Leveraging 3D Structure for Robust and Scalable Vehicle Panel Segmentation

Chung-Yu Wei[1]

MSc Data Science and Machine Learning

Industry Supervisor: Dr.Dimitri Zhukov

Academic supervisor: Professor Kaan Akşit

September 2025

Abstract

The manual annotation of vehicle components for AI models is a costly bottleneck in the automotive industry. This report presents a pipeline to automate this process by transferring 2D segmentation masks between images via a 3D representation.

The methodology leverages a sparse 3D model for camera poses and high-fidelity depth maps for surface geometry. The pipeline was systematically enhanced from a naive projection baseline to include robust geometric fitting, a full visibility check to resolve occlusions, and point cloud subsampling for performance optimization. A final multi-view aggregation strategy, the Four Corners method, was developed to combine projections from diverse viewpoints for maximum completeness.

The final, tuned pipeline demonstrated a significant improvement in the mean Intersection over Union (mIoU) over its baseline, while the optimization provided a dramatic increase in computational speed with only a minor loss in accuracy. The complete system offers a powerful solution for automating vehicle annotation, with direct applications in streamlining damage assessment workflows.

For the accompanying code, see: `https://github.com/weinana1005/3D-mask-transfer/tree/main`

Contents

Chapter 1

Introduction

In the field of artificial intelligence for automotive damage assessment, the quality and quantity of training data are paramount. However, the manual creation of pixel-perfect segmentation masks for vehicle components is a significant bottleneck, demanding considerable time and manual effort. This process slows down the development and improvement of AI models that are critical for the industry.

This project tackles this challenge directly by developing a pipeline that leverages 3D geometry to automatically transfer component masks from a single annotated image to multiple other views of the same vehicle. By utilizing a 3D model as a geometric bridge, our system can project annotations through 3D space, aiming to drastically reduce the manual labor required for dataset creation and augmentation.

This report details the iterative development of this mask transfer system, from initial baseline experiments to a final, optimized pipeline. We will explore various methodologies, analyze their performance, and discuss the key challenges and learnings involved in bridging 2D images with 3D scene understanding. The ultimate goal is to provide a robust and efficient solution for accelerating the data annotation workflow.

Chapter 2

Dataset

2.1 Overview

The foundation of this project is the **3DRealCar dataset**, an open-source collection of data designed for creating high-fidelity 3D models of vehicles [1]. The dataset is publicly available and can be utilized for commercial purposes, making it an excellent resource for academic and industrial research. It provides a rich combination of 2D images and dense 3D reconstructions of various cars, which is essential for the task of transferring segmentation masks between different views of the same vehicle.

The primary strength of the 3DRealCar dataset lies in its detailed and multi-modal data, captured using a professional 3D scanner. This allows for a precise mapping between the 2D images and the 3D surface of the car, a critical component for our proposed mask transfer pipeline.

2.2 Data Sources and Preprocessing

The success of the projection pipeline is fundamentally dependent on the quality and fusion of several distinct data sources. Each source provides a critical piece of information, from geometric structure to visual appearance. The following subsections detail each data type, its role, and any required preprocessing steps.

2.2.1 Sparse 3D Model and other input Data

The foundational geometric reference for this work is a sparse 3D model generated using the COLMAP Structure-from-Motion (SfM) photogrammetry pipeline [2]. This model is essential for establishing the geometric relationship and coordinate systems between all camera views, providing the camera intrinsic parameters and the precise 3D pose for each image. The model itself is composed of several key files:

`cameras.bin` A database that stores the intrinsic parameters (focal length, principal point, distortion coefficients) for every unique camera model used during the capture process.

`images.bin` This file acts as the primary link between the 2D images and the 3D world. For each image, it records the camera's extrinsic pose (position **tvec** and orientation **qvec**), a list of its detected 2D keypoints, and a crucial list of `point3D_ids` that links each 2D keypoint to its corresponding 3D point.

`points3D.bin` This file stores detailed information for every successfully reconstructed sparse 3D point. Each entry includes its unique `point3D_id`, its precise position (X, Y, Z) in the world coordinate system, its RGB color, and a "track" recording which 2D keypoints in which images observe it.

`points3D.ply` An exported, human-readable version of the data in `points3D.bin`. It stores the XYZ coordinates and colors of all 3D points in the standard Polygon File Format (`.ply`), which can be opened by visualization software like MeshLab to inspect the sparse point cloud.

High-Fidelity Depth Maps. Geometric ground truth is provided by 16-bit grayscale `.png` images (256 x 192 pixels) from a dedicated 3D scanner, where each pixel's value corresponds to a precise distance measurement. The high accuracy of this data is crucial for bypassing the inaccuracies of the sparse model. A critical preprocessing step was to **upsample** these low-resolution depth maps by a factor of 7.5x to match the RGB image dimensions, using **nearest-neighbor interpolation** (`cv2.INTER_NEAREST`) to preserve the integrity of the original measurements.

Besides the data generated from COLMAP. These additional Input data we also need to prepared:

High-Resolution RGB Images. The visual foundation of the dataset consists of standard `.jpg` color images captured by the primary vehicle cameras at a resolution of 1920 x 1440 pixels. These images serve as the high-resolution reference for the scene, defining the 2D component masks and providing the target dimensions for upsampling other data sources.

Component Segmentation Masks. The final data source consists of pixel-perfect annotations for 83 distinct vehicle components (e.g., "door_front_left"), stored in a COCO-format JSON file. This data defines the precise regions of interest on the source images that are to be transferred and serves as the ground truth on the target images for the final IoU evaluation. An Example for components segmentation masks can be seen in Fig2.2.

While the full dataset contains a wide variety of vehicles, for the purposes of a controlled and in-depth analysis, the experiments detailed in the following chapters will focus exclusively on two distinct cars. These vehicles were chosen to represent different classes and visual characteristics. All methodologies are developed and evaluated using these two specific datasets[3]:

Table 2.1: Specifications of the two vehicle datasets used for analysis.

Dataset ID	**Brand**	**Type**	**Vehicle Type**	**Color**	**Gloss**
2024_06_04_13_44_39	HongQi	H5	Sedan	White	Standard
2024_07_02_15_18_44	Honda	Odyssey	MPV	Black	Standard

The two vehicles example RGB images are shown in Figure 2.1.

(a) Dataset 1: The white HongQi H5 Sedan.

(b) Dataset 2: The black Honda Odyssey MPV.

Figure 2.1: The two vehicles selected for all subsequent experiments and analysis in this study. (Source image: 3Drealcar dataset. For details can be found by searching dataset ID[3]) Github: 3D-mask-transfer/Source images/RGB examples

