Supplementary Material: Long-Term Temporally Consistent Unpaired Video Translation from Simulated Surgical 3D Data

Dominik Rivoir[1,2], Micha Pfeiffer[1], Reuben Docea[1], Fiona Kolbinger[1,3], Carina Riediger[3], Jürgen Weitz[2,3], Stefanie Speidel[1,2]

[1]NCT/UCC Dresden, Germany, [2]CeTI, TU Dresden, [3]University Hospital Dresden

{dominik.rivoir, micha.pfeiffer, reuben.docea, stefanie.speidel}@nct-dresden.de

{fiona.kolbinger, carina.riediger, juergen.weitz}@uniklinikum-dresden.de

1. Implementation details

1.1. Neural Texture & Projection Mechanism

Here, we go into more detail on the projection mechanism between texture and pixel space and from which we obtain the image-sized feature map a_i^{tex}:

$$\boldsymbol{a_i^{tex}} = project(tex, view_i) \tag{1}$$

In the following, we describe how the projection works for a single pixel (x, y) of a given $view_i$.

First, a ray is cast through the pixel to the surface of the $3D$ scene. We obtain a surface coordinate $s \in \mathbb{R}^3$ and its corresponding object $o \in$ {'liver', 'gallbladder', 'ligament', 'abdominal wall', 'fat'}.

$$ray(view_i, x, y) = s, o \tag{2}$$

From the surface point s, we find the corresponding texel coordinates for each of the three texture planes which face the surface point (*i.e.* where the dot-product of normals is positive). Assuming surface and texture normals are never exactly orthogonal in practice, $tri(s)$ always returns three sets of texture plane coordinates.

$$\begin{aligned} tri(s) &= \big\{proj_p(s) | p \in \{1..6\} \wedge (n_s^T \cdot n_p) > 0\big\} \\ &= \{(x_{p_1}, y_{p_1}), (x_{p_2}, y_{p_2}), (x_{p_3}, y_{p_3})\} \end{aligned} \tag{3}$$

where x_{p_i}, y_{p_i} are the texel coordinates of s projected onto texture plane p_i and transformed into texel space.

To obtain the final, projected features in pixel space, we compute a weighted average of the texture features $tex[o, p, x_p, y_p]$ from the three planes.

$$\boldsymbol{a_i^{tex}[x, y]} = \sum_{x_p, y_p}^{tri(s)} w_{tri}(s, p) \cdot tex[o, p, x_p, y_p] \tag{4}$$

$$w_{tri}(s, p) = (n_s^T \cdot n_p)^2 \tag{5}$$

Features are weighted according to the squared dot-product of surface and texture normal, *i.e.* the texture plane which faces the surface point most, contributes most to the final feature vector. Note that $\|n_s\| = 1$ and n_p are orthogonal one-hot or negative one-hot vectors. Hence, the sum of weights is always $n_{s,x}^2 + n_{s,y}^2 + n_{s,z}^2 = \|n_s\|^2 = 1$.

To obtain texture features $tex[o, p, x_p, y_p]$ from continuous coordinates x_p, y_p, we use bilinear interpolation:

$$\begin{aligned} & tex[o, p, x_p, y_p] \\ = \;& (1-\Delta x) \;\; (1-\Delta y) \;\; tex\big[o, p, \lfloor x_p \rfloor, \lfloor y_p \rfloor\big] \\ + \;& (1-\Delta x) \;\; \Delta y \;\; tex\big[o, p, \lfloor x_p \rfloor, \lceil y_p \rceil\big] \\ + \;& \Delta x \;\; (1-\Delta y) \;\; tex\big[o, p, \lceil x_p \rceil, \lfloor y_p \rfloor\big] \\ + \;& \Delta x \;\; \Delta y \;\; tex\big[o, p, \lceil x_p \rceil, \lceil y_p \rceil\big] \end{aligned} \tag{6}$$

where $\Delta x = x_p - \lfloor x_p \rfloor$ and $\Delta y = y_p - \lfloor y_p \rfloor$.

