The Change You Want to See (Now in 3D)
Supplementary Material

1. Outline

In Section 2 we provide python pseudo-code for our architecture and the forward pass. In Section 3 we provide some examples of estimated registration in real world images. In Section 4 we outline some of the challenges with VIRAT-STD annotations [2].

2. Architecture pseudo-code

```
class Architecture(nn.Module):
    def __init__(self):
        super().__init__()
        self.feature_backbone = DinoViT() # returns dino features from the 3rd and 12th blocks
        self.register_and_subtract = DFRM() # returns the difference of registered features
        self.unet_encoder = UnetEncoder(
            input_channels=[768, 512, 512, 512],
            output_channels=[512, 512, 512, 512])
        self.unet_decoder = UnetDecoder(
            encoder_channels=[768, 512, 512, 512, 512],
            decoder_channels=[256, 256, 128, 128, 64],
            attention_type="scse")
        self.feature_fusion = nn.Sequential(
            create_conv_layer_with_kaiming_init(64 + 768, 256, kernel_size=3, stride=1, padding=1),
            create_batch_norm_layer_with_custom_init(256), F.gelu(),
            create_conv_layer_with_kaiming_init(256, 64, kernel_size=3, stride=1, padding=1),
            create_batch_norm_layer_with_custom_init(64), F.gelu())
        self.centernet_head = CenterNetHead(in_channel=64, feat_channel=64)

    def forward(self, batch):
        image1_dino_features = self.feature_backbone(batch["image1"])
        image2_dino_features = self.feature_backbone(batch["image2"])
        image1_encoded_features, image2_encoded_features = [image1_dino_features[1]], [
                                                            image2_dino_features[1]]
        for layer in self.unet_encoder:
            image1_encoded_features.append(layer(image1_encoded_features[-1]))
            image2_encoded_features.append(layer(image2_encoded_features[-1]))
        for i in range(len(image1_encoded_features)):
            image1_encoded_features[i], image2_encoded_features[i] = self.register_and_subtract(
                                                                    image1_encoded_features[i],
                                                                    image2_encoded_features[i])
        image1_decoded_features, image2_decoded_features = self.unet_decoder(*image1_encoded_features),
                                                             self.unet_decoder(*image2_encoded_features
                                                            )
        image1_decoded_features = self.feature_fusion(image1_dino_features[0], image1_decoded_features)
        image2_decoded_features = self.feature_fusion(image2_dino_features[0], image2_decoded_features)
        return self.centernet_head([image1_decoded_features]), self.centernet_head([
                                                            image2_decoded_features])
```

3. Registration examples

In Figure 1, we show examples of estimated correspondences and depth used to register the input images, and additionally the warped images. Please note that the depth maps (even "ground truth") are not pixel-perfect, as evident by the "floating points" in the warped images. This results in a false signal for multiple very small changes, which the model must ignore in favour of the actual changes.