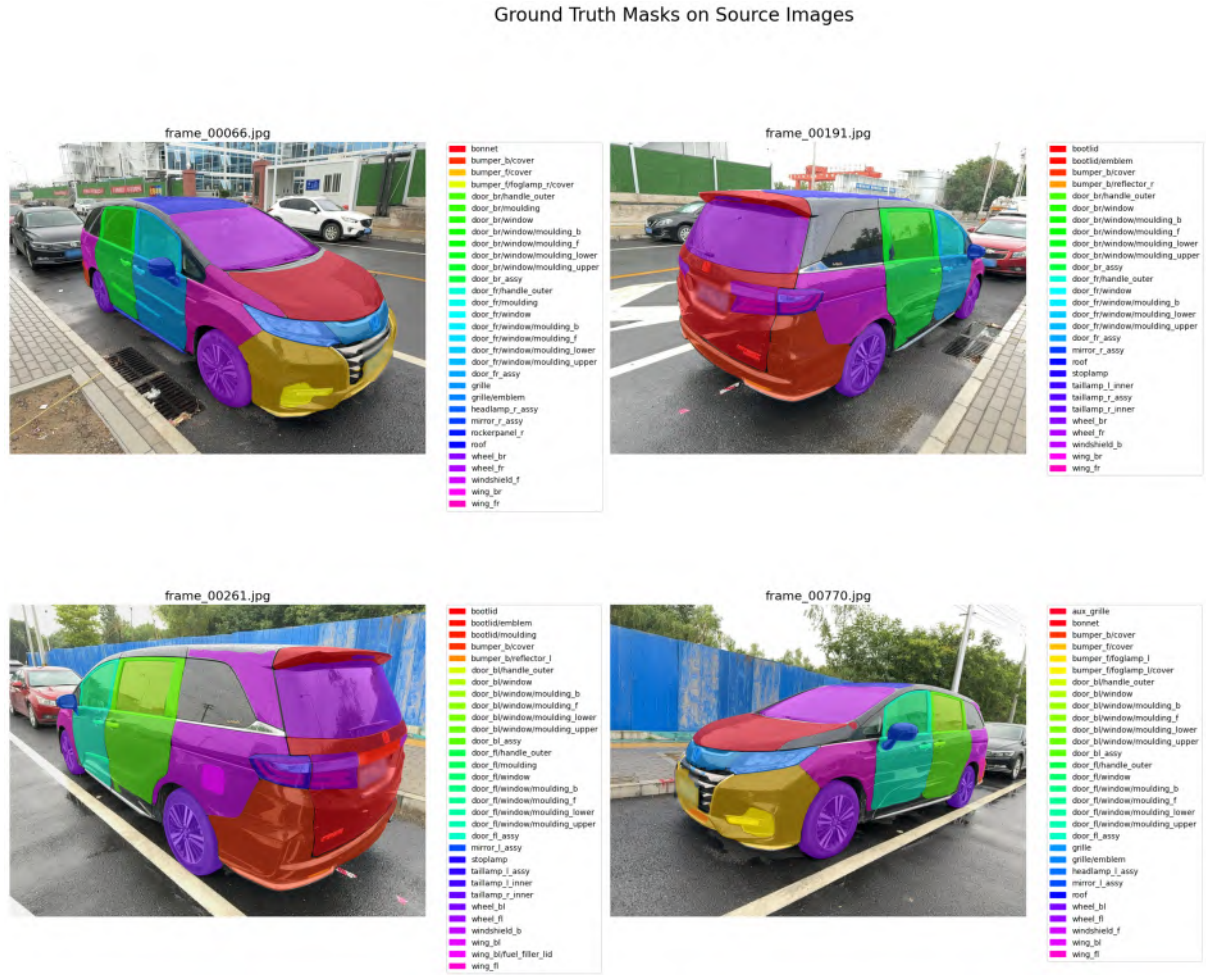

Figure 2.2: This is the visualization of ground truth mask of components images loaded from json file provided by Tractable. Every components that can be seen from the image are stored inside the json file, and been visualised in the notebook. It does not guaranteed to have 100% accuracy. However, manually annotation can be done instead to achieve better correctness (Source image: Generated by: Tractable, loaded from: `masktransfer_finalVer2.ipynb` Github: 3D-mask-transfer/Source images/json file)

Chapter 3

2D-to-2D Keypoint Matching via 3D Scene Geometry

This chapter presents the methodologies developed for establishing robust 2D-to-2D correspondences between images of a rigid object from different viewpoints. The core challenge in direct 2D matching lies in handling significant changes in perspective, scale, and occlusion. To overcome these challenges, the methods detailed herein leverage an intermediate 3D representation of the scene as a stable, view-invariant bridge. By projecting a point from a source image into this 3D space and subsequently re-projecting it into a target image, a geometrically consistent correspondence can be found. Two distinct approaches are investigated: first, a baseline method employing a sparse 3D point cloud, and second, our primary proposed method that utilises a dense depth map to enable correspondence for any pixel on the object's surface. A detailed analysis of the workflow, mathematical underpinnings, assumptions, and limitations is provided for each method.

3.1 Method 1: Sparse Correspondence using Pre-computed 3D Features

The initial approach serves as a baseline for performance evaluation and is founded upon the sparse 3D reconstruction generated by state-of-the-art Structure-from-Motion (SfM) software. This method is constrained to a sparse set of visually salient feature points identified during the SfM process.

Figure 3.1: This is the visualization of Sparse Points (Purple dots) generated by COLMAP with 3 different views. (Source image: Generated by `masktransfer_finalVer1.ipynb` Github: 3D-mask-transfer/Source images/Sparse Points)

3.1.1 Principle of Operation

The fundamental principle is to treat the pre-computed SfM model, specifically the output from COLMAP [2], as a static database. A correspondence is established not by calculating new geometric information, but by querying the existing 3D structure. A known 2D feature sparse point in a source image is used to look up its corresponding 3D sparse point coordinates in the model, which is then projected into the desired target view. This process is illustrated in Figure 3.2.

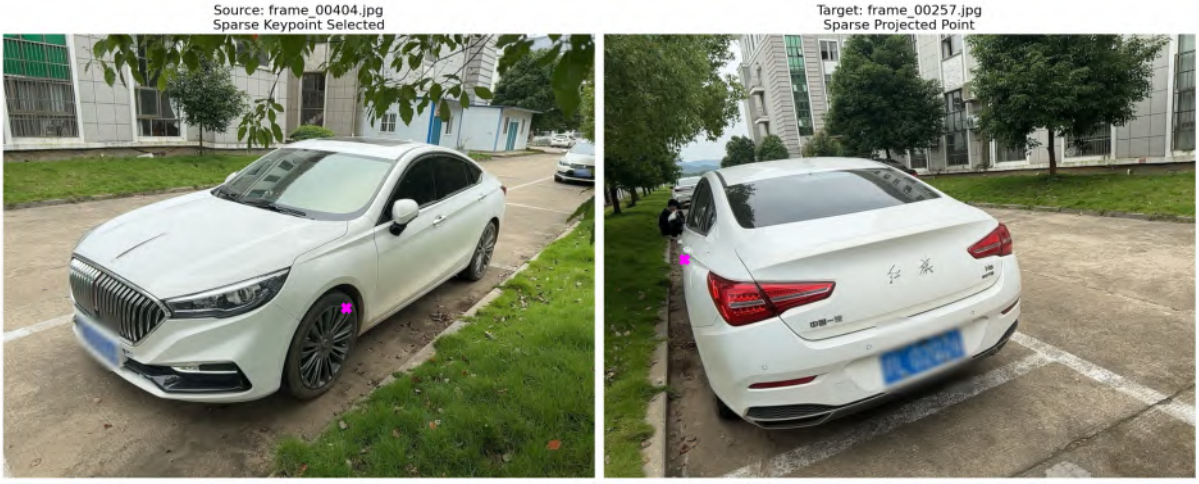

Figure 3.2: Conceptual diagram of the 3D projection pipeline. A point p_s in a source view is back-projected to a 3D point P_w using its depth, then re-projected to point p_t in a target view. (Source image: Generated by `masktransfer_finalVer1.ipynb` Github: 3D-mask-transfer/Source images/Sparse points projection)

3.1.2 Methodology and Implementation

The execution of this method requires a complete SfM model and a segmentation mask for the object of interest. The workflow is detailed in Algorithm 1.

Algorithm 1: Sparse 2D-to-2D Correspondence

Input: Source image I_s, Target image I_t, COLMAP model $\mathcal{M}$, Vehicle mask M_s

Output: Projected 2D point p_t in I_t

1. Keypoint Filtering:

Let $\mathcal{K}_s$ be the set of all 2D keypoints in I_s from $\mathcal{M}$.

Filter $\mathcal{K}_s$ using mask M_s to obtain a candidate set $\mathcal{K}_{cand} \subset \mathcal{K}_s$ of points on the vehicle surface.

2. Point Selection:

Randomly select a source keypoint $p_s \in \mathcal{K}_{cand}$.

Retain its original index, i, from the COLMAP keypoint list.

3. 3D Position Lookup:

Using index i, retrieve the corresponding 3D point ID, `point3D_id`, from the `images.bin` file in $\mathcal{M}$.

Use `point3D_id` to query the `points3D.bin` file and retrieve the 3D world coordinate P_w.

4. Re-projection:

Let C_t be the camera parameters (pose and intrinsics) for the target image I_t.

Project P_w into the target image frame: $p_t = \text{Project}(P_w, C_t)$.

3.1.3 Assumptions and Limitations

The viability of this method rests on three key assumptions: (i) the geometric accuracy of the camera poses computed by COLMAP, (ii) the rigidity of the vehicle, precluding any deformation between image captures, and (iii) the pixel-level accuracy of the provided vehicle segmentation mask.

The principal limitation is its inherent **sparsity**. Correspondence is restricted to the thousands of salient feature points that COLMAP detects. This method fails for arbitrary user-selected pixels, particularly on texture-deficient surfaces common on vehicles, such as painted panels, windows, or the hood, where no keypoints are likely to exist.

3.2 Proposed Method: Dense Correspondence via Depth Projection

To address the sparsity limitations of the baseline, we propose a dense correspondence method. This technique leverages per-pixel depth information, enabling the projection of any point on the object's surface, not just pre-determined feature points.

3.2.1 Principle of Operation

The core principle is the real-time calculation of a 3D world coordinate for any given pixel. This is achieved by combining the pixel's 2D coordinate with its corresponding depth value, which is retrieved from an aligned depth map. This calculated 3D point is then transformed and re-projected into the target view, establishing a dense correspondence field.

3.2.2 Mathematical Formulation and Methodology

The execution of the dense method relies on the camera parameters from the SfM model and a per-pixel depth map aligned with the source image. The complete mathematical workflow is detailed in Algorithm 2.

Algorithm 2: Dense 2D-to-2D Correspondence via Depth Projection

Input: Source image I_s, Target image I_t, COLMAP model $\mathcal{M}$, Source depth map D_s, Source point $p_s = (u, v)$

Output: Projected 2D point p_t in I_t

1. Depth Retrieval:

Read the depth value d for pixel p_s from the depth map D_s.

