Shape and Texture: What Influences Reliable Optical Flow Estimation? Supplemental Material

Libo Long, Xiao Hu, Jochen Lang
EECS, University of Ottawa
{ llong014, xhu008, jlang}@uottawa.ca

1. Optical Flow Update due to Shape Changes

1.1. Upscaling

Our dataset contains upscaled shapes in Flow-R-S. In this section, we will give the derivation of Equation 8 in Section 3.

Given that we have upscaled the objects (x) in frames I_1 and I_2, it is necessary to rescale the optical flow by re-projecting each point from I_1 to align with the new positions in frame I_2 inside the upscaled area $x^{\uparrow}$.

The upscaling is performed based on the common center of the bounding boxes of the object x and its upscaled counterpart $x^{\uparrow}$. Let $c_1 = (c_1^x, c_1^y)$ be the center point of x in I_1. We can determine the corresponding center point $c_2 = (c_2^x, c_2^y)$ in I_2 by referencing the ground truth of the optical flow f_{gt}.

Next, for each point $(i_1^{\uparrow}, j_1^{\uparrow})$ in the upscaled object $x^{\uparrow}$, we need to identify the corresponding point $(i_2^{\uparrow}, j_2^{\uparrow})$ in $I_2^{x^{\uparrow}}$. The new offset is therefore $(i_2^{\uparrow}, j_2^{\uparrow}) - (i_1^{\uparrow}, j_1^{\uparrow})$.

We proceed by only considering the first coordinate i for notational clarity. Then, given scale factor θ and center point c_1^x, $i_1^{\uparrow}$ is found by

$$i_1^{\uparrow} = c_1^x + \theta(i_1 - c_1^x) \tag{1}$$

Thus, we can find the original i_1 in I_1 by

$$i_1 = \frac{i_1^{\uparrow} - c_1^x}{\theta} + c_1^x \tag{2}$$

We can find the same corresponding relation in I_2 by

$$i_2^{\uparrow} = c_2^x + \theta(i_2 - c_2^x) \tag{3}$$

$$i_2 = \frac{i_2^{\uparrow} - c_2^x}{\theta} + c_2^x \tag{4}$$

For each i_1, we can find the corresponding i_2 with the flow f_{gt}.

$$i_2 = i_1 + f_i \tag{5}$$

where f_i is the flow for offset i, in i_1 index. Now, we can calculate the updated flow by

$$\begin{aligned} f_i^s = i_2^{\uparrow} - i_1^{\uparrow} &= c_2^x + \theta(i_2 - c_2^x) - i_1^{\uparrow} \\ &= c_2^x + \theta(i_1 + f_i - c_2^x) - i_1^{\uparrow} \\ &= c_2^x + \theta\left(\frac{i_1^{\uparrow} - c_1^x}{\theta} + c_1^x + f_i - c_2^x\right) - i_1^{\uparrow} \\ &= c_2^x + i_1^{\uparrow} - c_1^x + \theta c_1^x + \theta f_i - \theta c_2^x - i_1^{\uparrow} \\ &= c_2^x - c_1^x + \theta c_1^x - \theta c_2^x + \theta f_i \\ &= (\theta - 1)(c_1^x - c_2^x) + \theta f_i \end{aligned}$$

by adding back the second coordinate j, with $f_{ij} \in x \uparrow$, we arrive at the updated flow (Equation 8 of the main paper):

$$\begin{aligned} f_{ij}^s &= \theta f_{ij} + (\theta - 1)(c_1^x - c_2^x, c_1^y - c_2^y) \\ &= \theta f_{ij} + (\theta - 1)(c_1 - c_2) \end{aligned}$$

Optical flow for shape attack. In Section 4.1 Experiments - Shape attack, we rescale I_1 by θ. Consequently, the optical flow can be updated using the expression $i_2 - i_1^{\uparrow}$ in $x^{\uparrow}$.

$$\begin{aligned} i_2 - i_1^{\uparrow} &= i_1 + f_i - i_1^{\uparrow} \\ &= \frac{i_1^{\uparrow} - c_1^x}{\theta} + c_1^x + f_i - i_1^{\uparrow} \end{aligned}$$

