Generating Spatial Images using Monocular Depth Estimation

Karan Soin
Stanford University
Computer Science Department
ksoin@stanford.edu

Nick Riedman
Stanford University
Computer Science Department
nriedman@stanford.edu

Abstract

Recent advancements in spatial imaging, led by major technology companies, have introduced innovative ways to capture and view immersive 3D content. However, these experiences leave a lot to be desired in terms of immersion and true 3D effects. Moreover, they also require specialized hardware, limiting their accessibility to a wider audience. To address these limitations, we present an approach that enhances the immersiveness and engagement of 2D images when viewed through Virtual Reality (VR) or Mixed Reality (MR) devices. Our method combines latent diffusion-based monocular depth estimation and 3D stereo image techniques to generate spatial images from standard 2D photos. We train a latent diffusion model on the VirtualKitti 2 synthetic depth map dataset to estimate accurate depth maps from RGB images, using a U-Net architecture and a Variational Autoencoder. The resulting depth maps are then used to create stereo image pairs with enhanced depth perception, incorporating the Parallax Effect, Bokeh Effect, and a nearest neighbor hole-filling algorithm. The generated stereo images can be viewed side-by-side or integrated into VR/AR systems, providing users with an immersive 3D experience using standard 2D photos. Our evaluation and experimental results demonstrate the effectiveness of the proposed approach in creating a convincing sense of depth and improving the immersion in 2D images.

1. Introduction

In recent years, there has been a growing interest in transitioning from traditional 2D images to more immersive and engaging content. Traditional 2D images, once the dominant form of visual representation, are now being transformed into immersive experiences that offer a greater sense of depth, presence, and engagement, to make you feel like you were there. The introduction of new headsets and displays has opened up new possibilities for creating these experiences. Companies like Apple and Meta have been at the forefront of this development, releasing spatial image and video experiences that aim to provide users with a sense of depth and immersion [1]. However, despite the advancements made in this field, current spatial experiences still seem to fall short in terms of depth perception and overall immersion. When viewed through headsets or AR/MR devices, these experiences can still feel flat and lacking in depth, diminishing the sense of presence and engagement for the user. Moreover, the accessibility of these technologies remains a challenge, as they often require specialized cameras, headsets, or the most advanced smartphone versions to capture these spatial images, limiting their reach to a wider audience [1].

In this paper, we present an approach that addresses the problem of enhancing the accessibility and immersiveness and engagement of 2D images when viewed through headsets or AR/MR devices. Our proposed framework combines state-of-the-art depth estimation techniques using latent diffusion models with stereo image generation methods to create a more immersive 3D experience from readily available 2D images. The main idea behind our approach is to generate accurate depth maps from 2D images using Latent Diffusion Models (LDMs), which have shown excellent performance in capturing the three-dimensional structure of scenes [18, 37, 34]. The input to our algorithm is a standard 2D RGB image, which can be easily captured using common cameras or smartphones. We first encode the input image into a latent space representation using a Variational Autoencoder (VAE). The VAE learns a compact and meaningful representation of the image in the latent space, which becomes the input to the next steps in our methodology [17].

Next, we use an LDM to estimate an accurate depth map of the input image from its latent space representation. The LDM consists of a U-Net architecture that learns to map the latent space representation to the corresponding depth map [37, 31]. We trained the U-Net on the **Virtual KITTI 2 Dataset**, a virtual image and depth dataset which consists of 40,000 RGB-D images, to learn the mapping between an input RGB image and its corresponding depth map [4]. During inference, we use the trained LDM to estimate the depth map of the input image by sampling from the latent

space and passing it through the U-Net. This gives us a final generated depth map for any given 2D image.

The estimated depth map, along with the original RGB image, is then processed as the input to our stereo image generation pipeline, which outputs a stereo image pair (left and right perspectives) that simulates the parallax effect observed in human binocular vision. We incorporate depth-aware image processing techniques such as the Bokeh Effect [8], Parallax Shifting [28, 44], and image hole-filling based on nearest-neighbor non-zero pixels [13] to enhance the perception of depth and create a more complete representation of the left and right eye perspectives of the original 2D image. The generated stereo images can be viewed side by side or integrated into a VR/MR headset for an immersive 3D representation, with 3D foreground features popping out at the user.

2. Related Work

2.1. Monocular Depth Estimation

Monocular depth estimation is a classic problem in computer vision. Utilizing convolutional neural networks (CNNs) for depth estimation was pioneered by Eigen et al. [10] in 2014, and since then several milestones have advanced the state of the art. This includes ordinal regression [11], planar guidance [21], neural fields and deeper CNNs [23],

Monocular depth estimation from a single image is largely treated as a per-pixel regression task. Some works, like **AdaBins**, employ a mixed classification-regression approach that discretizes the depth values into bins and assigns each pixel to a bin [2]. However, many models take as input an RGB image and, for each pixel, output a corresponding depth $\mathbf{d} \in \mathbb{R}$ [42, 10, 3]. In general, these models can be classified into two categories depending on the output: metric and relative depth.

Relative depth estimation refers to models that produce a depth map whose values encode pairwise relative distance to the camera [11, 5]. That is, relative depth maps capture the order of the pixels, with no meaningful interpretation of degree. This allows for high generalizability across different scenes and datasets, but at the cost of reduced applicability as it is not possible to recover accurate geometric structure.