2. 3D Position Calculation:

This step computes the 3D world coordinate P_w from the 2D point p_s and its depth d.

(a) Back-projection to Camera Coordinates (P_c): Use the source camera's intrinsic matrix K_s to back-project p_s into the camera's local 3D space.

$$P_c = d \cdot K_s^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

(b) Transformation to World Coordinates (P_w): Use the source camera's pose (rotation R_s, translation t_s) to transform P_c into the global world frame.

$$P_w = R_s^T(P_c - t_s)$$

3. Re-projection to Target Image:

This step projects the 3D world point P_w into the target image to find the final coordinate $p_t = (u', v')$.

(a) Transformation to Target Camera Coordinates (P'_c): Use the target camera's pose (R_t, t_t) to bring P_w into the target camera's frame.

$$P'_c = R_t P_w + t_t$$

(b) Projection to Pixel Coordinates (p_t): Let $P'_c = (X', Y', Z')$. Use the target camera's intrinsics K_t to project P'_c onto the image plane.

$$\begin{bmatrix} u' \\ v' \\ 1 \end{bmatrix} = \frac{1}{Z'} K_t \begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix}$$

3.2.3 Assumptions and Limitations

This method shares the assumptions of camera pose accuracy and object rigidity with the baseline approach. However, it introduces a critical third dependency: **the quality and alignment of the depth map**. The accuracy of the projection is directly contingent on the precision of the depth data and its perfect spatial alignment with the source RGB image.

Several limitations must be noted. First, the method is entirely dependent on the availability of a high-quality depth map. Second, its performance is sensitive to noise in the depth data, which can be significant on specular or transparent surfaces like car paint and glass. Finally, the current formulation does not explicitly handle occlusions; a point visible in the source view may be occluded in the target view, leading to an erroneous projection onto a foreground object.

3.3 Validation: Quantifying Sparse Model Inaccuracy

To provide a quantitative justification for preferring the dense, depth-based approach over the sparse SfM model, a dedicated experiment was conducted to measure the geometric inaccuracy of the sparse reconstruction. This analysis was designed to isolate and quantify the error introduced by the SfM model's geometry, independent of any specific mask transfer task. The findings validate the necessity of the high-fidelity depth maps for achieving precision correspondence.

3.3.1 Experimental Methodology

The experiment measured the re-projection error, defined as the discrepancy between a point's projected position using the sparse model's geometry versus its position derived from the ground-truth 3D scanner data. The process was as follows:

1. **Point Selection**: A 2D keypoint, p_s, detected by COLMAP in a source image was selected. This keypoint has a corresponding 3D point, P_{sparse}, in the global coordinate system of the SfM model.
2. **Sparse Model Projection**: The 3D point P_{sparse} was projected into a designated target camera's view using that camera's pose and intrinsic parameters from the COLMAP model, resulting in a 2D coordinate, $p_{t,sparse}$.
3. **Ground-Truth Projection**: To establish a ground-truth correspondence, we used the high-fidelity depth map. At the coordinate of the original keypoint p_s in the source image, we retrieved its precise, scanner-measured depth, d_{true}. This depth value was used to back-project p_s into a "true" 3D point, P_{true}. This point was then projected into the same target camera's view, resulting in a ground-truth 2D coordinate, $p_{t,true}$.
4. **Error Calculation**: The re-projection error was calculated as the Euclidean distance in pixels between the two projected points in the target image: $E = ||p_{t,sparse} - p_{t,true}||_2$.

3.3.2 Results and Conclusion

The execution of this test across numerous keypoints revealed significant and unpredictable re-projection errors. The error magnitudes frequently exceeded several pixels, with outliers reaching tens of pixels—a level of inaccuracy unacceptable for precise pixel-level tasks like mask transfer.

The analysis confirms that the errors stem not from the projection mathematics, but from the inherent geometric inaccuracies of the sparse SfM model. The primary causes are:

- **Geometric Drift**: SfM reconstructions are known to accumulate small errors in camera poses and 3D point locations. Over a large object, this results in a slight warping or scaling of the model's geometry relative to the metrically accurate ground truth from the 3D scanner.
- **Lack of Surface Constraint**: The sparse feature points are not explicitly constrained to a single, coherent surface and may "float" slightly in front of or behind the true physical surface, leading to large projection errors from novel viewpoints.

This experiment provides the quantitative evidence to support our choice of methodology. It demonstrates that while the sparse model is invaluable for establishing camera poses, its 3D point cloud is not geometrically reliable enough for high-precision correspondence. This result validates the necessity of the dense, depth-map-based methodology (detailed in Section 3.2) for achieving the accuracy required by this project.

Figure 3.3: A conceptual visualization of the re-projection error analysis. The green dot represent projected locations using the sparse SfM model, while the purple dot show ground-truth locations derived from the depth scanner. The connecting line illustrate the error vector. The following chapter will discuss several methods to reduce the error and improve projection (Source image: Generated by `masktransfer_finalVer1.ipynb` Github: 3D-mask-transfer/Source images/Max projection error)

Chapter 4

Component mask transfer using dense method

4.1 Introduction

This chapter details the investigation and development of a robust pipeline for transferring two-dimensional (2D) component segmentation masks between different camera views, using a three-dimensional (3D) model as an intermediary. The primary objective is to automate the annotation process by projecting an existing mask from a source image onto a target image with high geometric fidelity.

While the dataset contains annotations for numerous vehicle parts, this work places a specific emphasis on a curated set of components selected for their strategic importance to Tractable's core business operations in automated damage assessment. The accurate and efficient segmentation of these parts is a primary driver for the company's AI-powered solutions. The target components for this study are:

- `bonnet`
- `bumper_f/cover` (Front Bumper)
- `windshield_f` (Front Windshield)
- `headlamp_l_assy` (Left Headlamp Assembly)
- `mirror_l_assy` (Left Mirror Assembly)
- `grille`
- `door_fl_assy` (Front-Left Door Assembly)

The successful transfer of these specific masks is paramount for improving the efficiency and accuracy of automated vehicle inspection.

To achieve this, we developed a dense projection pipeline that was systematically enhanced to address key challenges inherent in 3D-to-2D projection. The pipeline's accuracy was quantitatively validated using the primary metrics: the industry-standard **Intersection over Union (IoU)**, applied when a ground-truth component mask was available in the target view. The following sections describe the data prerequisites, the iterative development of the transfer methodology, and an analysis of the results.

The final mask transfer pipeline was developed through an iterative process, starting with a baseline analysis and progressively incorporating more sophisticated techniques to address observed failures.

4.2 V1: A Foundational Dense Projection Baseline Method

The V1 pipeline constitutes the foundational implementation of a dense mask transfer. It serves as the most direct and unadulterated application of the projection principle, designed to establish a functional baseline and clearly identify the primary challenges involved in transferring dense pixel regions.

The core methodology of V1 is a "brute-force" projection. For every single pixel located within the boundary of a source component mask, the system uses its corresponding depth value to calculate its precise 3D world coordinate. This 3D point is then immediately projected into the target camera's view. This process is repeated for all pixels in the source mask.

4.2.1 Implementation

The implementation follows a direct, four-step sequence:

1. **Data Preprocessing**: The low-resolution source depth map is upsampled using nearest-neighbor interpolation to match the dimensions of the high-resolution source RGB image and its corresponding component mask.

2. **Point Cloud Generation**: The pipeline iterates through all pixel coordinates (u, v) inside the source mask. Each coordinate is combined with its depth value to back-project it into a 3D point P_w in the world coordinate system.

3. **Direct Projection**: Every generated 3D point P_w is projected into the target image plane, resulting in a list of 2D coordinates (u', v').

4. **Mask Reconstruction**: A blank, black mask is created with the target image's dimensions. The projected 2D coordinates are then "drawn" onto this mask by setting the corresponding pixel values to white.

4.2.2 Results and Critical Limitations

The output of the V1 pipeline is not a solid, usable segmentation mask. As stated in the initial hypothesis and confirmed by experimentation, this direct projection method produces a visually disjointed "point cloud" in the target image. This outcome, conceptualized in Figure 4.1, is a direct consequence of two fundamental flaws.

Figure 4.1: Conceptual visualization of the V1 pipeline's output. The direct, raw projection of source pixels results in a sparse and disjointed point cloud in the target image, failing to form a coherent mask. (Source image: Generated by `masktransfer_finalVer1.ipynb` Github: 3D-mask-transfer/Source images/Component transfer output)

- **Lack of Surface Continuity**: The primary failure is the inability to produce a solid shape. Due to perspective distortion and the discrete nature of pixel grids, pixels that are adjacent in the source image are not guaranteed to be adjacent after projection. This results in a sparse output with significant gaps between the rendered pixels.
- **Failure to Handle Any Occlusion**: This simplistic method completely fails to account for perspective occlusions. It lacks any depth-checking mechanism, such as a Z-buffer. As a result, points from the far side of an object are projected just as readily as points from the near side, often appearing incorrectly on top of them. This applies to both self-occlusion and occlusion by other objects in the scene.

