**Phase 1: Setup, Data Preparation, and Understanding (June 6 - June 12, 2025)**

**Goal**: Set up the environment, understand the paper's methodology, and prepare datasets.

* **June 6-7: Environment Setup and Paper Review**
  + **Tasks**:
    - Set up a Python environment with PyTorch, CUDA, and necessary libraries (e.g., diffusers, transformers, Open3D, PyTorch3D, LPIPS for perceptual loss).
    - Install dependencies for 3D Gaussian Splatting (3DGS) and NeuS (refer to their respective GitHub repositories).
    - Thoroughly read the paper, focusing on Sections 3 (Approach) and 4 (Experiments). Take notes on the V3D pipeline, conditioning, and reconstruction methods.
    - Clone the V3D repository (<https://github.com/heheyas/V3D>) and study the codebase structure.
    - Review referenced papers for Stable Video Diffusion (SVD) [4], 3DGS [35], and NeuS [94] to understand dependencies.
  + **Deliverable**: Working environment with all dependencies installed; a clear understanding of V3D’s pipeline (dense view prediction, reconstruction, mesh extraction).
* **June 8-10: Dataset Acquisition and Preprocessing**
  + **Tasks**:
    - Download the Objaverse dataset [19] for object-centric 3D generation (~290k high-quality 3D meshes as per the paper).
    - Filter low-quality meshes as described in Section 3.2 (manually curate or use provided scripts from the V3D repository).
    - Render 360° orbit videos for each object (18 fixed camera poses, 512x512 resolution, normalized mesh, camera distance=2, elevation=0).
    - Preprocess and store latent and CLIP embeddings for training efficiency (Section 3.2).
    - For scene-level synthesis, download the MVImgNet dataset [107] (~219k posed video clips).
    - Preprocess MVImgNet videos: crop, recenter, and adjust camera parameters using an off-the-shelf background removal tool (e.g., Segment Anything).
  + **Deliverable**: Curated Objaverse dataset with rendered 360° videos; preprocessed MVImgNet dataset ready for fine-tuning.
* **June 11-12: Model and Pipeline Familiarization**
  + **Tasks**:
    - Study the Stable Video Diffusion (SVD) model architecture [4] and download pre-trained weights from Hugging Face or the official repository.
    - Explore the V3D codebase to understand how SVD is modified for 3D generation (e.g., removal of motion bucket ID, FPS ID, and elevation conditioning).
    - Set up a small test pipeline to generate multi-view images from a single image using the pre-trained SVD model (without fine-tuning).
    - Test 3DGS and NeuS implementations on sample data to ensure compatibility with your setup.
  + **Deliverable**: Initial test run of SVD generating multi-view images; familiarity with 3DGS and NeuS reconstruction pipelines.

**Phase 2: Fine-Tuning Video Diffusion Model (June 13 - June 19, 2025)**

**Goal**: Fine-tune the video diffusion model for object-centric and scene-level tasks.

* **June 13-15: Object-Centric Fine-Tuning**
  + **Tasks**:
    - Implement the fine-tuning pipeline for object-centric 3D generation using the Objaverse dataset (Section 3.2).
    - Modify SVD conditioning: inject CLIP embeddings via cross-attention, concatenate front-view latent, and remove irrelevant conditions (motion bucket ID, FPS ID).
    - Use the EDM framework [34] with a large noise distribution (P\_mean=1.5, P\_std=2.0) and classifier-free guidance (20% dropout for latent and CLIP embeddings).
    - Fine-tune for ~22.5k steps (approximately 1 day on a single GPU, as per the paper). Start with a small subset to debug.
    - Validate the model by generating multi-view images for a few test objects and checking for 360° consistency.
  + **Deliverable**: Fine-tuned video diffusion model capable of generating consistent 360° multi-view images from a single image.
* **June 16-18: Scene-Level Fine-Tuning**
  + **Tasks**:
    - Integrate a PixelNeRF encoder [106] into the SVD model for scene-level novel view synthesis (Section 3.4).
    - Implement conditioning with multiple input images and camera poses using PixelNeRF feature maps concatenated to the U-Net input.
    - Fine-tune on the MVImgNet dataset with down-sampled photometric loss for the PixelNeRF encoder (Section 3.4, Equation 2).
    - Randomly select 1-4 input images with 20% dropout for conditioning, as described in Section 4.
    - Fine-tune for one epoch on a 10-category subset of the CO3D dataset [71] for comparison with prior methods (Section 4.2).
    - Test the model on sparse input views (2, 3, or 6 views) to generate novel views along a specified camera path.
  + **Deliverable**: Fine-tuned model for scene-level novel view synthesis with precise camera path control.
* **June 19: Validation and Debugging**
  + **Tasks**:
    - Evaluate multi-view consistency for object-centric generation using qualitative checks (visual inspection of 360° videos).
    - For scene-level synthesis, compare generated views against ground truth from CO3D using metrics like PSNR, SSIM, and LPIPS (Table 3).
    - Debug any inconsistencies (e.g., blurry textures, geometric errors) by adjusting hyperparameters or noise schedules.
  + **Deliverable**: Validated models with initial results for object-centric and scene-level tasks; debugged issues.