Metric depth estimation refers to models that produce a depth map whose values are metric-scaled. The units of a metric depth map are real-world units, and can be used to locate the pixel in 3D space. This type of estimation has many important applications in autonomous driving [42], robotics [43], 3D object detection [41], and more. While these networks are immediately useful in their respective domain, they are often trained over a single dataset and do not generalize well to new data.

This is especially problematic since ground-truth depth data is difficult to collect at scale, with existing datasets focusing on one type of scene. For example, the **KITTI** dataset captures RGB-D images using LiDAR, but only contains outdoor scenes of roads [12]. The **NYU V2** dataset captures data using a spatial camera, but only for indoor scenes [25]. Furthermore, these datasets are prone to noise and prominent gaps in the ground truth where the spatial cameras were unable to collect depth information.

Several works seek to address this issue. Early efforts focused on creating large, diverse, and representative datasets. This includes **DIW** (which focused on relative depth "in the wild", i.e. RGB-D images with no other metadata such as camera intrinsics) [5]. Ranftl et al. enabled mixed-data training via a training objective invariant to depth range and scale [29]. This work has given rise to a family of **MiDaS** models that aim for more generalized metric depth estimation over diverse datasets [3]. Other works, such as **MegaDepth**, focus on producing high quality, large scale, and diverse datasets leveraging augmented Internet photos [22].

Similarly, several synthetic datasets have been developed that feature computer generated RGB-D images from robust 3D models. This includes **Hypersim** (for virtual indoor settings) [30] and Virtual Kitti (a synthetic version of **KITTI**) [4]. In order to be realistic, synthetic datasets require significant effort creating hand-crafted 3D environments, but the result is an extremely high quality ground truth with none of the noise or gaps from their real-world counterparts. The quality of the ground truth, while still retaining accurate fidelity to the real world, makes synthetic datasets ideal for supervised learning. Thus, we utilize the synthetic **Virutal KITTI 2** dataset for this work.

2.2. Stereo Image Generation

For stereo rendering from a single image, there are several approaches that use an elaborate setup, specialized headsets, or specialized equipment, like the Apple Vision Pro, or the Meta Quest spatial photos [1]. Other works have focused on generating stereo image pairs from monocular inputs using deep learning techniques. Some methods employ view synthesis networks that learn to generate novel views from a single image by leveraging geometry priors like depth or multiplane images [6]. Depth maps are a key component in generating stereoscopic 3D views or stereo image pairs, especially if our starting point is just a single 2D image. Therefore, there have been several open-source depth estimation models like ZoeDepth, MiDaS, and AdelaiDepth that have been historically been used for this purpose, resulting in somewhat satisfactory results [27].

There have also been some impressive new papers using Diffusion models to create training-free stereo images, and the StereoDiffusion method proposed by Chen et al.

proposed a depth-aware image rendering pipeline that incorporates depth cues like defocus blur and parallax shifting to synthesize stereo images from a single RGB-D input. Similar depth-based rendering techniques have been applied for view synthesis, with WildFusion 3D-aware LDMs [35]. While effective, these depth-guided methods often rely on having an accurate input depth map, which can be challenging to obtain for general scenes. Recent works have explored jointly estimating depth and synthesizing novel views in an end-to-end manner using techniques like multiplane images or neural radiance fields [23, 38]. Wang, Frisvad. et al. take a different approach by directly modifying the latent space of pre-trained diffusion models like Stable Diffusion to generate stereo image pairs in an end-to-end, training-free manner [40]. Their approach uses the estimated depth map to perform stereo pixel shifting in the latent space, along with techniques like symmetric pixel shift masking and self-attention modifications to maintain consistency between the left and right views, an idea that incorporates our second step into the latent diffusion model.

3. Methods

Our pipeline for generating spatial images using monocular depth estimation is composed of two distinct steps: (1) given an RGB image, generate a depth map using our Latent Diffusion Depth Estimator; and (2) given a depth map and its corresponding image, apply the Bokeh (Depth of Field) effect, generate a stereo pair of images using parallax, and hole filling algorithms. The spatial image effect then comes from displaying the stereo pair in a VR headset.

3.1. Depth Map Generation

3.1.1 Latent Diffusion Generation

Taking inspiration from Denoising Diffusion Probabilistic Models [15], we pose the task of depth estimation as modeling the conditional distribution $\mathcal{D}(\mathbf{d} \,|\, \mathbf{x})$ of $\mathbf{d} \in \mathbb{R}^{H \times W}$ (the depth) given $\mathbf{x} \in \mathbb{R}^{3 \times H \times W}$ (the RGB image).

In the forward pass, we start with $\mathbf{d}_0 = \mathbf{d}$, then sequentially add Gaussian noise at time steps $t \in \{1, 2, \dots T\}$ to get $\mathbf{d}_T$. Specifically:

$$\mathbf{d}_t = \sqrt{\alpha_t}\, \mathbf{d}_{t-1} + \sqrt{1 - \alpha_t}\, \boldsymbol{\epsilon}_t \tag{1}$$

where $\alpha_t = \prod_{s=1}^{t}(1 - \beta_s)$, $\{\beta_1, \beta_2, \dots \beta_T\}$ determines the noise schedule, and $\boldsymbol{\epsilon}_t \sim \mathcal{N}(0, I)$ is the noise applied at time t. After T time steps, the result $\mathbf{d}_T$ is akin to a random point outside the distribution $\mathcal{D}$. Then, in the reverse process, our parameterized denoising model $\boldsymbol{\epsilon}_\theta(\cdot)$ with parameters θ takes as input a noisy depth at an arbitrary time step $\mathbf{d}_{t'}$ and removes a step's worth of noise to produce $\mathbf{d}_{t'-1}$. Thus, given condition $\mathbf{x}$, the reverse process starts with $\mathbf{d}_T$ and iteratively removes noise using the parameterized denoising model, until after T steps we recover $\mathbf{d}_0 = \mathbf{d}$.