In conclusion, the V1 method successfully demonstrates the core concept of dense point transfer but is fundamentally impractical for producing usable segmentation masks. Its clear and predictable failures—the lack of continuity and the inability to handle occlusion—precisely define the essential problems that subsequent versions of the pipeline must solve.

4.3 V2: Geometric Model Fitting for Robust Transfer

The V2 pipeline was developed to overcome the critical limitations of the V1 method, specifically its tendency to distort component shapes and its high sensitivity to noise in the source depth data. The central principle of V2 is to introduce a crucial intermediate step of geometric regularization. Instead of directly projecting the raw, and potentially noisy, 3D point cloud, this method first analyzes the cloud to fit an idealized mathematical model of the component's surface. This clean, geometrically coherent model is then projected, resulting in a transfer that preserves the true shape of the component with much higher fidelity.

4.3.1 Methodology and Implementation

The V2 pipeline is a multi-stage process that incorporates data preprocessing, 3D geometric analysis, and robust projection techniques. The workflow, as implemented in the `transfer_mask_v2` function, proceeds through five distinct steps.

1. **Depth Map Preprocessing (In-painting)**: The process begins by addressing imperfections, or holes, in the source depth map. These are often caused by sensor limitations on specular or transparent surfaces. To create a complete surface for modeling, these holes are filled using the Navier-Stokes based in-painting algorithm provided by `cv2.inpaint`[4]. This technique treats the image intensity as a 2D fluid and propagates information from the boundary of the hole inwards, filling the gap in a physically plausible manner.
2. **3D Point Cloud Generation**: Using the now complete, in-painted depth map, a 3D point cloud $\mathcal{P} = \{p_1, p_2, ..., p_N\}$ is generated by back-projecting every pixel within the source component mask.
3. **Geometric Model Fitting**: This is the core innovation of the V2 pipeline. The raw point cloud $\mathcal{P}$ is processed to create a clean, idealized geometric representation.
    - **For Planar Components (RANSAC)**: For components classified as planar, a **RANSAC (RANdom SAmple Consensus)** algorithm is applied. RANSAC is an iterative, non-deterministic method designed to be highly robust to outliers. The process is as follows:
        (a) A minimal sample of 3 non-collinear points is randomly selected from $\mathcal{P}$.

(b) These points define a candidate plane with the equation $ax + by + cz + d = 0$.

(c) For all other points $p_i = (x_i, y_i, z_i)$ in $\mathcal{P}$, the perpendicular distance to this plane is calculated. If the distance is less than a predefined inlier threshold ϵ, the point is added to a consensus set.

$$\text{Distance}(p_i, \text{plane}) = \frac{|ax_i + by_i + cz_i + d|}{\sqrt{a^2 + b^2 + c^2}} < \epsilon$$

(d) Steps (a)-(c) are repeated for a set number of iterations. The plane with the largest consensus set (the most inliers) is selected as the final model. This process effectively filters out all noisy depth measurements, yielding a perfectly flat set of inlier points for projection.

- **For Curved Components (Delaunay Triangulation)**: For non-planar components, a **3D Delaunay triangulation** is used to generate a continuous mesh. The Delaunay triangulation has a critical mathematical property: the empty circumsphere condition. For every tetrahedron in the mesh, its circumsphere (the unique sphere passing through its four vertices) contains no other points from the input cloud $\mathcal{P}$ in its interior.

  This property maximizes the minimum angle of all triangles in the mesh, avoiding long, skinny triangles and creating a well-shaped, smooth surface representation. The final set of points to be projected are the unique vertices of this resulting mesh, which captures the component's underlying curvature while averaging out noise.

4. **Projection with Z-Buffering**: The clean set of points from the geometric model (either the RANSAC inliers or the mesh vertices) is then projected into the target view. This process utilizes a Z-buffer to correctly handle self-occlusion, ensuring that only the surfaces visible from the target camera's perspective are rendered.

5. **Final Mask Consolidation**: Even when projecting from a clean model, small gaps can appear in the resulting 2D mask. Therefore, a final morphological closing operation is applied. Using a moderately sized kernel and several iterations, this step consolidates the projected points into a single, solid, and coherent final mask.

4.3.2 Results and Discussion

The introduction of geometric fitting marked a significant improvement in transfer quality. The V2 pipeline produces masks that are far more geometrically faithful and robust to noise than the V1 method. However, its effectiveness remains fundamentally constrained by the quality of the initial point cloud derived from the upsampled depth map.

Figure 4.2 provides a clear case study of the V2 pipeline's performance on the ''bonnet'' component. The method successfully transfers the general location and shape of the component despite a significant change in viewpoint, demonstrating a clear advantage over the V1 method's distorted, "blobby" output.

Despite the improvements, the figure also reveals several characteristic artifacts that highlight the method's limitations. The most noticeable flaw is the "streaky" or "scanline" pattern visible in the transferred mask (Figure 4.2b). This is a direct consequence of the Delaunay triangulation algorithm operating on a point cloud that retains a grid-like structure from the upsampled low-resolution depth map.

Furthermore, the transferred mask exhibits incomplete coverage and small, erroneous flecks of projection noise. These visual inaccuracies are quantitatively reflected in the modest IoU score of 0.4870.

Figure 4.2: A representative result from the V2 pipeline for the ''bonnet'' component. Panel (a) shows the source mask, (b) displays the V2 transferred mask, and (c) provides the ground-truth mask for comparison. The result highlights the method's strengths in perspective handling alongside its characteristic flaws, such as patterned artifacts. (Source image: Generated by `masktransfer_finalVer1.ipynb` Github: 3D-mask-transfer/Source images/Component transfer output)

While this represents a significant 59% improvement over the 0.3063 IoU achieved by the V1 pipeline, it also highlights the remaining geometric flaws that prevent the method from achieving a higher score.

In summary, the V2 pipeline successfully solves the problem of shape distortion and demonstrates a powerful capability for preserving intended geometry. Its key improvements in shape fidelity and noise robustness were a major milestone. However, its performance is still highly dependent on the quality of the input data. Most critically, V2 inherits the fundamental limitation of V1: it fails to handle inter-object occlusions. Its Z-buffer operates only on the component's own geometry and has no knowledge of the wider scene context in the target view. This persistent problem was the primary motivation for the development of the V3 pipeline.

- **Key Improvement: Shape Preservation and Noise Robustness**. By projecting a clean, idealized model rather than raw pixels, V2 excels at preserving the true geometric shape of components. As shown in Figure 4.2, flat surfaces remain flat, and curved surfaces are smooth. The RANSAC and Delaunay steps make the pipeline exceptionally robust to the noisy depth data that plagued earlier versions.
- **Remaining Limitation: Inter-Object Occlusion**. Despite its advancements, the V2 pipeline inherits a critical limitation from V1: it still fails to handle occlusions by other, separate objects in the target scene. Its Z-buffer operates only on the geometry of the component being transferred and has no knowledge of the wider scene context in the target view. Therefore, it will still incorrectly project a mask on top of a nearer, occluding object.

The successful resolution of shape fidelity and noise in V2 was a major milestone. However, the persistent problem of inter-object occlusion demonstrated the need for a final, crucial enhancement: incorporating depth and mask information from the target view itself.

4.4 V3: Full Occlusion Handling via Target Scene Analysis

The V3 pipeline represents the culmination of the single-source transfer methodology, designed specifically to solve the critical problem of inter-object occlusion that limited the V1 and V2 pipelines. While previous versions could handle self-occlusion, they remained blind to the overall geometry of the target scene. The core innovation of V3 is the introduction of a final, decisive visibility check. This is achieved

by leveraging the depth map and vehicle mask of the *target* image to ensure that a projected pixel is only rendered if it is physically visible and correctly located on the vehicle's surface.

4.4.1 Methodology and Implementation

The V3 pipeline builds directly upon the geometric fitting framework established in V2. The initial steps of depth map in-painting, 3D point cloud generation, and geometric model fitting (detailed in Section 4.3) are performed identically. The key advancements are introduced in the final projection and masking stages.

1. **Initial Steps (Inherited from V2)**: The pipeline begins by performing the full V2 methodology up to the point of generating a clean, idealized set of 3D points (`points_to_project`) that represents the source component's geometry.

2. **Visibility-Checked Projection**: This step replaces the simple Z-buffered projection of previous versions with a more sophisticated scene-aware analysis. For each 3D point p_{3d} from the idealized model, its visibility in the target scene is determined.