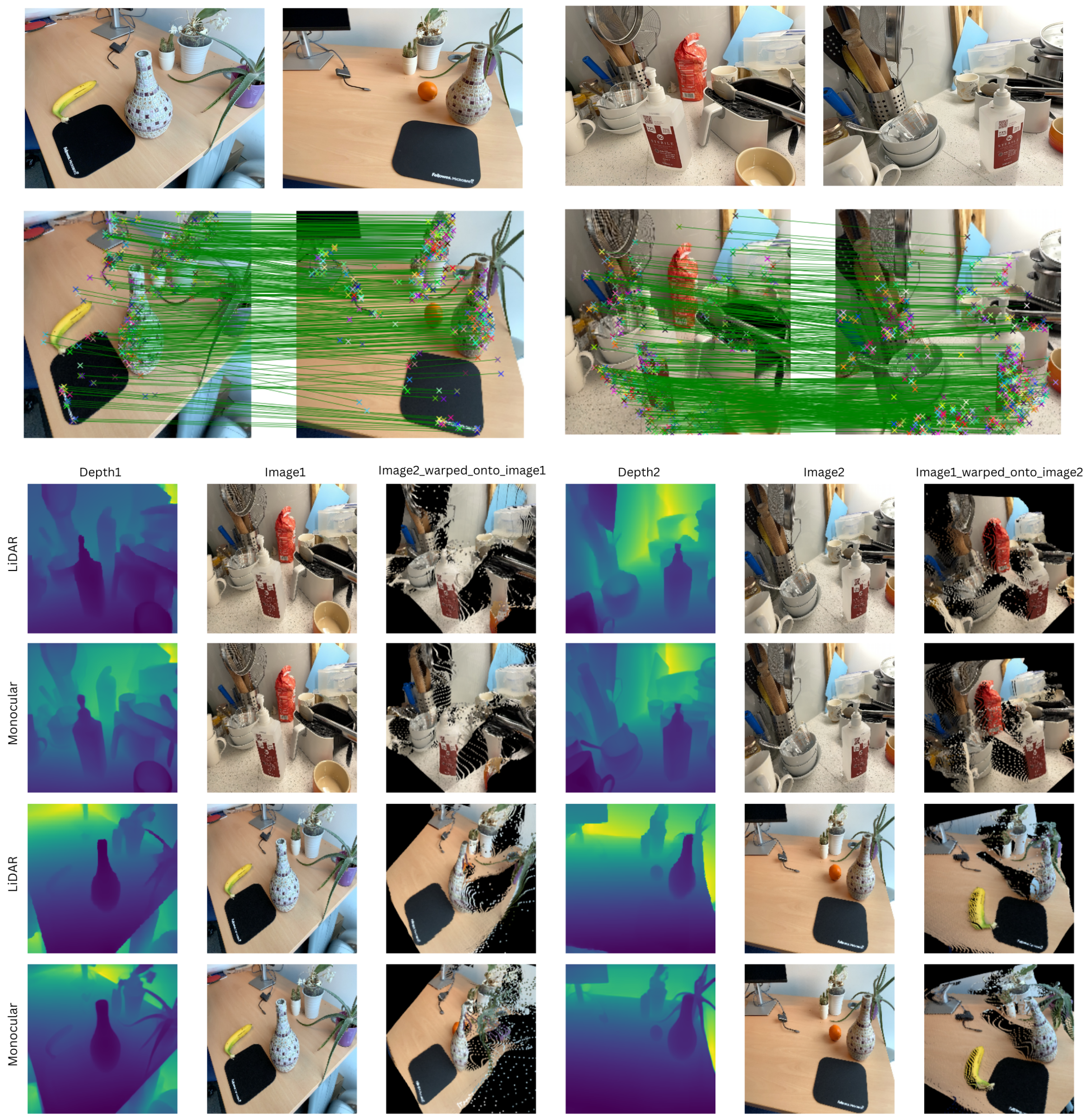

Figure 1. Examples of registration via DFRM on real world images, where correspondences are obtained using SuperGlue [3] and depth is either obtained using built-in LiDAR in Apple iPhone/iPad or using monocular depth estimation methods [1]. Note: Here we warp RGB input images for visualisation purposes only. In practise, we warp c-dimensional feature maps instead. Notice the black regions in the warped images. These regions were not visible in the original image and therefore after warping are "empty". We represent these using a visibility mask.

4. Extremely small annotations problem

In Figures 2 and 3, we display sample images from the VIRAT-STD dataset and their corresponding "ground truth" annotations in blue. Some annotations are so small that they remain difficult to see, even after resizing the images for visualization purposes. Additionally, certain regions that have undergone changes are not annotated. These examples are not selectively chosen; the entire test set is plagued by this issue, as can be verified. This presents a problem when the images are resized to 224×224 from the original resolution.

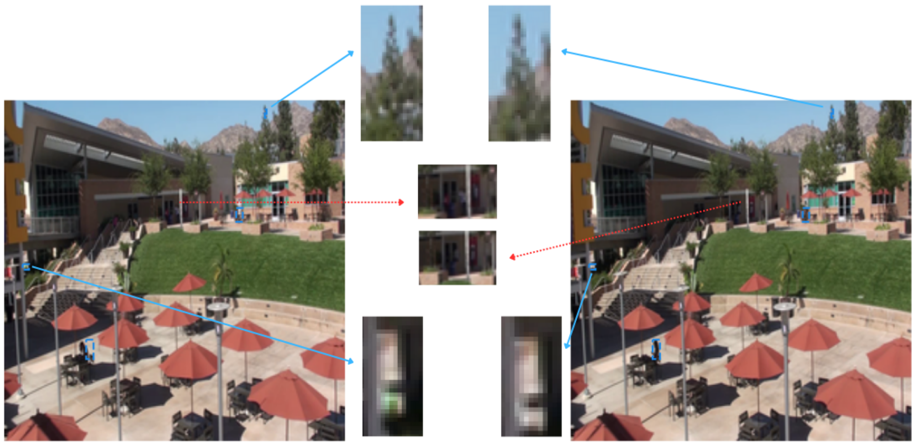

Figure 2. Using 2x blue (solid) arrows, we show some zoomed-in regions that are in the "ground truth" changes. Using 1x red (dashed) arrow, we draw attention to regions that are not part of the "ground truth" but in fact contain valid changes (a person in a white top just to the left of the pole in image 1 is not there in image 2).