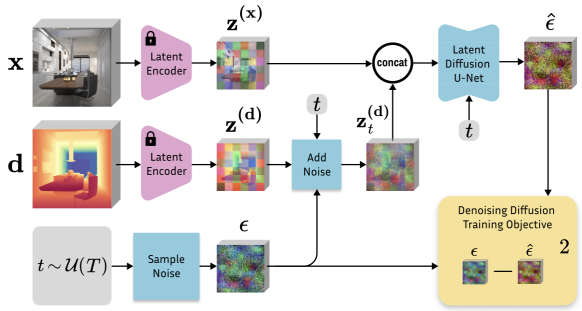

Figure 1. **The training process for one batch.** Both images and depth maps are embedded. Noise is added to the depth for time step random t according to the DDPM noise scheduler from [31], then the two latents are concatenated along the feature dimension and fed into the diffusion U-net. The predicted noise is evaluated using the objective function. Figure adapted with no changes from [18].

Then, at training time, we do the following. First, we take a sample from the distribution $(\mathbf{x}, \mathbf{d})$, and apply noise drawn from $\boldsymbol{\epsilon}$ at a random timestep t to get $\mathbf{d}_t$. Then, we use our parameterized model $\boldsymbol{\epsilon}_\theta$ to compute a predicted noise $\hat{\boldsymbol{\epsilon}} = \boldsymbol{\epsilon}_\theta(\mathbf{x}, \mathbf{d}_t, t)$. Finally, we update our parameters θ by minimizing the objective loss function [15]:

$$\mathcal{L} = \mathbb{E}_{\mathbf{d}_0, \boldsymbol{\epsilon} \sim \mathcal{N}(0,I), t \sim \mathcal{U}(0,T)} ||\boldsymbol{\epsilon} - \hat{\boldsymbol{\epsilon}}||^2 \tag{2}$$

That is, the objective is to minimize the expected (mean) squared error between the predicted noise $\hat{\boldsymbol{\epsilon}}$ and the actual noise $\boldsymbol{\epsilon}$. Tuning our parameters this way allows us to use our model to de-noise a noisy depth at inference time. Specifically, at inference time, we reconstruct the $\mathbf{d} = \mathbf{d}_0$ by starting from a random, normally distributed $\mathbf{d}_T$, then iteratively applying our denoising model $\boldsymbol{\epsilon}_\theta(\mathbf{x}, \mathbf{d}_t, t)$.

Up till now, we have been working directly with the depth/RGB data $\mathbf{d}$ and $\mathbf{x}$. However, in Latent Diffusion, we perform the denoising training and inference in a low-dimension latent space [31]. This increases computational efficiency by compressing our data while preserving perceptually important features [31]. In our implementation, similarly to Marigold and Stable Diffusion [18, 31], the latent space is the bottle-neck of a Variational Auto-Encoder (VAE) that is trained beforehand. Like Marigold, we use the pre-trained VAE from Stable Diffusion V2 [18, 31] [1].

Now, before we train the denoising diffuser, we embed our data into the latent space. We use the encoder of the VAE $\mathcal{E}$ to encode the depth and image, where the encodings are $\mathbf{z}^{(\mathbf{x})} = \mathcal{E}(\mathbf{x})$ and $\mathbf{z}^{(\mathbf{d})} = \mathcal{E}(\mathbf{d})$. Then, instead of training our parameterized model with the data itself, we train with the latent encodings: $\boldsymbol{\epsilon}_\theta(\mathbf{z}^{(\mathbf{x})}, \mathbf{z}_t^{(\mathbf{d})}, t)$. At inference time, once we reconstruct $\mathbf{z}_0^{(\mathbf{d})}$, we recover the depth

[1] The pre-trained VAE we used can be found at: https://huggingface.co/stabilityai/stable-diffusion-2

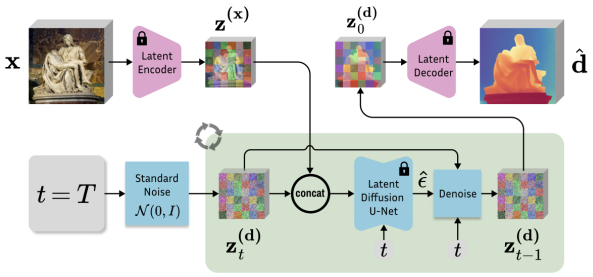

Figure 2. **The inference process.** The input image is encoded into the latent space. $\mathbf{z}_T^{(\mathbf{d})}$ is drawn from a Normal distribution, and is concatenated with the latent image along the feature dimension and fed into the denoising U-net. The predicted noise is used to iteratively de-noise the depth latent. Once $\mathbf{z}_0^{(\mathbf{d})}$ is reconstructed, it is decoded, resulting in the generated depth map. Figure adapted with no changes from [18].

map in the original space with the VAE's encoder D. That is, $\mathbf{d} = \mathrm{D}(\mathbf{z}_0^{(\mathbf{d})})$. See Figure 1 for a visual summary of the training process, and Figure 2 for a visual summary of the inference process.

