch using the CPU version.

#### 3. Alternative Resolution

- Added the following to the Python path: **uv\_unwrapper** directory:
    - **import sys**
    - **sys.path.append('F:/ComfyUI-master/custom\_nodes/stable-fast-3d/uv\_unwrapper')**
- 

### iv) Hugging Face Authentication

#### 1. Create Token

- Created a token off of the [Hugging Face Token Page](#).

#### 2. Log in

- Authenticated with huggingface-cli login.
  - Set up **HF\_TOKEN** as an environment variable:
    - **set HF\_TOKEN=<your\_token>**
- 

### v) Running the Script

#### 1. Execution

- Ran the main script with:
  - **python run.py demo\_files/examples/chair1.png --output-dir output/ --device cpu**

#### 2. Testing Output

- Verified that the generated **.glb** file imports correctly in Blender.

## vi) Challenges Faced and Resolutions

### 1. Torch Import Issues

- **Problem: ModuleNotFoundError: No module named 'torch'** even after installation.
- **Resolution:**
  - Verified the virtual environment activation.
  - Reinstalled **torch** for the correct platform, CPU.

### 2. Conflicting Python Paths

- **Issue:** Additional Python installs disrupted module resolution.
- **Resolution:**
  - Updated PATH to prioritize the virtual environment.

### 3. Hugging Face Token

- **Issue: Invalid token** error during authentication.
- **Resolution:**
  - Pasted the token correctly and logged in using the CLI.
  - Made sure tokens and permissions are valid on Hugging Face.

### 4. Dependency Installation Issues

- **Issue: subprocess-exited-with-error** while attempting installation of **uv\_unwrapper**.
  - **Resolution:**
    - Manually installed the required dependencies first before trying to install **uv\_unwrapper**.
- 

## vii) Key Learnings and Optimizations

### 1. Effective Dependency Management

- Using CPU-specific dependencies for a system that has no GPU avoids resource constraints.

### 2. Isolation in Environment

- Ensure a clean and isolated environment to prevent conflicts.

### 3. Troubleshooting Python Imports

- Tools like **pip list**, **where python**, and Python's **sys.path** are very useful during debugging.

#### 4. Authentication and Access

- Setting environment variables for tokens provides a more seamless workflow.

### Task-2

#### Documentation: Creating a One-Click Installer

##### Overview

This documentation outlines the steps for developing a one-click installer for the **Stable Fast 3D** project. The installer automates the setup process, including dependency installation, environment configuration, and ensuring offline functionality. The goal is to minimize user input and make the project deployment seamless on any machine.

---

#### Steps to Develop a One-Click Installer

##### Step 1: Understand the Environment

- **Project Dependencies:**
  - Python 3.12 or later.
  - Required libraries (e.g., torch, torchvision, rembg, etc.).
  - Local installations like uv\_unwrapper and texture\_baker.
- **Challenges Identified:**
  - Compatibility with CPU-only environments.
  - Offline installation for dependencies like uv\_unwrapper and texture\_baker.
  - Avoiding issues with environment variables and Python paths.

##### Step 2: Prepare Resources

- Collect and bundle all required dependencies in a requirements.txt file or pre-download wheels for offline installation.
- Gather the necessary Python scripts and ensure they are executable in any environment.

##### Step 3: Write the Installer Script

Installer Script: Key Functionalities

##### 1. Create and Activate a Virtual Environment:

```
python -m venv sf3d_env
source sf3d_env/bin/activate # Linux/Mac
sf3d_env\Scripts\activate   # Windows
```

##### 2. Install Dependencies:

- Online mode:

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

- Offline mode: Use pre-downloaded wheels:

```
pip install ./dependencies/*.whl
```

### 3. Install Local Modules:

```
pip install ./uv_unwrapper/  
pip install ./texture_baker/
```

### 4. Set Environment Variables (if needed):

- Update PATH to include the Python environment:

```
setx PATH "%PATH%;F:\ComfyUI-master\custom_nodes\stable-fast-  
3d\sf3d_env\Scripts"
```

### 5. Verify Installation: Run a test command:

```
python run.py demo_files/examples/chair1.png --output-dir output/ --device cpu
```

---

## Script Implementation

Here's the **installer script**:

```
import os  
import subprocess  
import sys  
  
def run_command(command):  
    """Helper function to execute system commands."""  
    try:  
        subprocess.run(command, shell=True, check=True)  
    except subprocess.CalledProcessError as e:  
        print(f'Error: {e}')  
        sys.exit(1)  
  
def main():  
    print("Starting installation...")  
  
    # Step 1: Create virtual environment  
    if not os.path.exists("sf3d_env"):  
        print("Creating virtual environment...")  
        run_command("python -m venv sf3d_env")  
    else:  
        print("Virtual environment already exists.")  
  
    # Step 2: Activate the environment  
    activate_command = (
```

```

        ".\\sf3d_env\\Scripts\\activate" if os.name == "nt" else "source sf3d_env/bin/activate"
    )
    print("Activating virtual environment...")
    run_command(activate_command)

    # Step 3: Install dependencies
    print("Installing dependencies...")
    run_command("pip install -r requirements.txt")

    # Step 4: Install local modules
    print("Installing local modules...")
    run_command("pip install ./uv_unwrapper/")
    run_command("pip install ./texture_baker/")

    # Step 5: Verify installation
    print("Verifying installation...")
    run_command("python run.py demo_files/examples/chair1.png --output-dir output/ --
device cpu")

    print("Installation complete!")

if __name__ == "__main__":
    main()