   - The point is first projected to its 2D coordinate (u', v') in the target image. Simultaneously, its depth with respect to the target camera, z_{proj}, is calculated.
   - The ground-truth depth of the target scene at that exact coordinate, z_{gt}, is then retrieved from the target view's high-fidelity depth map.
   - The point is considered visible only if its projected depth is less than or equal to the ground-truth depth of the scene, plus a small tolerance τ_{depth} to account for minor sensor noise and precision differences. The mathematical condition for rendering a pixel is:

     $$z_{proj} \leq (z_{gt} + \tau_{depth}) \tag{4.1}$$

     If this condition is not met, the point is determined to be occluded by a nearer object in the target scene and is discarded.

   This check is the crucial step that prevents masks from appearing on the wrong side of the car or being projected through foreground objects.

3. **Final Masking and Clipping**: The set of pixels that pass the visibility check are rendered onto a raw mask. This mask is then consolidated using the same morphological closing operation from previous versions. As a final step in ensuring physical correctness, the resulting solid mask is clipped using a bitwise AND operation with the target image's overall vehicle mask. This removes any minor projection errors that may have landed just outside the vehicle's silhouette.

4.4.2 Results and Discussion

The V3 pipeline produces the most physically accurate and visually correct masks of any single-source method developed. By incorporating information from the target scene, it successfully resolves the inter-object occlusion problem, which was the primary failure mode of V2.

Figure 4.3 provides a case study of the V3 pipeline's performance, which can be directly compared to the V2 result in the previous section (Figure 4.2). The most notable improvement is the elimination of erroneous projection noise. The small green flecks that were visible on the bumper and wheel arch in the V2 result have been successfully filtered out in the V3 output. This is a direct result of the final

Figure 4.3: A representative result from the V3 pipeline for the ‘’bonnet‘‘ component. The final mask (b) is cleaner than the V2 equivalent, with projection noise outside the main component area removed. This demonstrates the effectiveness of the final visibility check and clipping stages. (Source image: Generated by `masktransfer_finalVer1.ipynb` Github: 3D-mask-transfer/Source images/Component transfer output)

clipping step, which performs a bitwise AND operation with the target vehicle's ground-truth silhouette mask, ensuring the final output is strictly confined to the car's body.

Interestingly, this qualitative improvement in cleanliness is not reflected by a major increase in the quantitative score. The V3 result achieves an Intersection over Union (IoU) of 0.4781, a slight decrease from the 0.4870 achieved by the V2 pipeline on the same component. This highlights a key aspect of the evaluation:

- The primary factor depressing the IoU score in both V2 and V3 is the large-scale incomplete coverage caused by the streaky artifacts from the geometric fitting stage (a flaw inherited from V2).
- The noise removed by V3 constitutes a very small pixel area. Removing these few "false positive" pixels provides a cleaner, more precise result, but it does not fix the much larger "false negative" area of the missing streaks.

Therefore, the true benefit of V3 is not always a higher IoU, but a significant increase in the physical plausibility and reliability of the output. The resulting mask is more trustworthy, even if the raw score is comparable to V2.

The primary trade-off for this increased accuracy is a significant increase in data dependency. The success of the V3 method is critically contingent on the availability and pixel-perfect accuracy of the depth map and vehicle mask for the target view. This makes the method powerful but less flexible, as it requires high-quality ground-truth data for both the source and target scenes.

4.5 V3-Fast: Optimization via Point Cloud Subsampling

4.5.1 Motivation and Principle

While the V3 pipeline achieves a high degree of physical accuracy, its methodology is computationally expensive. Processing every pixel for large components creates a bottleneck for large-scale applications. The V3-Fast method was developed to address this efficiency problem.

The core principle is that the surface of a vehicle component is often geometrically redundant and can be accurately reconstructed from a fraction of its total surface points. V3-Fast exploits this by introducing a **2D subsampling** step at the beginning of the workflow. The hypothesis is that a significant speedup can be achieved with only a minimal, acceptable loss in accuracy.

4.5.2 Implementation and Mathematical Formulation

The V3-Fast pipeline is a direct modification of the V3 method, introducing a `subsample_step` parameter, n. Instead of processing the full set of source mask pixels, $\mathcal{P}_{src}$, it operates on a smaller, subsampled set, $\mathcal{P}_{sub}$. A pixel (u, v) from the original mask is selected if its coordinates are integer multiples of the step n. This selection criterion is expressed as:

$$\mathcal{P}_{sub} = \{(u, v) \in \mathcal{P}_{src} \mid u \pmod n = 0 \land v \pmod n = 0\}$$

This operation reduces the number of points to be processed by a factor of approximately n^2. The relationship between the number of points before ($|\mathcal{P}_{src}|$) and after ($|\mathcal{P}_{sub}|$) subsampling is:

$$|\mathcal{P}_{sub}| \approx \frac{|\mathcal{P}_{src}|}{n^2}$$

This quadratic reduction in the point cloud size is the source of the dramatic performance gain, as all subsequent computationally intensive steps operate on this much smaller set of points.

4.5.3 Performance Analysis: The Speed vs. Accuracy Trade-off

To determine the optimal subsample step, an experiment was conducted on the seven target components with n values from 3 to 20. The average Intersection over Union (IoU) and processing time were recorded for each configuration.

The results, summarized for the ''bonnet'' component in Table 4.1, reveal a clear trade-off. For n values from 3 to 9, processing time decreases exponentially while the IoU score degrades gracefully. Beyond $n = 9$, the IoU drops sharply as the point cloud becomes too sparse for the geometric fitting algorithms to function reliably.

Table 4.1: Performance of V3-Fast on the ''bonnet'' component with varying subsample steps (n).

Subsample Step ($n \times n$)	**Average IoU**	**Processing Time (s)**
3x3	0.7450	271.81
5x5	0.7399	89.47
7x7	0.7163	44.02
9x9	**0.7026**	**28.42**
10x10	0.5921	23.97
20x20	0.3386	9.96

In Figure4.4. The data from the experiment reveals a classic speed-versus-accuracy trade-off:

- **Processing Time**: The standard V3 pipeline required **10.93 seconds** to complete the transfer. In contrast, the V3-Fast pipeline finished in just **0.21 seconds**. This represents a massive **52-fold speedup**.
- **Intersection over Union (IoU)**: The V3 method achieved a high IoU of **0.8324**. The V3-Fast method scored a slightly lower but still excellent IoU of **0.7766**. This constitutes a relatively small **6.7% decrease** in the IoU score.

Based on this analysis, a subsample step of $\boldsymbol{n = 9}$ was selected as the optimal parameter. At this value, the pipeline achieves a theoretical speedup of up to **81×** with a relatively minor drop in average IoU, providing the best balance between high-fidelity transfer and computational efficiency.

(a) Method V3 Result

(b) Method V3-Fast Result

Figure 4.4: Direct comparison of the (a) V3 and (b) V3-Fast pipelines for the 'windshield_f' component. The images showcase the trade-off between processing time and final IoU score. (Source image: Generated by `masktransfer_finalVer1.ipynb` Github: 3D-mask-transfer/Source images/Component transfer output)

4.6 V5: Final Pipeline Tuning and Refinement

4.6.1 Motivation

While the V3-Fast pipeline is computationally efficient, its accuracy is highly sensitive to its fixed hyperparameters, particularly the depth tolerance and morphological kernel size. A strict, small depth tolerance can cause the visibility check to incorrectly discard valid pixels due to minor sensor noise, resulting in undesirable "holes" in the final mask. Conversely, a small morphological kernel may fail to bridge the larger gaps introduced by the subsampling process.

The V5 pipeline was developed to address these issues. It is not a change in the core algorithm but rather the result of a systematic tuning process to find the optimal parameters. The objective of V5 is to maximize the final accuracy and robustness of the transfer by refining the pipeline's hyperparameters and adding critical pre-processing safeguards.

4.6.2 Hyperparameter Optimization and Implementation

Three key aspects of the V3-Fast pipeline were targeted for improvement.

Optimizing Depth Tolerance (τ_{depth})

The visibility check, $z_{proj} \leq (z_{gt} + \tau_{depth})$, is critical for handling occlusions. The tolerance, τ_{depth}, determines how forgiving this check is. A small tolerance is precise but brittle, whereas a large tolerance

is more robust to noise but risks incorrectly merging objects that are at different depths.

To find the optimal value, an experiment was conducted on the ''bonnet'' component, running the pipeline with various tolerance values from 10cm to 60cm. The results are presented in Figure 4.5.