1.2. Warping

To enforce the view-consistency loss L_{vc}, we have to warp the translated image $\hat{b}_j$ of $view_j$ into the pixel space of $view_i$. To this end, we define the warping operator $w_i(\cdot)$ to obtain the warped image

$$\boldsymbol{w_i(\hat{b}_j)}. \tag{7}$$

To map pixel coordinates x, y from $view_j$ to $view_i$, we first use the ray function $ray(view_j, x, y)$ from $view_j$ to obtain a $3D$ surface point $s = (s_x, s_y, s_z)$. Using the $view_i$'s intrinsic and extrinsic camera parameters, s is projected back into $view_i$ and we obtain the corresponding pixel coordinates x_i, y_i.

$$\begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} = C_i \cdot RT_i \cdot \begin{pmatrix} s_x \\ s_y \\ s_z \\ 1 \end{pmatrix}, \tag{8}$$

where $C_i \in \mathbb{R}^{3\times3}$ is $view_i$'s intrinsic matrix with focal lengths f_u, f_v and principal point (u_0, v_0) and $RT_i \in \mathbb{R}^{3\times4}$ is its extrinsic matrix with rotational and translational parameters $\{r_{..}\}, \{t_.\}$. All parameters can be extracted from the simulated scene.

$$C_i = \begin{pmatrix} f_u & 0 & u_0 \\ 0 & f_v & v_0 \\ 0 & 0 & 1 \end{pmatrix} \tag{9}$$

$$RT_i = \begin{pmatrix} r_{00} & r_{01} & r_{02} & t_0 \\ r_{10} & r_{11} & r_{12} & t_1 \\ r_{20} & r_{21} & r_{22} & t_2 \end{pmatrix} \tag{10}$$

Using this correspondence between pixel coordinates of both views, RGB values can now be mapped into the warped image:

$$\boldsymbol{w_i(\hat{b}_j)[x_i, y_i]} = \begin{cases} \hat{b}_j[x, y] & \text{if } \neg occl_i(s, s_i) \\ 0, & \text{else.} \end{cases} \tag{11}$$

where $s, \_ = ray(view_j, x, y)$ is the $3D$ surface point we want to warp and $s_i, \_ = ray(view_i, x_i, y_i)$ is the surface point we obtain if a ray is cast into the surface from the warped location in the target view i. Note that that we do not map surface points from $view_j$ if they are occluded in $view_i$. To this end, we define the boolean occlusion function $occl_i(s_1, s_2)$:

$$occl_i(s_1, s_2) = depth_i(s_1) > (depth_i(s_2) + \epsilon), \tag{12}$$

where $depth_i(s)$ is the depth of a surface point s from $view_i$. To account for inaccuracies, we say it is an occlusion if the former depth is larger than the latter by a margin of $\epsilon = 1$mm in the blender scene.

1.3. Network architectures

We repurpose MUNIT's [2] network architectures for encoders E_A, E_B, decoders G_A, G_B and discriminators D_A, D_B, but remove styles from the encoders and decoders. *AdaIN* layers are replaced by instance normalization layers.

1.4. Translation

In each training iteration, two simulated views and one real image are sampled:

$$\begin{pmatrix} a_i^{tex} \\ a_i^{ref} \end{pmatrix}, \begin{pmatrix} a_j^{tex} \\ a_j^{ref} \end{pmatrix} \in A, \qquad b \in B \tag{13}$$

Equations 14 to 18 show all translations and reconstructions performed during each training step. Note that the second view j from domain A is only translated to domain B and not used for reconstructions or cycles.