3.1.2 Network Architecture, Training, and Inference

For the parameterized denoising model $\boldsymbol{\epsilon}_\theta$, the backbone of our architecture, we utilize a modified U-net from [31]. Specifically, this is the time-conditional U-net architecture from the Stable Diffusion V2 LDM (see Figure 3 in [31] for details), which is trained on the Internet-scale dataset LAION-5B [33]. We condition the architecture by concatenating the inputs, and use self-attention in the cross attention layers. As in Marigold, to accommodate both the depth and image latents, we modified the first layer to have twice the number of channels [18]. The number of output channels remains the same.

Finally, our training and inference procedures are adapted from the simple and efficient fine-tuning and inference procedures introduced in [18]. See Figures 1 and 2 for details.

3.1.3 Depth Normalization

Before training, we apply an affine-invariant depth normalization to the depth maps as in [18]. This ensures that the depth maps have values mostly in the range of $[-1, 1]$, which is what the inputs to the VAE expect. Additionally, this enforces a standard scale-invariant depth representation, which may simplify training [18]. Specifically, for each depth map in the ground truth $\mathbf{d}$, we compute the 2^{nd} and 98^{th} percentile depth values **d2** and **d98**, then apply the following linear transformation before encoding:

$$\tilde{\mathbf{d}} = \left(\frac{\mathbf{d} - \mathbf{d2}}{\mathbf{d98} - \mathbf{d2}} - 0.5\right) \times 2 \tag{3}$$

We use the 2^{nd} and 98^{th} percentiles to protect against outliers, and proceed with the training procedure by then encoding $\tilde{\mathbf{d}}$ into the latent space.

3.2. Stereo Image Pair Generation

Stereo image generation is a technique used to create a pair of images that simulates the left and right views of a scene, providing a sense of depth and immersion when viewed through a stereoscopic display or VR headset. We first apply the Bokeh effect, and once that Depth of Field blurring has been applied, the stereo image generation algorithm takes in a 2D input image, its corresponding depth map, and additional parameters to generate left and right stereo views [26, 27]. The process involves many crucial steps.

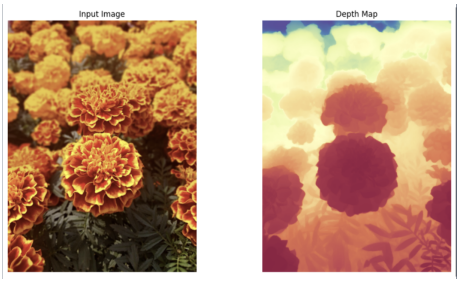

Figure 3. Sample Image with Marigold Generated Depth Map (using Marigold[18] since our compute limitations caused noisy depth maps)

3.2.1 Bokeh Effect

Given an image and a depth map generated by our previous method, we first apply the Bokeh or Depth of Field (DoF) effect to blur objects further behind in depth. In human vision, when we focus on a specific object or region, the objects closer to or farther from the focal plane appear blurred, which creates a visually appealing effect, i.e. the Bokeh Effect. [9]

In the context of this project, the bokeh effect is applied to the input image based on the depth information provided by the depth map, simulating the depth-of-field effect observed in human vision and photography. The depth map provides information about the distance of objects from the camera. Therefore, applying a Gaussian blur filter to regions of interest (ROI) around each pixel based on their depth values, the function creates a depth-dependent blurred version of the input image. Pixels closer to the focal plane (smaller depth values) will have less blur applied, while pixels farther from the focal plane (larger depth values) will have more blur applied, mimicking the behavior of camera lenses [9, 24].

The amount of blur is controlled by the bokeh intensity parameter, which determines the strength or amount of blur

to be applied. The kernel size for the Gaussian blur is calculated using the following formula:

$$\text{kernel\_size} = \text{depth\_value} \times \text{bokeh\_intensity} \quad (4)$$

3.2.2 Depth Map Normalization

The algorithm begins by normalizing the depth map to a range of [0, 1], where 0 represents the farthest depth and 1 represents the closest depth. This normalization step ensures consistent depth values across different images and facilitates the subsequent pixel shifting operations [27].

3.2.3 Parallax Shift

The parallax shift is a key component of the stereo image generation process. It simulates the binocular disparity observed in our human vision, where the left and right eyes perceive slightly different views of the same scene due to their horizontal separation. The algorithm calculates the parallax shift for each pixel based on its depth value and the interpupillary distance (IPD) we give as a user-defined parameter [16, 14]. The formula used for calculating the parallax shift in pixels is:

$$\text{parallax\_shift\_px} = \text{int}\left(\frac{\text{ipd\_px}}{2} \times (1 - \text{depth})\right) \quad (5)$$

Pixels with depth values closer to 1 (closer to the camera) will have a smaller parallax shift, while pixels with depth values closer to 0 (farther from the camera) will have a larger parallax shift [16].