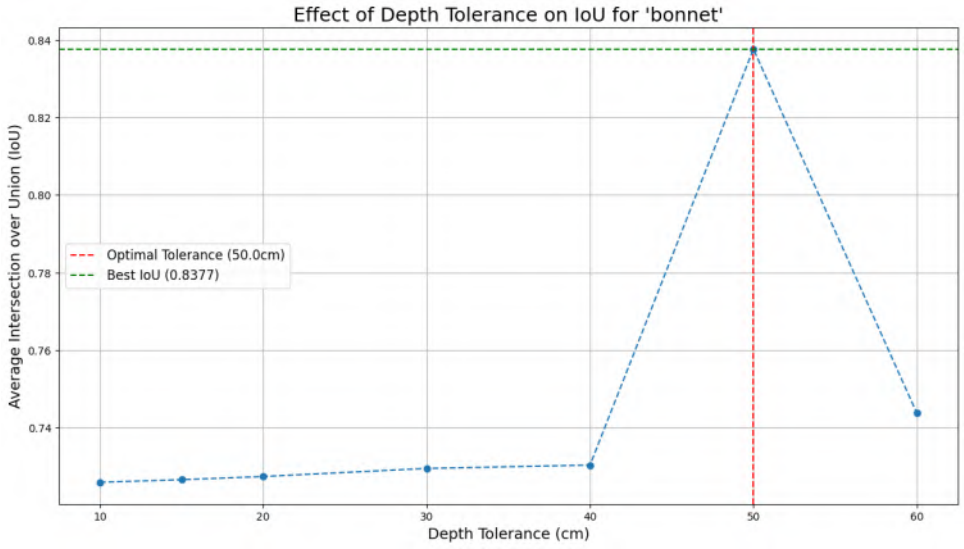

Figure 4.5: The effect of the depth tolerance parameter (τ_{depth}) on the average IoU for the ''bonnet'' component. A clear performance peak is observed at 50cm.

The analysis reveals a distinct performance peak, with the maximum average IoU of **0.8377** occurring at a depth tolerance of **50cm**. This value appears to provide the optimal balance, being large enough to overcome typical sensor noise without being so permissive as to cause false merges.

More Aggressive Gap Filling

The '9x9' subsampling introduced in V3-Fast can create larger gaps between projected points than a denser sampling would. The morphological closing kernel must be sufficiently large to bridge these gaps. Through qualitative analysis, it was determined that increasing the kernel size from '7x7' to '11x11' was more effective at producing a single, solid, and coherent final mask without introducing significant shape distortion.

Smarter Source Mask Filtering

To improve the overall reliability of the pipeline, a pre-processing safeguard was added. Before initiating the transfer, a function (`get_mask_size_threshold`) checks if the source mask is large enough to be considered valid. This step prevents the system from wasting computational resources on tiny, noisy, or insignificant source masks that are unlikely to produce a meaningful result, thereby improving the robustness of the entire process.

4.7 Summary of Methodologies

The iterative development process resulted in several distinct versions of the mask transfer pipeline, each with unique characteristics. Table 4.2 provides a concise comparison of the final five methodologies.

Table 4.2: Comparative summary of the developed mask transfer methodologies.

Method	Description	Pros	Cons
V1	Projects all pixels from the source mask, then uses a Z-buffer and morphological closing.	Simple and relatively fast. Handles self-occlusion.	Can distort shape ("blobby" results). Fails on inter-object occlusion.
V2	Fits a geometric model (plane/mesh) to the source point cloud first, then projects the clean model.	Preserves true geometric shape. Robust to depth noise.	Fails on inter-object occlusion. Can be computationally intensive.
V3	Performs V2, then adds a visibility check against the target's depth map and vehicle mask.	The most physically accurate single-source method. Handles all occlusions.	Requires the most data (target depth & mask). Can be slow.
V3-Fast	An optimized version of V3 that subsamples the source pixels before processing.	Dramatically faster than V3 with minimal loss in accuracy for a given task.	Can lose very fine, single-pixel details due to the subsampling.
V5 (Tuned)	A tuned version of V3-Fast with an optimized depth tolerance, a larger closing kernel, and a source mask size filter.	**Highest accuracy (mIoU)**. Robust to sensor noise, fixing "missing holes." The most reliable and refined pipeline.	Hyperparameters are empirically tuned for the dataset. Retains the high data dependency of V3.

4.8 Analysis of Inaccuracies and Future Improvements

Despite the success of the `v3-fast` and multi-view methods, certain inaccuracies persist. These can be attributed to two main categories: input data imperfections and algorithmic limitations.

4.8.1 Key Reasons for Inaccuracy

- **Input Data Imperfections**: The most significant issue is the **massive resolution mismatch** between the low-resolution ('256x192') depth map and the high-resolution ('1920x1440') image space, which creates "blocky" artifacts. Additionally, **depth sensor noise** on reflective surfaces causes small projection errors.
- **Algorithmic Limitations**: The method's **independent pixel projection** does not enforce surface continuity. Furthermore, the **"brute force" morphological closing** operation is a non-geometric fix that can distort the true shape of a component.

4.8.2 Recommendations for Further Improvements

The most impactful future improvement would be the acquisition of a **better depth map**.

- **Higher Resolution**: A depth map with a resolution closer to the RGB images would provide a near one-to-one correspondence between depth and colour pixels, resulting in dramatically sharper

and more accurate mask transfers.

- **Better Data Completeness**: A higher-quality 3D scanner would be more robust to challenging materials like glass and reflective paint, reducing the number of holes that require in-painting.
- **Higher Accuracy & Less Noise**: A sensor with a higher signal-to-noise ratio would provide more precise depth measurements, translating to smoother, more geometrically correct surfaces and more accurate final projections.

Chapter 5

Multi-View Aggregation for Enhanced Robustness

5.1 Introduction

While the single-source V5 pipeline (detailed in the 4.6) achieves a high degree of accuracy, its output is still fundamentally limited by the quality and perspective of the single source view chosen. A given viewpoint may suffer from partial occlusions or may not capture the full extent of a component that wraps around the vehicle's body.

To overcome these limitations, this chapter explores multi-view aggregation strategies. The core principle is that by combining information from several diverse viewpoints, the final transferred mask can be made more complete and robust. Errors or occlusions present in one source view are unlikely to be present in all others. By aggregating the results, a more accurate consensus can be formed.

This chapter introduces and evaluates two distinct strategies for selecting multiple source views:

1. A dynamic, data-driven Four Corners Method that selects views from the corners of a bounding box defined by the component's visibility.
2. A geometrically-fixed Four Angles Method that selects views from predefined angular sectors around the vehicle.

For both strategies, the individually transferred masks are combined using a pixel-wise union operation to maximize the completeness of the final result.

5.2 The Four Corners Method

5.2.1 Principle of Operation

The Four Corners method is an intelligent, data-driven approach for selecting a set of maximally diverse source views for a specific component. The key insight is that views from different "corners" relative to the vehicle will capture different aspects of a component's surface. By projecting from each vantage point and combining the results, a more comprehensive reconstruction can be achieved.

5.2.2 Implementation and Mathematical Formulation

The view selection process, implemented in the `find_corner_views` function, dynamically defines the "corners" based on the set of camera positions that can actually see the component in question. This

makes the method highly adaptive.

1. **Identify Visible Views**: For a given component, the system first retrieves the set of all available camera views where that component is visible, $\mathcal{C}_{vis}$. Each view is associated with a 3D camera position vector, $\vec{t_i} = (x_i, y_i, z_i)$.

2. **Calculate Geometric Bounding Box**: To establish a spatial frame of reference, a 2D bounding box is computed from the camera positions in the X-Z ground plane. The boundaries of this box, (x_{min}, z_{min}) and (x_{max}, z_{max}), are defined as:

$$x_{min} = \min_{\vec{t_i} \in \mathcal{C}_{vis}} (x_i) \quad ; \quad x_{max} = \max_{\vec{t_i} \in \mathcal{C}_{vis}} (x_i)$$
$$z_{min} = \min_{\vec{t_i} \in \mathcal{C}_{vis}} (z_i) \quad ; \quad z_{max} = \max_{\vec{t_i} \in \mathcal{C}_{vis}} (z_i)$$

3. **Define Quadrants via Center Point**: The center of this bounding box, $\vec{p_c} = (x_c, z_c)$, is calculated to serve as the origin for the four quadrants:

$$x_c = \frac{x_{min} + x_{max}}{2} \quad ; \quad z_c = \frac{z_{min} + z_{max}}{2}$$

   A camera view $\vec{t_i} = (x_i, y_i, z_i)$ is assigned to one of four quadrants based on its position relative to this center point. Assuming a coordinate system where +Z is forward, the quadrant sets are defined by the following conditions:

$$\mathcal{Q}_{FL} = \{\vec{t_i} \in \mathcal{C}_{vis} \mid (x_i < x_c) \wedge (z_i > z_c)\} \quad \text{(Front-Left)}$$
$$\mathcal{Q}_{FR} = \{\vec{t_i} \in \mathcal{C}_{vis} \mid (x_i > x_c) \wedge (z_i > z_c)\} \quad \text{(Front-Right)}$$
$$\mathcal{Q}_{RL} = \{\vec{t_i} \in \mathcal{C}_{vis} \mid (x_i < x_c) \wedge (z_i < z_c)\} \quad \text{(Rear-Left)}$$
$$\mathcal{Q}_{RR} = \{\vec{t_i} \in \mathcal{C}_{vis} \mid (x_i > x_c) \wedge (z_i < z_c)\} \quad \text{(Rear-Right)}$$

4. **Select Corner Views**: The algorithm iterates through each of the four quadrant sets and selects one representative camera view from each, resulting in a list of up to four unique image IDs that provide diverse perspectives of the component.