```

---

## Challenges and Solutions

- **Issue:** Missing modules like torch during local module installations.
    - **Solution:** Ensure **torch** and dependencies are explicitly installed before running the installer.
  - **Issue:** Dependency conflicts or version mismatches.
    - **Solution:** Pin versions in **requirements.txt** and validate compatibility offline.
  - **Issue:** Offline functionality.
    - **Solution:** Bundle all dependencies as **.whl** files in a **dependencies/** directory.
- 

## Testing the Installer

### 1. Run the script:

```
python installer.py
```

### 2. Verify output:

- Ensure the virtual environment is created.
  - Check that dependencies are installed without errors.
  - Confirm the test command runs successfully.
-

Improvements and Optimizations

- **Bundle Resources:**
  - Package the installer and dependencies into a .zip file for easy distribution.
- **Error Handling:**
  - Add more detailed logs for troubleshooting.
- **Cross-Platform Support:**
  - Test the script on Windows, Linux, and Mac environments.

Screenshots:

```
(sf3d_env) F:\ComfyUI-master\custom_nodes\stable-fast-3d\python run.py demo_files/examples/chair1.png --output-dir output/
Device used: cpu
F:\stable-fast-3d\env\Lib\site-packages\timm\models\layers\_init_.py:48: FutureWarning: Importing from timm.models.layers is deprecated, please import via timm.layers
warnings.warn(f"Importing from {__name__} is deprecated, please import via timm.layers", FutureWarning)
F:\stable-fast-3d\env\Lib\site-packages\open_clip\factory.py:128: FutureWarning: You are using 'torch.load' with 'weights_only=False' (the current default value), which uses the default pickle module implicitly. It is possible to construct malicious pickle data which will execute arbitrary code during unpickling (See https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-models for more details). In a future release, the default value for 'weights_only' will be flipped to 'True'. This limits the functions that could be executed during unpickling. Arbitrary objects will no longer be allowed to be loaded via this mode unless they are explicitly allowlisted by the user via 'torch.serialization.add_safe_globals'. We recommend you start setting 'weights_only=True' for any use case where you don't have full control of the loaded file. Please open an issue on GitHub for any issues related to this experimental feature.
checkpoint = torch.load(checkpoint_path, map_location=map_location)
2024-12-04 23:59:20.6687120 [E:onnxruntime:Default, provider_bridge_ort.cc:1848 onnxruntime::TryGetProviderInfo_TensorRT] D:\a_work\1\onnxruntime\core\session\provider_bridge_ort.cc:1539 onnxruntime::ProviderLibrary::Get [ONNXRuntimeError] : 1 : FAIL : LoadLibrary failed with error 126 "" when trying to load "F:\stable-fast-3d\sf3d_env\Lib\site-packages\onnxruntime\capi\onnxruntime_providers_tensorrt.dll"
***** EP Error *****
EP Error D:\a_work\1\onnxruntime\python\onnxruntime_pybind_state.cc:587 onnxruntime::python::RegisterTensorRTPluginsAsCustomOps Please install TensorRT libraries as mentioned in the GPU requirements page, make sure they're in the PATH or LD_LIBRARY_PATH, and that your GPU is supported.
when using ["TensorrtExecutionProvider", "CUAAExecutionProvider", "CPUExecutionProvider"]
Falling back to ["CUAAExecutionProvider", "CPUExecutionProvider"] and retrying.
2024-12-04 23:59:21.2081353 [E:onnxruntime:Default, provider_bridge_ort.cc:1862 onnxruntime::TryGetProviderInfo_CUDA] D:\a_work\1\onnxruntime\core\session\provider_bridge_ort.cc:1539 onnxruntime::ProviderLibrary::Get [ONNXRuntimeError] : 1 : FAIL : LoadLibrary failed with error 126 "" when trying to load "F:\stable-fast-3d\sf3d_env\Lib\site-packages\onnxruntime\capi\onnxruntime_providers_cuda.dll"
[OK] | 0/1 [00:00?, ?it/s]
After Remesh 9878 19756 | 1/1 [04:28:00-00, 268.42s/it]
```

	input	04-12-2024 23:59	PNG File	112 KB
	mesh	05-12-2024 00:03	3D Object	784 KB

Name: Pulkit Agrawal  
Contact: +91 7340025509  
E-mail: [pulkitag2003@gmail.com](mailto:pulkitag2003@gmail.com)