3.2.4 Pixel Shifting Algorithm

The pixel shifting process is a direct result of applying the parallax shift to the input image. It creates the left and right stereo views by horizontally displacing pixels based on their depth values and the calculated parallax shift [39, 14]. For each pixel in the input image, the algorithm does the following:
(1) Calculate the parallax shift (parallax_shift_px) using the formula mentioned above,
(2) Determine the new column positions for the pixel in the left and right stereo views.
For the right view: col_r = col − parallax_shift_px,
and for the left view: col_l = col + parallax_shift_px By shifting the pixels horizontally based on their initial parallax, the algorithm creates two slightly different perspectives of the scene, simulating the left and right views that would be perceived by the human user's eyes.

3.2.5 Hole Filling Algorithm

After applying the pixel shifts, the algorithm performs hole filling to address any gaps or holes that may have been created during the shifting process. The hole filling step ensures a smooth and continuous appearance of the stereo images [39].

The hole filling algorithm does the following: [39, 20]:
(1) Identify the pixels that have become "holes" or gaps after the pixel shifting process, and holes are identified by 0 pixel values.
(2) For each identified hole pixel, search for the nearest non-hole pixel in a horizontal direction (left and right) within a specified range or offset.
(3) If a non-hole pixel is found within the specified range, fill the hole pixel with the value of the nearest non-hole pixel, and then we repeat the process for all neighboring holes

This hole filling algorithm effectively fills the gaps created during the pixel shifting process, ensuring a much smoother and more coherent pair of stereo images [20].

Figure 4. Final Stereo Images created after complete implementation of Bokeh, Parallax, Bokeh, and Hole Filling)

3.2.6 Unity Stereoscopic Rendering

In this part of the project, we utilized Unity to render distinct images to the left and right eyes, to finally create a stereoscopic effect essential for the 3D immersion we are looking for.

We began by creating custom layers `LeftEye` and `RightEye` to manage which objects each camera would render. Two cameras, `LeftEyeCamera` and `RightEyeCamera`, were set up within a VR rig, with their Culling Masks adjusted to render only objects assigned to their respective layers. We created quads to display the images for each eye, assigned the images to materials (`LeftEyeMaterial` and `RightEyeMaterial`), and applied these materials to the quads (`LeftEyeQuad` and `RightEyeQuad`). The quads were scaled and positioned directly in front of their respective cameras, to maintain the aspect ratio. The VR scene was then tested using a VR headset to ensure each eye correctly saw its designated image, creating the desired stereoscopic effect. We discuss results in section 5.

4. Dataset and Latent Encoding

We trained our Latent Diffusion Model (LDM) on the Virtual KITTI 2 dataset [4], a synthetic dataset specifically designed for autonomous driving research. The dataset consists of 5 sequence clones from the KITTI tracking benchmark, with each sequence providing multiple variants such as modified weather conditions (e.g., fog, rain) and camera configurations (e.g., rotated by 15 degrees). In total, there are 42,520 RGB-D image pairs, each of an outdoor scene featuring a street and in various weather conditions.

To prepare the data for training, we applied the following preprocessing transformations:

1. Resized the images and depth maps to a consistent resolution of 1216×320 pixels using bilinear interpolation [7].

2. Normalized each depth map according to the method described in 3.2.2.

As for the dataset split, we split the dataset into training, validation, and test sets using a random shuffled 80-10-10 split. The training set contains 34,016 image-depth pairs, while the validation and test sets each have 4,252 pairs. Figure 5 shows examples of RGB images from the Virtual KITTI 2 dataset along with its depth map representation.

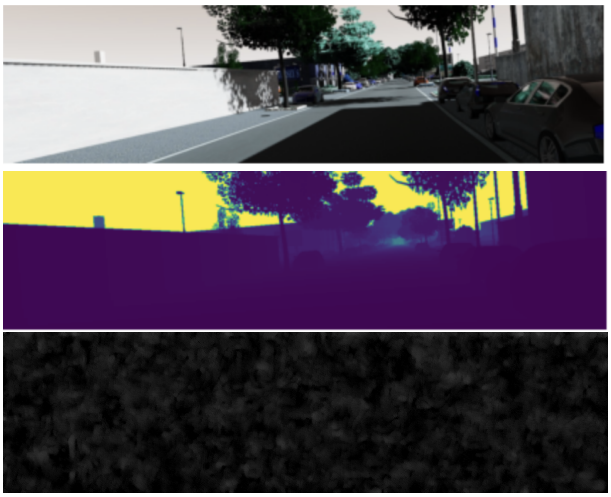

Figure 5. **Top:** An RGB image from Virtual Kitti 2. **Middle:** The corresponding ground truth depth map. **Bottom:** The generated depth map from our LDM.

We extract features from the images and depth maps by embedding them in a latent space as described in 3.1.1. Given the initial resolution of our images, this results in latents of size 155×46.

5. Results: Latent Diffusion Depth Map Generation

5.1. Implementation

We implemented the network architecture[2] described in 3.1.2 using PyTorch and Huggingface's Diffusion library. We used a learning rate of 3×10^{-5}, and a learning rate scheduler that reduced the learning rate by a factor of 10 whenever the validation loss did not reduce significantly over the course of 3 epochs. The optimizer we utilized was Adam with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. Furthermore, we applied noise using the DDPM noise scheduler from Stable Diffusion V2 [31], with 1,000 denoising steps during training and 40 denoising steps during inference. We used a batch size of 8, and trained for 2 epochs, for a total of approximately 9,000 iterations. Hyperparameter choices such as inference denoise steps and batch size were decided based on compute limitations. Initially, the learning rate was higher, but we reduced it after experiencing sudden spikes in loss.