Domain A:

$$\begin{pmatrix} a_i^{tex} \\ a_i^{ref} \end{pmatrix} \xrightarrow{E_A} c^{a_i} \xrightarrow{G_B} \hat{b}_i \xrightarrow{E_B} c^{a_i}_{rec} \xrightarrow{G_A} \begin{pmatrix} a_{i,cyc}^{tex} \\ a_{i,cyc}^{ref} \end{pmatrix} \tag{14}$$

$$\begin{pmatrix} a_i^{tex} \\ a_i^{ref} \end{pmatrix} \xrightarrow{G_A \circ E_A} \begin{pmatrix} a_{i,rec}^{tex} \\ a_{i,rec}^{ref} \end{pmatrix} \tag{15}$$

$$\begin{pmatrix} a_j^{tex} \\ a_j^{ref} \end{pmatrix} \xrightarrow{G_B \circ E_A} \hat{b}_j \tag{16}$$

Domain B:

$$b \xrightarrow{E_B} c^b \xrightarrow{G_A} \begin{pmatrix} \hat{a}^{tex} \\ \hat{a}^{ref} \end{pmatrix} \xrightarrow{E_A} c^b_{rec} \xrightarrow{G_B} b_{cyc} \tag{17}$$

$$b \xrightarrow{G_B \circ E_B} b_{rec} \tag{18}$$

1.5. Losses

Generator/texture update: Encoders E_A, E_B, decoders G_A, G_B and neural textures tex are learned by minimizing Equation 19.

$$L_{total} = L_{translation} + \lambda L_{vc} \tag{19}$$

$$L_{translation} = L_{adv} + L_{cyc} + L_{rec} + L_c + L_{ssim}, \tag{20}$$

$$L_{adv} = \left\| D_A \begin{pmatrix} \hat{a}_i^{tex} \\ \hat{a}_i^{ref} \end{pmatrix} - 1 \right\|_2^2 + \left\| D_B(\hat{b}) - 1 \right\|_2^2 \tag{21}$$

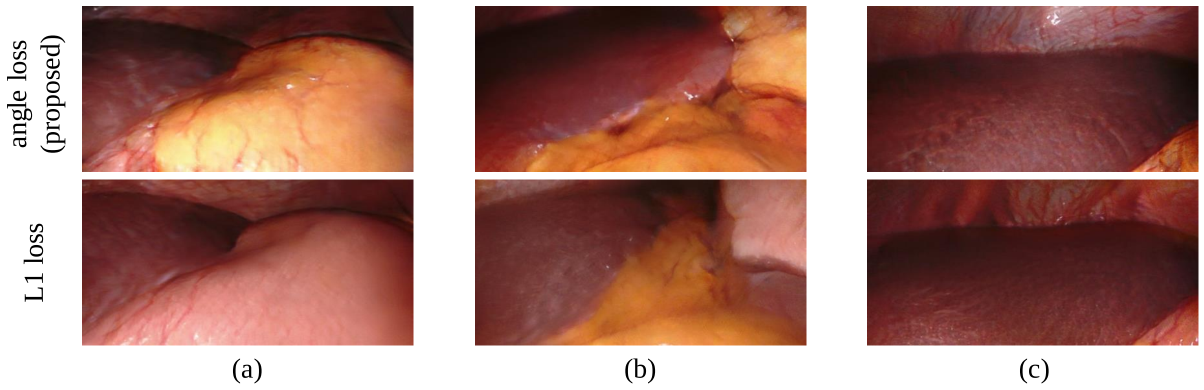
$$L_{cyc} = \lambda_{cyc} \left(\left\| \begin{pmatrix} a_i^{tex} - a_{i,cyc}^{tex} \\ a_i^{ref} - a_{i,cyc}^{ref} \end{pmatrix} \right\|_1 + \|b - b_{cyc}\|_1 \right) \tag{22}$$

$$L_{rec} = \lambda_{rec} \left(\left\| \begin{pmatrix} a_i^{tex} - a_{i,rec}^{tex} \\ a_i^{ref} - a_{i,rec}^{ref} \end{pmatrix} \right\|_1 + \|b - b_{rec}\|_1 \right) \tag{23}$$