Hence, in two dimensions, we have

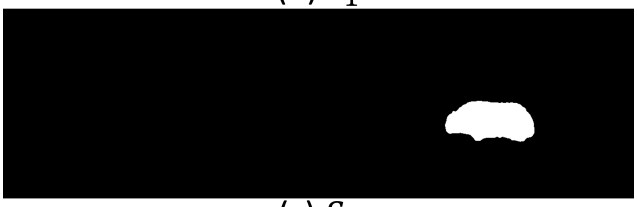
$$(i_2, j_2) - (i_1^{\uparrow}, j_1^{\uparrow}) = \frac{(i_1^{\uparrow}, j_1^{\uparrow}) - (c_1^x, c_1^y)}{\theta} + (c_1^x, c_1^y) + f_{i,j} - (i_1^{\uparrow}, j_1^{\uparrow})$$

1.2. General Affine Transformation

In the main paper, we only discuss shape change due to upscaling as this leads to realistic and significant shape

(a) Screenshot of APP

(b) S_1



(c) S_2

Figure 1. **Screenshot of our APP**. (a) GUI of the app. It takes mouse clicks as input, and pressing the "Generate Mask" button will trigger the automatic generation of a pair of masks as shown in (b) and (c).

changes for many objects, i.e., similar to the object approaching the camera. Here, we derive the optical flow update due to a general affine transformation which includes (up)scaling but also rotations, shearing and translations.

- $\Theta_1 \in \mathbb{R}^{3\times3}$ and $\Theta_2 \in \mathbb{R}^{3\times3}$ are the affine transformation matrices of the object in the frame 1 and 2, respectively.
- $c_1 \in \mathbb{R}^2$ is the center position of the object bounding box in frame 1. $C_1 \in \mathbb{R}^{3\times3}$ is the corresponding translation matrix between the origin and c_1. $C_1 = \begin{bmatrix} 1 & 0 & c_1^x \\ 0 & 1 & c_1^y \\ 0 & 0 & 1 \end{bmatrix}$
- $p_1^t \in \mathbb{R}^2$ represents a pixel belonging to the object in the frame 1. t is the pixel index.
- $\hat{p}_1^t \in \mathbb{R}^2$ represents the object pixel p_1^t after the affine transformation in the frame 1.
- $f_{ij} \in \mathbb{R}^2$ is the optical flow for the pixel p^t from frame 1 to frame 2. $F_{ij} \in \mathbb{R}^{3\times3}$ is the corresponding translation matrix from p_1^t to p_2^t. $F_{ij} = \begin{bmatrix} 1 & 0 & f_i \\ 0 & 1 & f_j \\ 0 & 0 & 1 \end{bmatrix}$
- $f_{ij}^s \in \mathbb{R}^2$ is the optical flow for the object pixel p^t after the affine transformation.

In our setup, all the affine transformations are centered in the object bounding box. The goal is to calculate the relationship between f_{ij} and f_{ij}^s. p^t and $\hat{p}^t$ are converted into homogeneous coordinates for the affine transformations. The transformation matrix from $\hat{p}_1^t$ to $\hat{p}_2^t$ can be calculated as:

$$\begin{aligned} \hat{p}_2^t &= C_2\Theta_2C_2^{-1}p_2^t \\ p_2^t &= F_{ij}p_1^t \\ p_1^t &= C_1\Theta_1^{-1}C_1^{-1}\hat{p}_1^t \\ \hat{p}_2^t &= C_2\Theta_2C_2^{-1}F_{ij}C_1\Theta_1^{-1}C_1^{-1}\hat{p}_1^t \end{aligned}$$

Considering a scenario when the affine transformations Θ_1 and Θ_2 for two frames are the same and don't contain any translations. Then, with $\theta \in \mathbb{R}^{2\times2}$ being the non-translation matrix and I the identity matrix, the above equations can be simplified without using the homogeneous coordinates as:

$$\begin{aligned} \hat{p}_2^t &= c_2 + \theta(-c_2 + f_{ij} + c_1 + \theta^{-1}(-c_1 + \hat{p}_1^t)) \\ &= c_2 - \theta c_2 + \theta f_{ij} + \theta c_1 - c_1 + \hat{p}_1^t \\ f_{ij}^s &= \hat{p}_2^t - \hat{p}_1^t \\ &= (I-\theta)c_2 + \theta f_{ij} + (\theta - 1)c_1 \\ &= \theta f_{ij} + (I-\theta)(c_2 - c_1) \\ &= \theta f_{ij} + (\theta - I)(c_1 - c_2) \end{aligned}$$

2. Occlusion Handling

In this section, we will discuss how to handle the occlusion for Flow-R.