5.2. Evaluation

We ran the inference procedure using the trained model over a random subset of 150 depth/image pairs from **Virtual Kitti 2**, generating predicted depth maps that we then used to evaluate. Specifically, we evaluated the outputs using **Absolute Relative Error**, defined as $|\frac{\hat{\mathbf{d}}-\mathbf{d}}{\mathbf{d}}|$, and $\boldsymbol{\delta}_t$. These metrics give us the following quantitative evalution:

Table 1. Comparison of Different Methods

	δ_1	δ_2	δ_3	Abs Rel Error
Marigold [18]	**0.916**	–	–	9.9
MiDaS [29]	0.630	–	–	23.9
Eigen et al. [10]	0.556	**0.752**	**0.870**	–
DiverseDepth [45]	0.631	0.694	–	19.0
Our LDM	0.632	0.640	0.642	**5.4**

We see that our model was unable to outperform the benchmark models (including the state of the art LDM-based depth estimation Marigold [18]) in any of the δ_t metrics. However, the δ_t scores are somwhat comparable to the benchmarks. For example, our LDM achieved a slightly higher δ_1 score than **DiverseDepth** [45]. Surprisingly, the absolute relative error of our LDM was lower than that of the benchmark models.

Examining 5, we seet that the generated depth maps are little more than noise. They vaguely reflect the ground truth depth maps, but overall struggle to generate an accurate depiction of the ground truth.

[2]See https://huggingface.co/docs/diffusers/en/api/models/unet2d

6. Results: Stereo Image Pair Generation

6.1. Quantitative

Using our sample image across all the evaluation metrics, we get the following in our quantitative evaluations.

6.1.1 Laplacian Variance (Blur Metric)

The Laplacian variance is computed to measure the amount of blur in the Bokeh Effect image. It provides an indication of the effectiveness of the Bokeh Effect in simulating the depth-of-field blur [39]. By applying the OpenCV Laplacian operator to the bokeh effect image and calculating the variance of the resulting Laplacian image, we were able to qualify the level of blur [32].

$$\text{Laplacian Variance Ranges} \begin{cases} 100 - 300 : \text{Low/Average Sharpness} \\ 300 - 800 : \text{Good Sharpness} \\ 1000+ : \text{Excellent Sharpness} \end{cases} \tag{6}$$

A higher Laplacian variance indicates a sharper image with less blur, while a lower variance suggests a more blurred image. In the context of our project, a lower Laplacian variance was more desirable since it indicated a stronger and more pronounced bokeh effect [39]. The obtained Laplacian variance value of **140.436** suggested a low to moderate level of blur in the bokeh effect image, indicating that the bokeh effect is successfully simulating the depth-of-field blur in the majority of the background of the image. [32].

6.1.2 Peak Signal-to-Noise Ratio (PSNR)

Our evaluation metrics involved using PSNR to assess the quality of the final generated left and right stereo pair image compared to the original image. In our project, we calculate the PSNR for both the left and right stereo images to evaluate the quality and similarity of the generated stereo pair [9, 8].

The PSNR is calculated using the following formula:

$$\text{PSNR} = 20 \cdot \log_{10} \left(\frac{\text{MAX}_I}{\sqrt{\text{MSE}}} \right) \tag{7}$$

Where:

- MAX_I is the maximum possible pixel value of the image (typically 255 for 8-bit images).
- MSE is the MSE between the original image and the processed image.

A higher PSNR value indicates better image quality and less distortion or noise in the processed image compared to the original image.

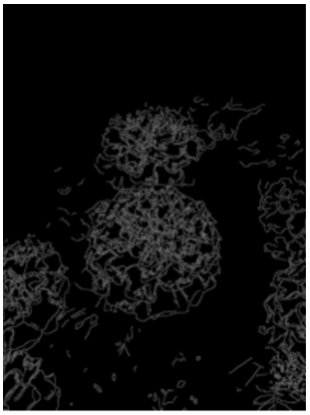

Figure 6. *Result from Edge Detection Consistency*

$$\text{PSNR Ranges} \begin{cases} 40 - 50 : \text{Excellent} \\ 30 - 40 : \text{Good} \\ 20 - 30 : \text{Average/Reasonable} \end{cases} \tag{8}$$

In our project, we obtained average PSNR values of **29.526** for the left image and **29.507** for the right image in the pairs generated. These values suggest a reasonable level of similarity between the original image and the generated stereo pair, so the quality and fidelity of the generated stereo images [9] are good, and the left and right images are consistent with each other.

6.2. Qualitative

We also qualitatively assessed the bokeh effect images, and stereo images to determine the effectiveness of the depth perception.

6.2.1 Edge Detection Consistency

We used the Edge Detection consistency to visually assess the alignment between the edges in the bokeh effect image and the depth map. As seen in Figure 6, the Canny edge detection algorithm was applied to both the bokeh effect image and the depth map [9, 19]. The edges are then separated into foreground and background regions based on a depth threshold, which allows us to clearly see the consistency of the bokeh effect with respect to the depth information. The effect works well because it maintains the sharpness of the foreground edges while effectively blurring the background. This consistency between the bokeh effect and the depth map ensures that the depth-of-field simulation is aligned with the scene's depth structure [19].