**Phase 3: 3D Reconstruction and Mesh Extraction (June 20 - June 26, 2025)**

**Goal**: Implement and optimize the 3D reconstruction and mesh extraction pipelines.

* **June 20-22: 3D Gaussian Splatting Reconstruction**
  + **Tasks**:
    - Implement the 3DGS reconstruction pipeline (Section 3.3) using generated multi-view images.
    - Use perceptual loss (LPIPS [110]) combined with MSE and D-SSIM (Equation 1) to handle view inconsistencies.
    - Implement space-carving initialization: segment foregrounds with a background removal tool, project voxels to create an occupancy grid, and use marching cubes [54] to initialize Gaussian positions.
    - Optimize 3DGS for ~3-minute reconstruction time, testing on a few objects.
    - Validate reconstruction quality by rendering 360° views and comparing with input images.
  + **Deliverable**: Functional 3DGS pipeline producing high-quality 3D Gaussians within 3 minutes.
* **June 23-25: Mesh Extraction Pipeline**
  + **Tasks**:
    - Implement the NeuS-based mesh extraction pipeline (Section 3.3) with multi-resolution hash grid [61].
    - Add normal smooth loss and sparse regularization loss to handle sparse views (<40 images).
    - Refine mesh textures using LPIPS loss with generated multi-views, keeping geometry fixed (15-second refinement, Section 3.3).
    - Test mesh extraction on sample objects, ensuring high-fidelity textures and reasonable geometry.
  + **Deliverable**: Mesh extraction pipeline producing textured meshes from multi-view images.
* **June 26: Integration and Testing**
  + **Tasks**:
    - Integrate the fine-tuned video diffusion model with 3DGS and NeuS pipelines to form the complete V3D framework.
    - Test the end-to-end pipeline: input a single image, generate multi-view images, and reconstruct 3D Gaussians or meshes.
    - Evaluate results qualitatively (visual quality) and quantitatively (e.g., Chamfer distance for point clouds, as in Section 4.2).
  + **Deliverable**: Integrated V3D pipeline capable of generating 3D assets from a single image.

**Phase 4: Evaluation, Ablation Studies, and Documentation (June 27 - July 5, 2025)**

**Goal**: Evaluate performance, conduct ablation studies, and document findings.

* **June 27-29: Quantitative and Qualitative Evaluation**
  + **Tasks**:
    - Evaluate object-centric generation on the Google Scanned Objects dataset [20] using metrics like PSNR, SSIM, and LPIPS (Table 2).
    - Compare with baselines (SyncDreamer [50], Wonder3D [52], LGM [85], TriplaneGaussian [117]) using provided scripts or re-implemented metrics.
    - For scene-level synthesis, evaluate on the CO3D dataset [71] with 2, 3, and 6 input views (Table 3).
    - Conduct a small user study (if feasible) to assess alignment and fidelity, mimicking Table 1.
    - Generate 360° videos for qualitative comparison (as in Figures 3 and 4).
  + **Deliverable**: Evaluation results with metrics and visualizations; comparison with state-of-the-art methods.
* **June 30 - July 2: Ablation Studies**
  + **Tasks**:
    - Perform ablation studies (Section 4.3) to validate key design choices:
      * Pre-training: Train a model from scratch on Objaverse for 45k steps and compare with the fine-tuned model.
      * Fine-tuning steps: Test different step counts (e.g., 10k, 22.5k, 30k) to assess texture and geometry quality.
      * Noise distribution: Compare large noise (P\_mean=1.5, P\_std=2.0) vs. smaller noise (P\_mean=0.7, P\_std=1.6).
      * Camera conditioning: Test learnable camera embeddings and Plücker ray embeddings vs. PixelNeRF (Table 3).
    - Document qualitative and quantitative results (e.g., visual comparisons as in Figure 6).
  + **Deliverable**: Ablation study results highlighting the importance of pre-training, noise distribution, and conditioning.
* **July 3-5: Documentation and Presentation**
  + **Tasks**:
    - Write a detailed report summarizing the implementation, experiments, and findings. Include:
      * Overview of the V3D pipeline.
      * Challenges faced (e.g., dataset curation, fine-tuning stability).
      * Evaluation results and comparisons with baselines.
      * Ablation study insights.
    - Prepare a presentation for your university team, including visualizations (360° videos, reconstructed meshes, point clouds).
    - If required, submit code and documentation to your supervisor or repository.
    - Back up all code, models, and results.
  + **Deliverable**: Comprehensive report, presentation, and organized codebase.

**Notes and Tips**

* **Resource Management**: Ensure access to GPUs (e.g., NVIDIA A100 or V100) for fine-tuning and reconstruction. Use cloud platforms (e.g., AWS, Google Cloud) if university resources are limited.
* **Time Management**: Prioritize object-centric generation if time is constrained, as it’s the core of V3D. Scene-level synthesis can be scaled down (e.g., test on fewer CO3D categories).
* **Debugging**: Monitor training for overfitting (simplistic textures) or underfitting (inaccurate geometry). Adjust fine-tuning steps or noise schedules as needed.
* **Collaboration**: If stuck, consult the V3D GitHub issues or reach out to the authors (e.g., via email provided in the paper: [chenzl22@mails.tsinghua.edu.cn](mailto:chenzl22@mails.tsinghua.edu.cn)).
* **Supplemental Material**: Check the V3D repository for supplemental material (training details, failure cases) to guide implementation.

This plan balances setup, implementation, evaluation, and documentation within the one-month timeframe. Adjust tasks based on your progress and feedback from your supervisor. Good luck with your internship!