$$L_c = \lambda_c (\|c^{a_i} - c^{a_i}_{rec}\|_1 + \|c^b - c^b_{rec}\|_1) \tag{24}$$

$$\begin{aligned} L_{ssim} =& \lambda_{ssim}^{ab} \text{MS-SSIM}(gray(a^{ref}), gray(\hat{b})) \\ &+ \lambda_{ssim}^{ba} \text{MS-SSIM}(gray(b), gray(\hat{a}^{ref})) \end{aligned} \tag{25}$$

$$L_{vc} = \frac{1}{|M_{\hat{b}_i \hat{b}_j}|} \sum_{(x,y)}^{M_{\hat{b}_i \hat{b}_j}} cos^{-1} \left(\frac{\hat{b}_i^{xy} \cdot w_i(\hat{b}_j)^{xy}}{\|\hat{b}_i^{xy}\| \|w_i(\hat{b}_j)^{xy}\|} \right), \tag{26}$$



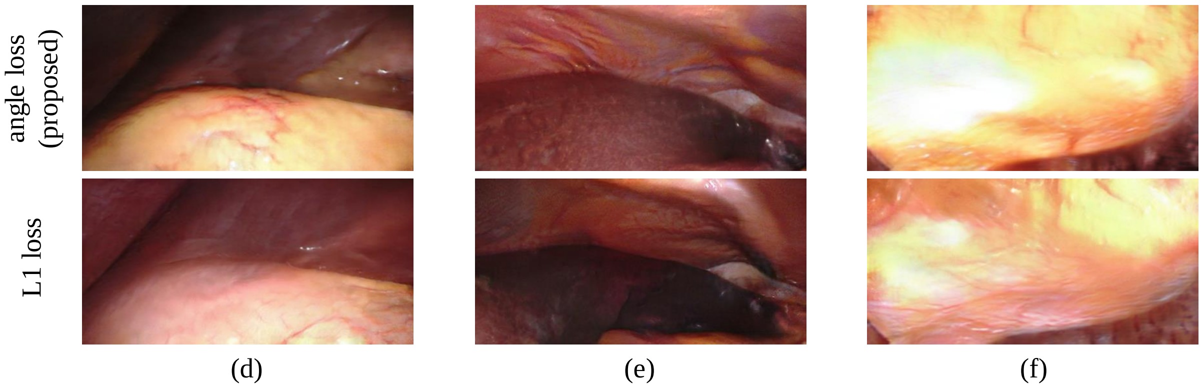


Figure 1. More examples of our proposed angle-based loss vs. an L1 loss. While the angle-based loss allows close and central parts of the image to be brightened, the L1 loss leads to either uniform brightness (*(a)-(d)*) or unrealistic lighting conditions (*(e)* and *(f)*).

We use $\lambda_{cyc} = 10$, $\lambda_{rec} = 10$, $\lambda_c = 1$, $\lambda^{ab}_{ssim} = 5$, $\lambda^{ba}_{ssim} = 3$. For the Multi-Scale Structural Similarity (MS-SSIM) loss, we use Jorge Pessoa's implementation[1] with 5 scales and a window size of 11×11. Similarity is only enforced on the brightness (gray values) of images. The weight for the view-consistency loss is initialized with $\lambda = 0$ and set to 20 after 10k iteration.

Discriminator update: Discriminator networks D_A, D_B are learned by minimizing:

$$L_{dis} = \left\| D_A \begin{pmatrix} a^{tex} \\ a^{ref} \end{pmatrix} - 1 \right\|_2^2 + \| D_B(b) - 1 \|_2^2 \\ \left\| D_A \begin{pmatrix} \hat{a}^{tex} \\ \hat{a}^{ref} \end{pmatrix} - 0 \right\|_2^2 + \left\| D_B(\hat{b}) - 0 \right\|_2^2 \quad (27)$$

1.6. Training

We train our model for 500,000 iterations and use the Adam optimizer with initial learning rates of 10^{-4} for network parameters and 10^{-3} for neural textures. Learning rates are halved every 100,000 iterations. For all experiments including ablations and baselines, we use a batch size of 1. Note, however, that the definition of a batch varies across methods. In our method we *e.g.* sample one real and two simulated views per batch. In *SSIM-MUNIT* only one image from each domain is sampled per batch, while for ReCycle or SSIM-Recycle 3 images for each domain are sampled.