Occlusion in Flow-R-T. The optical flow of Flow-R-T is the same as the original flow. To ensure consistency, we use the ground truth optical flow to warp the new texture from I_2 to I_1. Thus, all occluded pixels should be the same.

Occlusion in Flow-R-S. To easily handle the occlusion area, we only upscale the objects x. Thus, the optical flow should be recalculated in $x^{\uparrow}$. To indicate the new occlusion area, we first create an occlusion map $Mask_x$ of x. Then, we use nearest neighbor interpolation to generate the new occlusion map for $x^{\uparrow}$.

Occlusion in Flow-R-O. We use an affine transformation to represent the motion of the new object from I_1 to I_2. In practice, we avoid using rotation and moving out of the image, as these could generate complex occlusions. Instead, we only add linear motion, ensuring that each pixel of the new object in I_1 can project to the corresponding pixels in

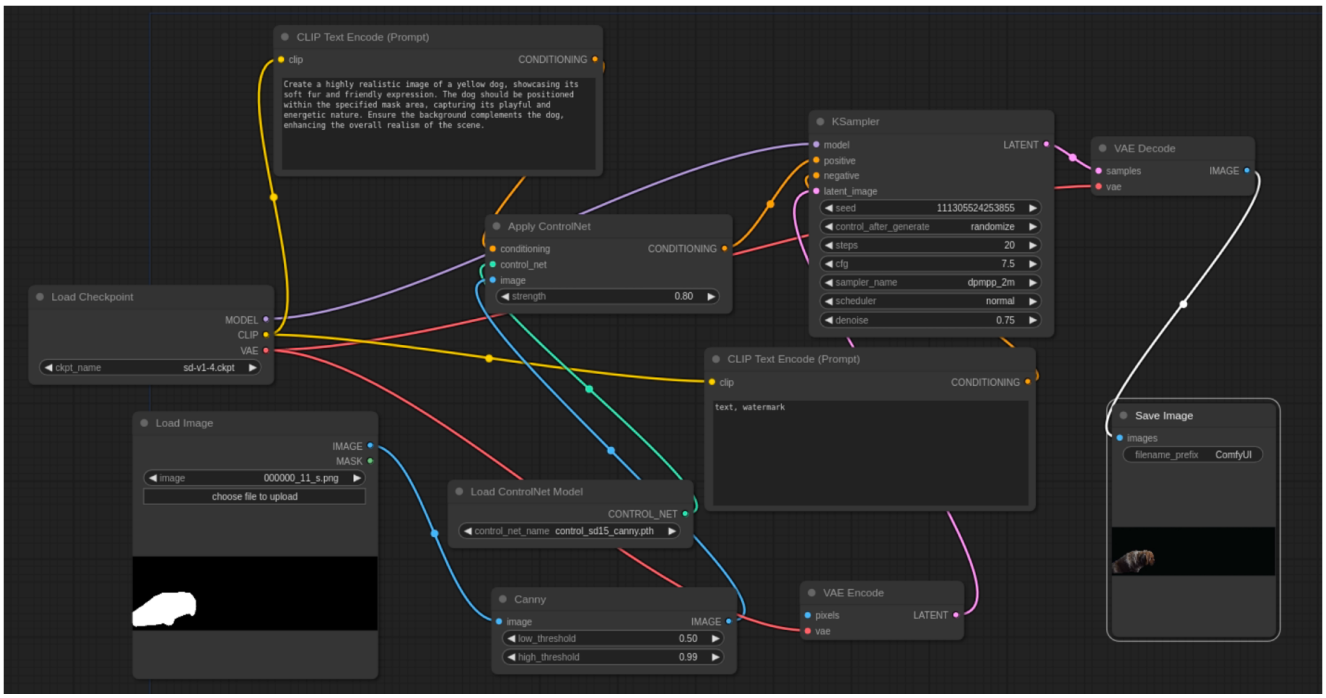


Figure 2. **The pipeline for texture generation.**

I_2. Thus, there is no occlusion of optical flow in the new object region. We find that even with these linear motion cases, all methods still fail to predict unseen objects, which raises our concern.