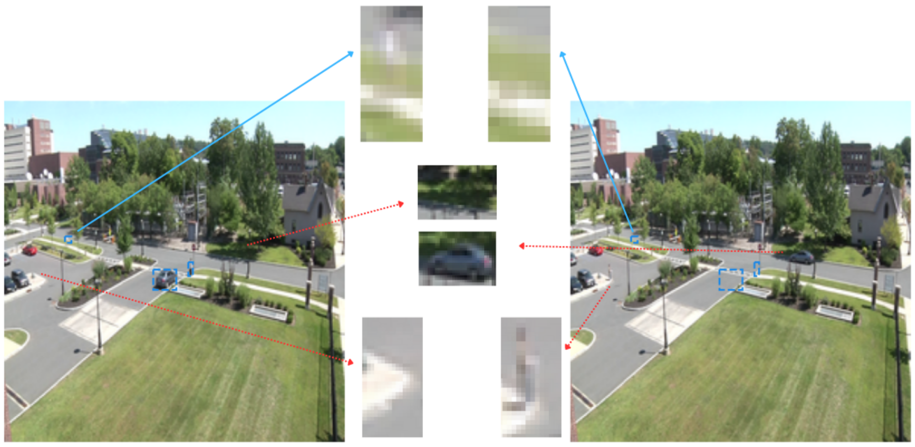

Figure 3. Using 1x blue (solid) arrow, we show some zoomed-in regions that are in the "ground truth" changes. Using 2x red (dashed) arrows, we draw attention to regions that are not part of the "ground truth" but in fact contain valid changes (a car and a person, that do not exist in image 1).

To partially address this issue, we remove any annotations where the area of the bounding box is less than 5×5 px (in the original image, i.e. before resizing). However, this is still a challenging task. In the paper, we only use the filtered annotations for both our results and those of [2]. We did not attempt to resolve the problem of missing annotations. Figure 4 illustrates examples of the annotations that we retained and those that we discarded after filtering.

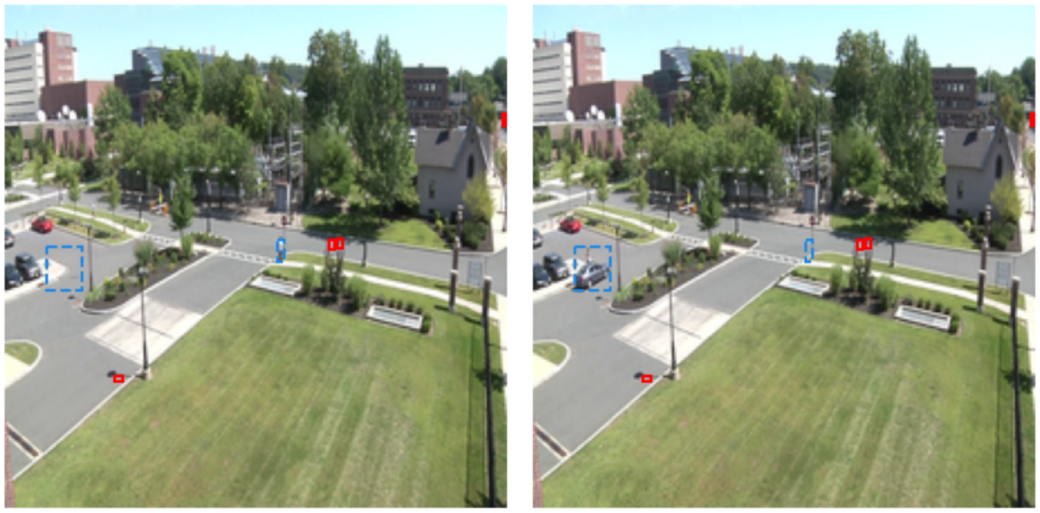

Figure 4. We keep the annotations in blue (dashed) but remove the ones in red (solid). Note the images are resized here for visualisation purposes, the actual input resolution is much smaller.

References

[1] Shariq Farooq Bhat, Reiner Birkl, Diana Wofk, Peter Wonka, and Matthias Müller. Zoedepth: Zero-shot transfer by combining relative and metric depth, 2023. 2

[2] Ragav Sachdeva and Andrew Zisserman. The change you want to see. In *Winter Conference on Applications of Computer Vision (WACV)*, 2023. 1, 3

[3] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4938–4947, 2020. 2