5.2.3 Mask Aggregation via Union

Once the four corner source views are selected, the V5 mask transfer pipeline is executed independently for each one, projecting its version of the component mask onto the same single target image. These individual masks are then aggregated into a single, final mask using a pixel-wise union (logical OR) operation. This strategy maximizes the **completeness** of the final mask, ensuring that if any part of the component is visible from at least one of the corner views, it will be included in the final output.

5.2.4 Qualitative Results and Analysis

The effectiveness of the Four Corners method is best illustrated through a step-by-step visualization of the entire process, from source selection to final mask aggregation. Figure 5.1 presents a complete analysis for transferring the ''bonnet'' component to a target image.

The analysis of the process, as depicted in the figure, is as follows:

- **Source View Selection**: The top row displays the four source images chosen by the adaptive quadrant algorithm. It successfully selected a set of diverse viewpoints (front-left, front, front-right, and a high-angle front-right) that each provide a unique perspective on the bonnet.

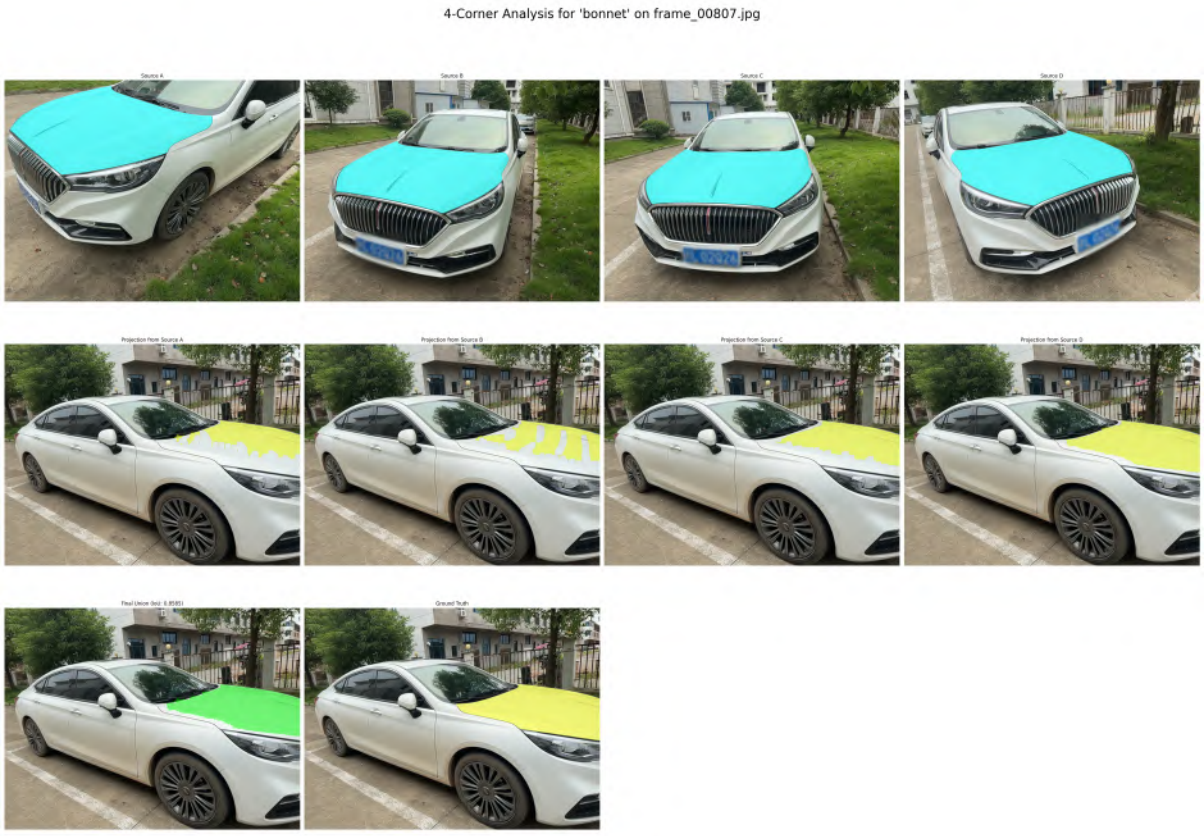

Figure 5.1: A complete visualization of the 4-Corner analysis pipeline. Top Row: The four diverse source views selected by the algorithm. Middle Row: The incomplete results from projecting each source view individually. Bottom Row: The final, aggregated union mask (green) shown next to the ground-truth mask (yellow). (Source image: Generated by `masktransfer_finalVer1.ipynb` Github: 3D-mask-transfer/Source images/4 corner and 4 angles output)

- **Individual Projections**: The middle row demonstrates the inherent limitation of any single-source transfer. Each projection, while geometrically plausible from its own viewpoint, only captures a fraction of the component's full surface. For example, the projection from Source A only covers the left side of the bonnet, while the projection from Source D only covers the right. This partial visibility is the primary problem that multi-view aggregation is designed to solve.
- **Mask Aggregation via Union**: The bottom row reveals the power of the aggregation strategy. The final union mask (shown in green) is created by combining the four incomplete projections. The result is a single mask that is far more **complete** and accurate than any of its individual inputs. It successfully reconstructs almost the entire surface of the bonnet by integrating the partial data from all four viewpoints.

This example provides a strong qualitative validation for the Four Corners method. It proves that aggregating projections from multiple, diverse views using a union operation is a highly effective strategy for overcoming the partial visibility issues that limit single-source pipelines, thereby maximizing the completeness of the final transferred mask.

5.3 Final System Performance and Results

This section details the performance of the final, optimized mask transfer system. The ultimate goal is to combine the most effective multi-view selection strategy with the most accurate single-source projection pipeline. The results confirm that the 4-Corners method, when powered by the highly-tuned V5 projection engine, yields the most accurate and complete results of this study.

5.3.1 Establishing the Optimal Projection Engine: V5

Before evaluating the multi-view strategy, it was necessary to finalize the single-source projection engine. As detailed in the previous chapter, the V5 pipeline improves upon the V3-Fast baseline through a process of careful hyperparameter tuning.

The combination of a more forgiving depth tolerance, a more aggressive closing kernel, and an intelligent source filter results in the final V5 pipeline. As shown in Table 5.1, these refinements lead to a dramatic improvement in the mean Intersection over Union (mIoU) for key target components when compared to the V3-Fast baseline.

Table 5.1: Mean IoU comparison between the V3-Fast baseline and the tuned V5 pipeline.

Component	**V3-Fast mIoU**	**V5 mIoU**
Bonnet	0.6252	**0.8601**
Bumper (Front)	0.6370	**0.8069**

For the ''bonnet'', the mIoU increased from 0.6252 to 0.8601, a 37.6% relative improvement. For the ''bumper'', the mIoU increased from 0.6370 to 0.8069, a 26.7% relative improvement. These results confirm that the V5 pipeline successfully addresses the remaining sources of error from the V3-Fast method, yielding the most accurate and robust single-source pipeline of this study.

5.3.2 Aggregated Performance with the 4-Corners Method

The 4-Corners method leverages the power of this newly optimized V5 pipeline. By projecting from up to four diverse viewpoints and aggregating the results with a union operation, it directly addresses the primary remaining limitation of any single-source method: incomplete coverage due to partial occlusion or extreme viewing angles.

While the V5 pipeline produces a highly accurate mask from its own perspective, the 4-Corners method ensures that the final mask is also highly complete, integrating information from all sides of the component. The final system therefore represents a two-fold solution: the V5 pipeline provides the per-view accuracy through hyperparameter tuning, while the 4-Corners strategy provides overall completeness through multi-view aggregation. This combined approach yields the most robust and reliable mask transfer results of this study.

5.3.3 Practical Limitations

Despite its effectiveness, the Four Corners method has a significant practical limitation: it is entirely dependent on having a rich set of pre-existing annotations. The algorithm begins by identifying views where the component is already known and masked. In a real-world production scenario, requiring a component to be manually annotated from four diverse angles before it can be automatically transferred is impractical and defeats the purpose of automation.

This limitation motivates the exploration of an alternative view selection strategy that does not rely on pre-existing component visibility. The Four Angles method, discussed in the next section, addresses this by selecting views based on their fixed geometric position relative to the car.

5.4 The Four Angles Method: A Manual Selection Approach

5.4.1 Principle and Motivation

The Four Angles method provides an alternative multi-view aggregation strategy that addresses the practical limitations of the Four Corners method. Instead of algorithmically searching for source views based on pre-existing annotations, this approach relies on a fixed, manually selected set of four "golden" source images intended to represent optimal, cardinal-like views of the vehicle (e.g., front, back, and sides).