[1] https://github.com/jorge-pessoa/pytorch-msssim

1.7. Baselines

SSIM-MUNIT: This model is trained for 370k iterations as in the original paper. For realism and label preservation experiments, we translate each training image once with randomly drawn styles and once with a style extracted from randomly drawn images. Test video sequences are translated with a randomly drawn but fixed style to guarantee a fair comparison with respect to temporal consistency.

ReCycle & SSIM-ReCycle: Both models are trained for the suggested 40 epochs. Longer training times lead to degrading performance. Triplets of simulated images are obtained through random, linear trajectories comparable to the motion in real sequences.

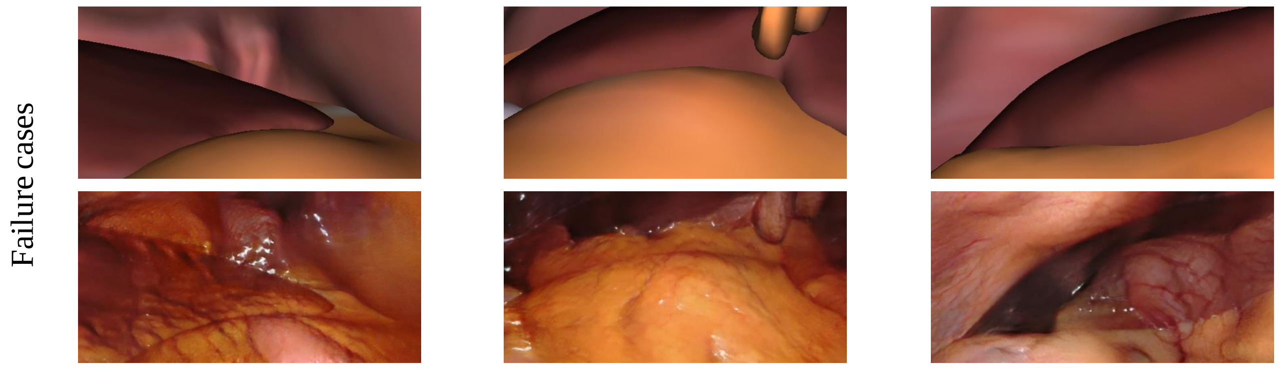


Figure 2. Common failure cases include the rendering of fat or stomach texture on areas which are supposed to be liver (in the simulated scene).

OF-UNIT: We train for 1 Mio. iterations due to the stronger correlation between subsequent samples. We replace the view-consistency loss with an $L1$ loss and increase the weight λ to 50 since it gave better results. Intuitively, the angle loss is not required here since the views on which the loss is applied are consecutive frames and exhibit only minor lighting changes. Synthetic sequences are obtained from the triplets used for ReCycle.

2. Additional Results

Figure 1 shows more translated views comparing our proposed angle-based view-consistency loss to a naïve $L1$ loss. Our proposed loss allows for view-dependent effects while an $L1$ loss actively discourages them.

3. Limitations

Realism of 3D Shapes The largest limitation of our current method is the lacking realism of the simulated 3D meshes. While liver meshes are obtained from real CT scans, all other organs were designed manually. Especially liver ligaments are difficult to model realistically. However, even the liver meshes often differ from realistic settings since intra-operative deformations (*e.g*. by inflating the abdominal cavity) are different from ones observed during a CT scan. We manually deformed the liver meshes to resemble intra-operative shapes.