We can see that the foreground edges are well-maintained and sharp. This tells us that the bokeh effect has effectively preserved the details of objects closer to the

camera, as we expected. On the other hand, the background edges are minimally present, which shows us that the bokeh effect has effectively blurred the background regions, and this aligns well with the depth-of-field simulation [19].

7. Conclusion Future Work

In this paper, we have presented a novel approach for generating spatial images using monocular depth estimation. Our method leverages latent diffusion models and 3D stereo image techniques to transform standard 2D photos into immersive 3D experiences. Through a combination of depth map generation with an LDM consisting of a pre-trained VAE and our we trained U-Net, Bokeh depth of field effect, parallax shifting, and hole filling algorithms, we have demonstrated an effective pipeline for the simulation of depth perception in 3D spatial images, especially when viewing 2D images through VR or MR devices. The evaluation of our method showed promising results, with our Latent Diffusion Model outperforming other methods such as MiDaS and DiverseDepth in terms of absolute relative error. Among the algorithms we evaluated, while Marigold exhibited the best performance, our model seemed to outperform the others in terms of the absolute relative error. This was the result of our chosen U-net architecture's ability to capture high-level features and spatial structure of the images through the latent space representation when combined with VAE. The resulting depth maps were then used to create stereo image pairs with enhanced depth perception by incorporating the Parallax Effect, Bokeh Effect, and a nearest-neighbor hole-filling algorithm. Given our computational limitations and noise generating depth maps generated by our diffusion model, we used a sample image to show the effectiveness of our stereo rendering algorithm, results of which are seen in Figure 4. The generated stereo image pairs exhibited good quality and consistency, as seen by the PSNR, Laplacian Variance values and our qualitative assessment of edge detection consistency in the foreground and background.

While our proposed approach demonstrates promising results in generating spatial images using monocular depth estimation, there are several avenues for promising future research and improvements. With more compute, and more time to train better models, future work could focus on fine-tuning the LDM parameters further, specifically for depth estimation tasks, and maybe even leveraging larger and more diverse datasets with indoor datasets like NYU Depth V2, and exploring advanced architectures like transformer-based models [36] or multi-task learning approaches. Further, our current hole filling algorithm, based on nearest-neighbor interpolation, works well for small gaps but may not be sufficient for larger gaps or more complex scenes, and would cause the image to appear inconsistent and confusing towards the edges. As a result, filled-in regions near the edges of the image may appear slightly stretched or noisy. To address this limitation, future work could explore more advanced hole filling techniques, such as generative models for this second part as well through Generative Adversarial Networks (GANs). These models could learn to synthesize plausible content for the missing regions based on the surrounding context and depth information, resulting in more natural and coherent stereo images. With our proposed methods and the mentioned ongoing work, it is clear that the field of immersive spatial photo generation is an ever-growing field which continues to yield some innovative solutions to allow people to truly capture a moment in their life, and relive it.

References

[1] Apple Inc. Capture, view, and share spatial photos and videos on apple vision pro. https://support.apple.com/en-gb/guide/apple-vision-pro/dev7068c3c93/visionos, 2024. Apple Support.

[2] S. F. Bhat, I. Alhashim, and P. Wonka. Adabins: Depth estimation using adaptive bins. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4009–4018, 2021.

[3] R. Birkl, D. Wofk, and M. Müller. Midas v3. 1–a model zoo for robust monocular relative depth estimation. *arXiv preprint arXiv:2307.14460*, 2023.

[4] Y. Cabon, N. Murray, and M. Humenberger. Virtual kitti 2, 2020.

[5] W. Chen, Z. Fu, D. Yang, and J. Deng. Single-image depth perception in the wild. *Advances in neural information processing systems*, 29, 2016.

[6] Y.-C. Chen and T.-S. Chang. Image synthesis with efficient defocus blur for stereoscopic displays. *IEEE Access*, 8:176304–176312, 01 2020.

[7] M. Darji. Implementing bilinear interpolation for image resizing. https://meghal-darji.medium.com/implementing-bilinear-interpolation-for-image-resizi February 24 2021. Medium.

[8] S. Dutta. Depth-aware blending of smoothed images for bokeh effect generation. *Journal of Visual Communication and Image Representation*, 77:103089, May 2021.

[9] S. Dutta. Depth-aware blending of smoothed images for bokeh effect generation. *Journal of Visual Communication and Image Representation*, 77:103089, May 2021.

[10] D. Eigen, C. Puhrsch, and R. Fergus. Depth map prediction from a single image using a multi-scale deep network. *Advances in neural information processing systems*, 27, 2014.

[11] H. Fu, M. Gong, C. Wang, K. Batmanghelich, and D. Tao. Deep ordinal regression network for monocular depth estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2002–2011, 2018.

[12] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[13] J. A. Habigt. *Hole-Filling Algorithms for Depth-Image-Based Rendering*. Dissertation, Technische Universität München, Munich, Germany, 2020.

[14] T. Hachaj. Adaptable 2d to 3d stereo vision image conversion based on a deep convolutional neural network and fast inpaint algorithm. *Entropy*, 25, 2023.

[15] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[16] JonathanFly. Jonathanfly/stereo_image_generator_from_single_image. This repository contains code to generate stereo (side by side) image from a single image. https://github.com/JonathanFly/stereo_image_generator_from_single_image, n.d.

[17] J. Jordan. Variational autoencoders. https://www.jeremyjordan.me/variational-autoencoders/, 2018. Retrieved May 25, 2024.