3. Mask Segmentation

To facilitate the corresponding segmentation, we developed a simple GUI-based application using SAM [4]. This application allows users to interactively segment the same object in a pair of images simply by clicking with the mouse.

Given I_1, I_2, f_{gt}, we can get the segmentation of any instance in I_1 by

$$S_1 = SAM(I_1, Prompt(P1)) \tag{6}$$

Where $P1$ is a set of indices obtained by a series of mouse clicks. We can match the corresponding points from $P1$ in I_1 to I_2 using the ground truth optical flow f_{gt}. The mask of the same object in I_2 is

$$P2 = P1 + f_{gt}(P1), \tag{7}$$

$$S_2 = SAM(I_2, Prompt(P2)) \tag{8}$$

The screenshot of the APP is shown in Figure 1.

4. Generate Textures with ControlNet

To generate high-quality textures with fixed shape, we utilize a Large Language Model (LLM) [5] to generate a precise prompt for the mask S_2. Specially, we feed the S_2 into GPT4 and ask the following questions:

Q1:Can you recognize which object this mask represents? Please list n of the most likely common objects that fit in the mask.

Here, n is a number of objects, then we select the most common objects and ask the following question.

Q2:Provide a prompt for ControlNet [9]: [object] with [color/texture] that appears realistic in the real world, ensuring it fits within the provided [mask].

With the output prompt N of and S_2, the new image can be generated by:

$$Image(N) = ControlNet(N, S_2) \tag{9}$$

We use ComfyUI [6, 7] to batch-generate texture images. A brief pipeline is shown in Figure 2.

5. More Examples of Flow-R

We present more examples of Flow-R in Figure 3. The first, second, third, and fourth columns indicate the corresponding data from KITTI, Flow-R-T, Flow-R-S, and Flow-R-O, respectively. For every three rows, indicate frames 1 and 2, along with the visualization of optical flow.

Code and data are available at https://github.com/llesky/Robustness-Analysis-Optical-Flow.