Freedom of Neural Textures Figure 2 shows failure cases of our method. Sometimes the model misinterprets objects in the simulated scene and *e.g*. renders fat or stomach textures on liver surfaces. Compared to previous work, this seems to happen slightly more frequently as indicated by our liver-segmentation experiments. We believe that neural textures give the model more freedom to interpret the simulated scene. Hence, enabling a higher level of detail and consistency comes with the trade-off of more misinterpretations.

Variability We propose a deterministic, style-less model since we believe that current state of the art for style-dependent translation (AdaIN [1]) is unsuitable and not theoretically sound in the video setting. To demonstrate this, we implement a variant of our model with AdaIN styles as used in MUNIT [2] or SSIM-MUNIT [3].

Firstly, styles operate at image level and do not introduce variability at texture level; *e.g*. locations of vessels do not vary, but only their appearance (Fig. 3). Hence, the variability is mostly limited to color changes but changing textures would be highly useful for creating diverse training and evaluation environments.

Secondly, AdaIN styles directly control lower-order statistics (mean, variance) of feature distributions and thereby enforce how much fat, blood, etc. are rendered in a frame[2]. So, using a single AdaIN style for the whole video enforces a *static* feature distribution (*i.e*. static amount of fat, etc.) across frames although the field of view changes over time. Figure 4 shows how *e.g*. a fatty style image results in translated views with a lot of yellow color regardless of its content. Thus, even if the current view contains only liver, AdaIN styles force the model to render fat. In the image setting, this problem could possibly be circumvented by selecting style images with similar content. In the video setting, however, finding a style that matches all video frames cannot be guaranteed and using multiple styles induces temporal inconsistencies.

For both problems, we believe that introducing styles at texture level would provide a possible solution and that this would be a useful direction for future research.

[2]Fig. 3 of SSIM-MUNIT's supplementary `http://opencas.dkfz.de/image2image/supplementary.pdf`

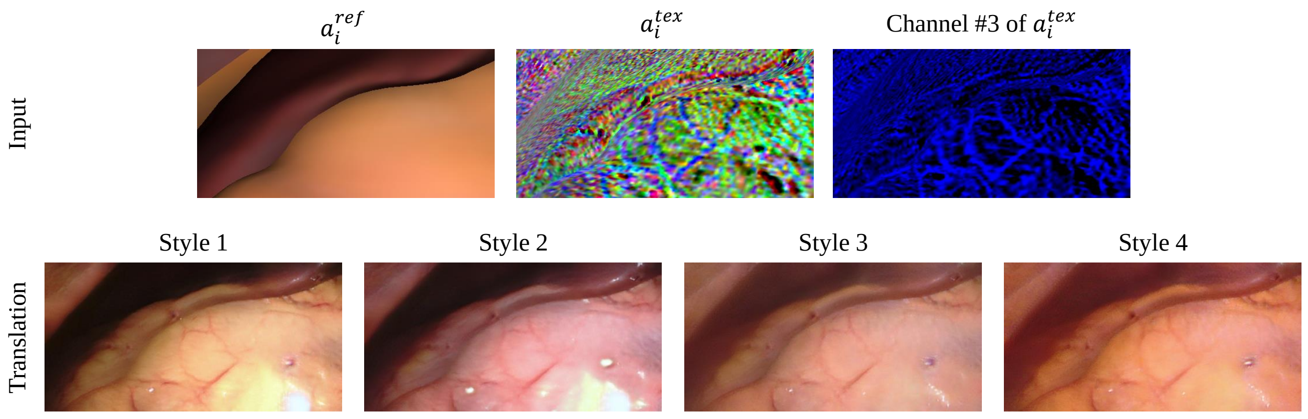


Figure 3. Styles operate only at image-level but details such as vessels are stored in the neural textures (see *Channel #3 of* a_i^{tex}). Hence, the resulting variability is mostly restricted to color changes. It can be seen that vessels have identical locations in all samples. We believe research towards styles at texture-level would be a useful direction for future work.