[18] B. Ke, A. Obukhov, S. Huang, N. Metzger, R. C. Daudt, and K. Schindler. Repurposing diffusion-based image generators for monocular depth estimation. *arXiv preprint arXiv:2312.02145*, 2023.

[19] E. Kosheleva, S. Jaiswal, F. Shamsafar, N. Cheema, K. Illgner-Fehns, and P. Slusallek. Edge-aware consistent stereo video depth estimation, 2023.

[20] P. Kozankiewics. Fast algorithm for creating image-based stereo images - wscg. http://wscg.zcu.cz/wscg2002/Papers_2002/A67.pdf, 2022.

[21] J. H. Lee, M.-K. Han, D. W. Ko, and I. H. Suh. From big to small: Multi-scale local planar guidance for monocular depth estimation. *arXiv preprint arXiv:1907.10326*, 2019.

[22] Z. Li and N. Snavely. Megadepth: Learning single-view depth prediction from internet photos. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2041–2050, 2018.

[23] F. Liu, C. Shen, and G. Lin. Deep convolutional neural fields for depth estimation from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5162–5170, 2015.

[24] L. Liu, L. Zhou, and Y. Dong. Bokeh rendering based on adaptive depth calibration network, 2023.

[25] P. K. Nathan Silberman, Derek Hoiem and R. Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012.

[26] Ophthalmic Photographers' society. Stereo photography - ophthalmic photographers' society. https://www.opsweb.org/blogpost/777793/Stereo-Photography?tag=anaglyph, n.d.

[27] Owl3D. 2d to stereoscopic 3d with ai: Depth map from a single image. https://www.owl3d.com/blog/2d-to-stereoscopic-3d-with-ai-depth-map-from-a-single-image, n.d.

[28] K. Pryzemyslaw. Fast algorithm for creating image-based stereo images. http://wscg.zcu.cz/wscg2002/Papers_2002/A67.pdf, 2022. WSCG.

[29] R. Ranftl, K. Lasinger, D. Hafner, K. Schindler, and V. Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE transactions on pattern analysis and machine intelligence*, 44(3):1623–1637, 2020.

[30] M. Roberts, J. Ramapuram, A. Ranjan, A. Kumar, M. A. Bautista, N. Paczan, R. Webb, and J. M. Susskind. Hypersim: A photorealistic synthetic dataset for holistic indoor scene understanding. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10912–10922, 2021.

[31] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[32] A. Rosebrock. Blur detection with opencv. PyImageSearch, April 17 2021. https://pyimagesearch.com/2015/09/07/blur-detection-with-opencv/.

[33] C. Schuhmann, R. Beaumont, R. Vencu, C. Gordon, R. Wightman, M. Cherti, T. Coombes, A. Katta, C. Mullis, M. Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *Advances in Neural Information Processing Systems*, 35:25278–25294, 2022.

[34] K. Schwarz, S. W. Kim, J. Gao, S. Fidler, A. Geiger, and K. Kreis. Wildfusion: Learning 3d-aware latent diffusion models in view space, 2024.

[35] K. Schwarz, S. W. Kim, J. Gao, S. Fidler, A. Geiger, and K. Kreis. Wildfusion: Learning 3d-aware latent diffusion models in view space, 2024.

[36] J. Shao, Y. Yang, H. Zhou, Y. Zhang, Y. Shen, M. Poggi, and Y. Liao. Learning temporally consistent video depth from video diffusion priors, 2024.

[37] G. B. M. Stan, D. Wofk, S. Fox, A. Redden, W. Saxton, J. Yu, E. Aflalo, S.-Y. Tseng, F. Nonato, M. Muller, and V. Lal. Ldm3d: Latent diffusion model for 3d, 2023.

[38] B. Wang, D. Zhang, Y. Su, and H. Zhang. Enhancing view synthesis with depth-guided neural radiance fields and improved depth completion. *Sensors*, 24(6):1919, 2024.

[39] F. Wang, Y. Zhang, Y. Ai, and W. Zhang. Rendering natural bokeh effects based on depth estimation to improve the aesthetic ability of machine vision. *Machines*, 10(5):286, 2022.

[40] L. Wang, J. R. Frisvad, M. B. Jensen, and S. A. Bigdeli. Stereodiffusion: Training-free stereo image generation using latent diffusion models, 2024.

[41] X. Wang, W. Yin, T. Kong, Y. Jiang, L. Li, and C. Shen. Task-aware monocular depth estimation for 3d object detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 12257–12264, 2020.

[42] Y. Wang, W.-L. Chao, D. Garg, B. Hariharan, M. Campbell, and K. Q. Weinberger. Pseudo-lidar from visual depth estimation: Bridging the gap in 3d object detection for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[43] D. Wofk, F. Ma, T.-J. Yang, S. Karaman, and V. Sze. Fastdepth: Fast monocular depth estimation on embedded systems. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6101–6108, 2019.

[44] W.-W. Wu and Z.-Y. Wang. Research and implementation of 3d stereo display based on parallax. In *2010 International Conference on Audio, Language and Image Processing*, pages 307–311, 2010.

[45] W. Yin, X. Wang, C. Shen, Y. Liu, Z. Tian, S. Xu, C. Sun, and D. Renyin. Diversedepth: Affine-invariant depth prediction using diverse data, 2020.