The motivation is two-fold. First, it bypasses the need for rich, pre-existing annotations, making it possible to initiate a transfer for any component. Second, it serves as a controlled experiment to evaluate the pipeline's performance when provided with a consistent, high-quality set of source viewpoints, removing the variable of the selection algorithm itself. While faster and less computationally demanding at the selection stage, the overall accuracy of this method is highly dependent on the quality of the initial manual selection.

5.4.2 Experimental Procedure

A large-scale experiment was conducted to evaluate the performance of this method across two datasets. The procedure was as follows:

1. **Manual Source Selection**: For each dataset, a fixed set of four source images was manually chosen.

2. **Target Iteration**: For a given component (e.g., ''bonnet''), the system identified all possible target images within the dataset that contained a ground-truth mask for that part.

3. **Transfer and Aggregation**: The system looped through every target image. In each loop, the V5 pipeline projected the component mask from each of the four manually-selected source images onto the current target. The four resulting masks were then combined into a single mask using a union operation.

4. **Evaluation**: Finally, this aggregated union mask was compared to the target's ground-truth mask to calculate an IoU score. The average of these scores across all targets yielded the final mIoU for that component.

5.4.3 Quantitative Results and Analysis

The performance of the Four Angles method was quantitatively evaluated across two datasets, with a specific focus on the effect of the `depth_tolerance` hyperparameter on Dataset 2. The results, summarized in Table 5.2, reveal two primary findings.

First, a significant performance disparity is evident between Dataset 1 and Dataset 2, most notably for the ''windshield_f'' component (0.0146 vs. 0.4600). This highlights the method's high sensitivity to the quality of the manually selected source images and the underlying dataset characteristics.

Second, the analysis on Dataset 2 shows that a `depth_tolerance` of **0.5 (50cm)** provides the optimal balance of completeness and precision. While the overall average mIoU improves marginally at a tolerance

of 0.7, the performance on critical components such as ''bumper_f/cover'' and ''headlamp_l_assy'' clearly peaked at 0.5. This justifies the selection of 0.5 as the optimal tolerance for this method, as it maximizes performance on key components without being overly permissive.

Table 5.2: Mean IoU results for the Four Angles method. The table compares performance on two datasets and analyzes the effect of varying the depth tolerance (τ_{depth}) on Dataset 2. The highest score for each component on Dataset 2 is bolded.

Component	**Dataset 1**	**Dataset 2 mIoU at Tolerance (τ_{depth})**			**Trend on**
	(τ_{depth}=0.2)	0.2	0.5	0.7	**Dataset 2**
bonnet	0.6574	0.6204	0.6400	**0.6485**	Improving
bumper_f/cover	0.5908	0.6792	**0.6933**	0.6808	Peaked at 0.5
windshield_f	0.0146	0.4600	0.5893	**0.6414**	Improving
headlamp_l_assy	0.2699	0.5828	**0.6332**	0.6316	Peaked at 0.5
mirror_l_assy	0.3114	0.0677	0.0809	**0.0853**	Improving
grille	0.7005	0.3729	**0.3750**	0.3738	Stable
door_fl_assy	0.4011	0.3037	0.4118	**0.4209**	Improving
Overall Average	**0.4208**	**0.4410**	**0.4891**	**0.4975**	**Improving**

In conclusion, the Four Angles method serves as a fast and computationally inexpensive alternative for multi-view transfer, but its reliance on a fixed, manually-curated set of source images makes its performance less consistent and ultimately less accurate than the adaptive Four Corners approach.

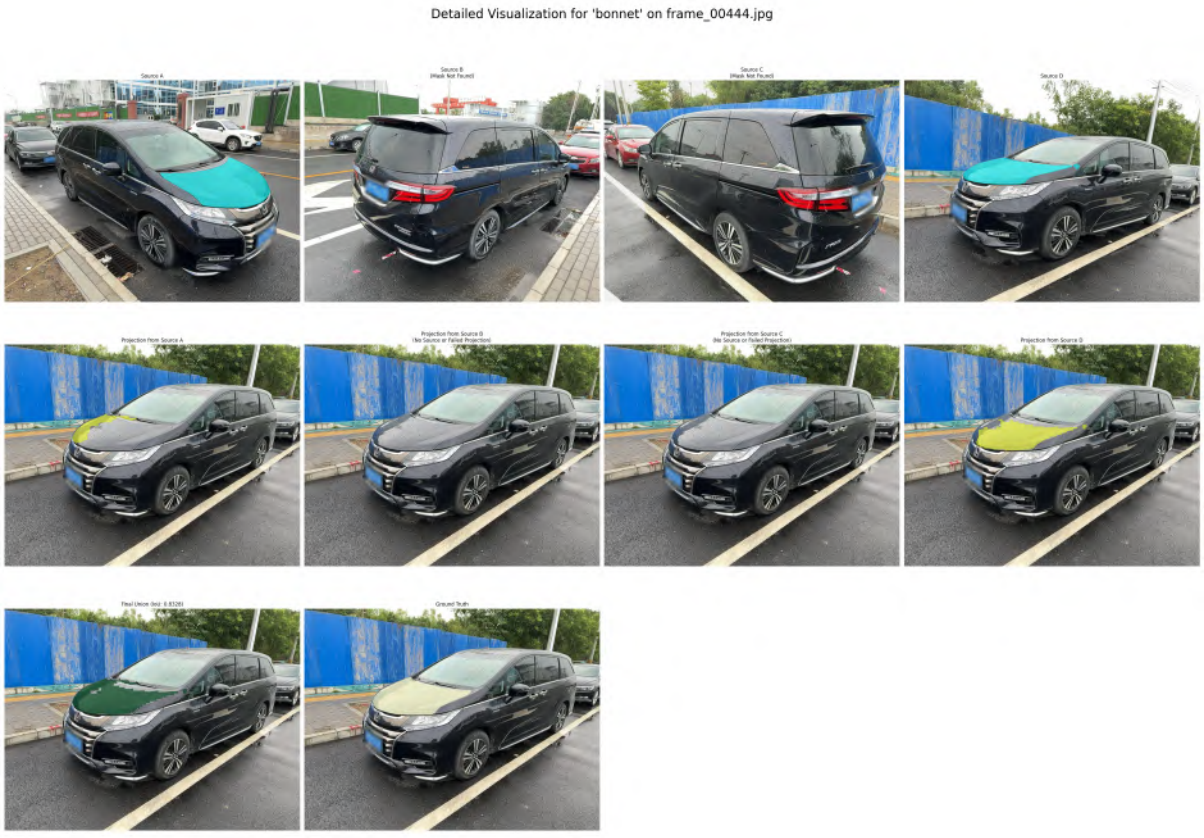

Figure 5.2: An example of a successful mask transfer for the ''bonnet'' component on Dataset 2 using the Four Angles method with the optimized 0.5 depth tolerance. (Source image: Generated by `masktransfer_finalVer2.ipynb` Github: 3D-mask-transfer/Source images/Component transfer output)

Chapter 6

Conclusion

This project set out to address the significant challenge of manual data annotation in the automotive AI sector. By leveraging 3D geometry, we have successfully designed, implemented, and validated a comprehensive pipeline for the automated transfer of vehicle component masks between different camera views.

Through an iterative development process, the methodology evolved from a naive baseline to a sophisticated, multi-stage system. Key advancements included the integration of robust geometric fitting to preserve component shape, a full visibility check using target scene depth to resolve complex occlusions, and performance optimization via point cloud subsampling. The final V5 pipeline, refined through careful hyperparameter tuning, was established as the most accurate and reliable single-source transfer engine.

Furthermore, we demonstrated that the limitations of a single viewpoint can be effectively overcome by aggregating information from multiple perspectives. Two distinct multi-view strategies were explored: a manually-curated Four Angles approach and an adaptive, data-driven Four Corners method. While the Four Angles method served as a fast and computationally inexpensive alternative, the Four Corners method, powered by the V5 engine, ultimately proved to be the more effective and robust strategy for maximizing the completeness and accuracy of the final transferred mask.

In conclusion, this work has produced a powerful and efficient tool that successfully automates a critical part of the data annotation workflow. The final system stands as a robust proof-of-concept, with the potential to significantly accelerate the development of AI-powered damage assessment solutions in the automotive industry.

Bibliography

[1] Xiaobai Du, *3DRealCar*. https://xiaobiaodu.github.io/3drealcar/

[2] COLMAP https://colmap.github.io/tutorial.html

[3] 3Drealcar Dataset https://studentutsedu-my.sharepoint.com/personal/xiaobiao_du_student_uts_edu_au/_layouts/15/onedrive.aspx?id=%2Fpersonal%2Fxiaobiao%5Fdu%5Fstudent%5Futs%5Fedu%5Fau%2FDocuments%2Fdataset%2F3DRealCar&ga=1

[4] cv2.inpaint https://docs.opencv.org/3.4/df/d3d/tutorial_py_inpainting.html