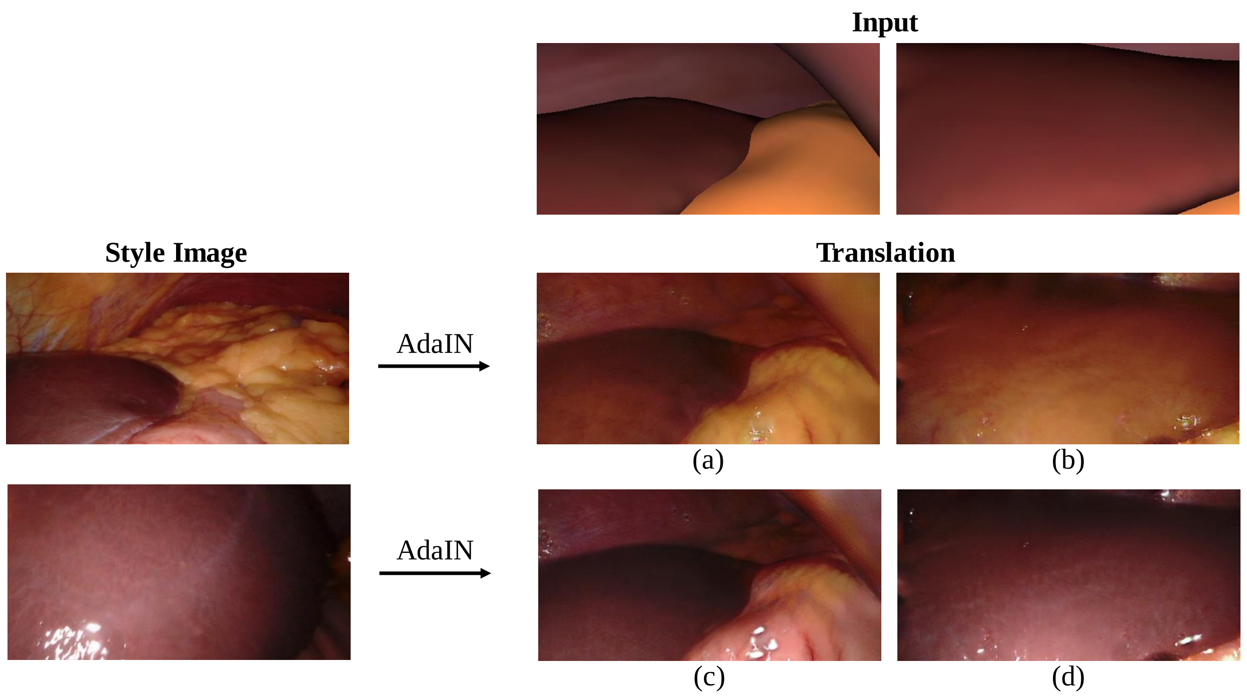


Figure 4. In the video setting, *AdaIN* enforces static feature distributions even though the field of view changes over time. We translate two views of the same scene once with a style image of similar content (a and d) and once with dissimilar content (b and c). We observe that using a style image with a lot of fat forces the model to render yellow color in a view that contains mostly liver (b) but produces reasonable results if the content matches (a). We did, however, observe that this effects seems to be most prominent with 'fatty' style images and less drastic in some other cases. Example c shows that the model is able to draw stomach texture on the lower right corner although there seems to be no such texture in the style image. Possibly, the similar hue of liver and stomach allow for this.

References

[1] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1501–1510, 2017. 4

[2] Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. Multimodal unsupervised image-to-image translation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 172–189, 2018. 2, 4

[3] Micha Pfeiffer, Isabel Funke, Maria R Robu, Sebastian Bo-

denstedt, Leon Strenger, Sandy Engelhardt, Tobias Roß, Matthew J Clarkson, Kurinchi Gurusamy, Brian R Davidson, et al. Generating large labeled data sets for laparoscopic image processing tasks using unpaired image-to-image translation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 119–127. Springer, 2019. 4