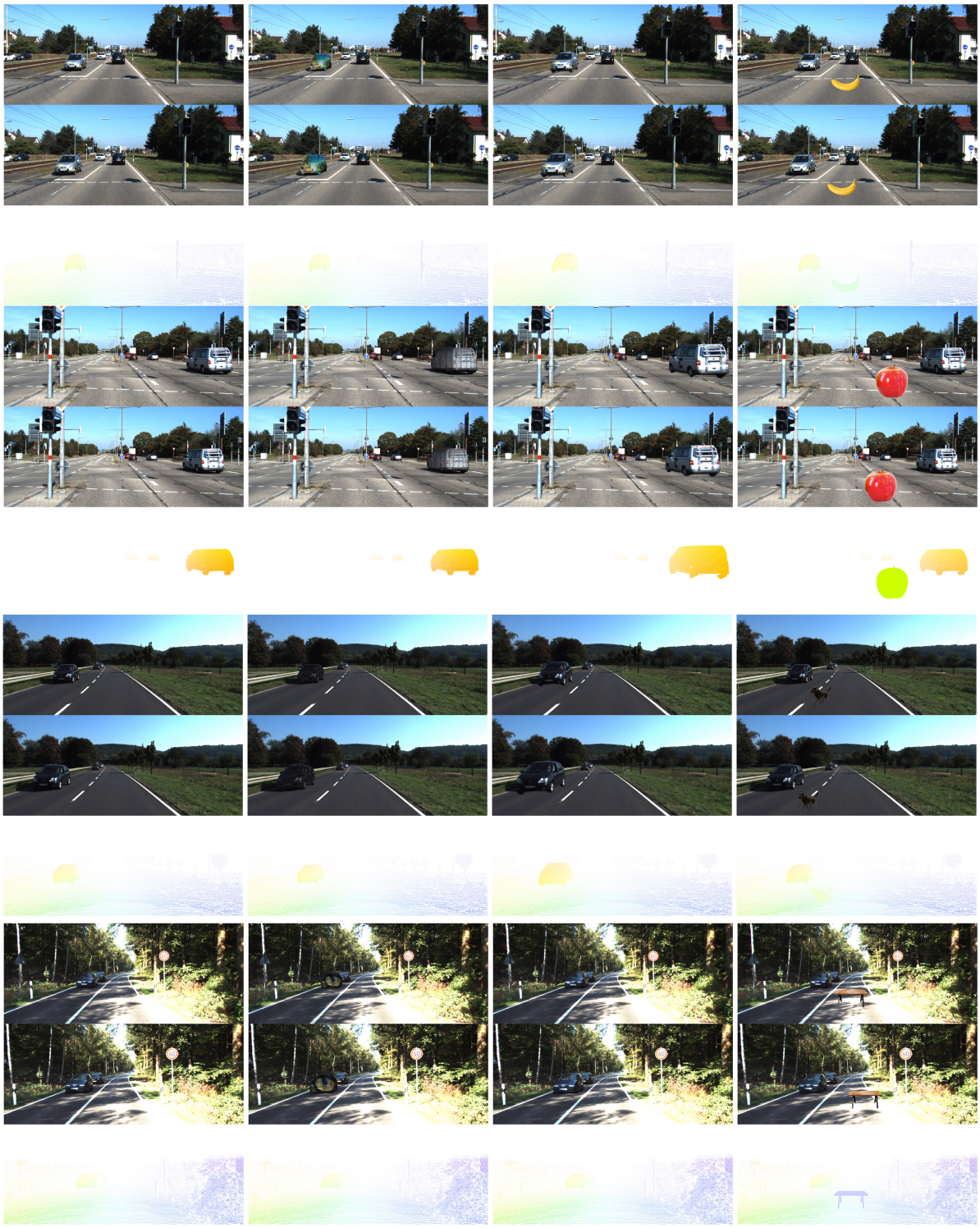


Figure 3. **More Examples of Flow-R.**

Method	Sintel-R-T		Sintel-R-S		Sintel-R-O(SEEN)		Sintel-R-O(UNSEEN)	
	Clean	Final	Clean	Final	Clean	Final	Clean	Final
RAFT[8]	0.83	1.33	1.76	2.28	2.24	2.67	6.8	7.26
GMA[3]	0.71	1.21	1.60	2.14	2.31	2.53	6.72	7.11
Flowformer[2]	0.54	0.85	1.37	1.66	2.27	2.55	6.94	7.42

Table 1. **Quantitative results on MPI-Sintel [1] dataset.** We report the average end-point error (AEPE).

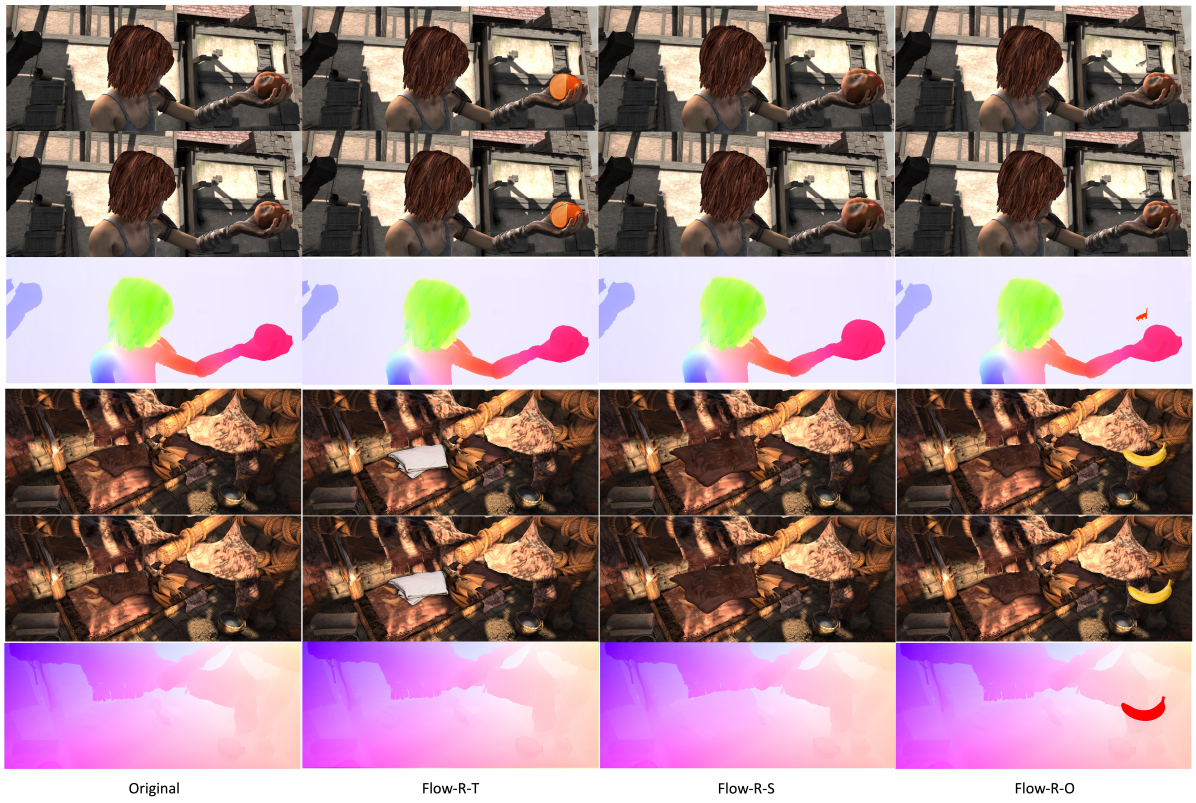


Figure 4. **Our Flow-R Method applied to MPI-Sintel [1].** Two example image pairs with results for Flow-R-T, Flow-R-S and Flow-R-O.

6. Additional Experiments

We report additional experiments with **MPI-Sintel**[1] (see Figure 4), and observe similar behavior as with the KITTI data (see Table 1). One interesting phenomenon is that by using the weights trained in Stage 3 (T+K+S+H) and testing on MPI-Sintel, unseen objects tend to be ignored. However, we find that the failure cases are fewer than in KITTI. We assume that training with more domains or more scenes could help reduce this error further.

We conducted further experiments with affine transforms (Figure 5) of scaling $[0.3, 2]$, rotation $[-30°, +30°]$, and translation $[-50, +50]$ (original offset Δx + translation). According to Table 2, we observe that rotation significantly increases errors because less pixels are matched.

	Scaling	Rotation	Translation	Mix
RAFT	1.93	2.87	2.36	3.10
GMA	1.86	2.64	2.31	3.02

Table 2. **Quantitative results with affine transforms.**

References

[1] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In *European Conference on Computer Vision*, pages 611–625, 2012. 5

[2] Zhaoyang Huang, Xiaoyu Shi, Chao Zhang, Qiang Wang, Ka Chun Cheung, Hongwei Qin, Jifeng Dai, and Hongsheng Li. FlowFormer: A transformer architecture for optical flow.

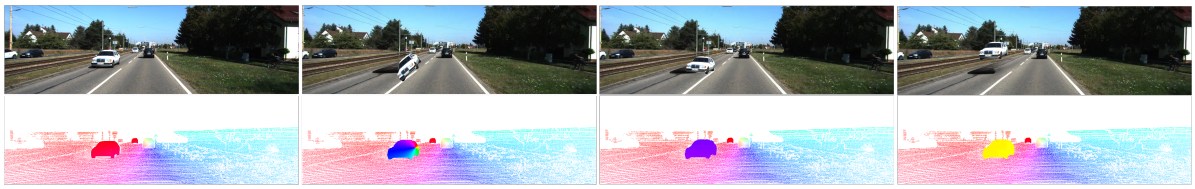


Figure 5. **Visualization of affine transformation.**

European Conference on Computer Vision, 2022. 5

[3] Shihao Jiang, Dylan Campbell, Yao Lu, Hongdong Li, and Richard Hartley. Learning to estimate hidden motions with global motion aggregation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9772–9781, 2021. 5

[4] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything. *arXiv:2304.02643*, 2023. 3

[5] OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, and Florencia Leoni Aleman et al. Gpt-4 technical report, 2024. 3

[6] Comfy Org. Comfyui: A modular gui for stable diffusion, 2023. Accessed: 2024-11-21. 3

[7] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10807–10817. IEEE, 2022. 3

[8] Zachary Teed and Jia Deng. RAFT: Recurrent All-Pairs Field Transforms for Optical Flow. In *European Conference on Computer Vision*, page 402–419, Glasgow, UK, 2020. 5

[9] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3836–3847, 2023. 3