Supplementary Material

Generating large labeled data sets for laparoscopic image processing tasks using unpaired image-to-image translation

I. Architecture

Our framework builds on the MUNIT framework. Fig. 1 shows how the cycle consistency works in the original MUNIT framework. Fig. 2 explains how latent codes are reconstructed. An overview of all used symbols and formulas, as well as the difference between MUNIT and our framework can be found in Table I.

Note: In the article, we describe the encoders E_A and E_B and say that they encode an image to style *and* content. Here, we slightly change that nomenclature and split each of the two encoders into style and content encoders, i.e. E_A is split up into E_{sA} and E_{cA} (and E_B into E_{sB} and E_{cB}). This is done to be able to describe the processes in more detail here while keeping the original article concise.

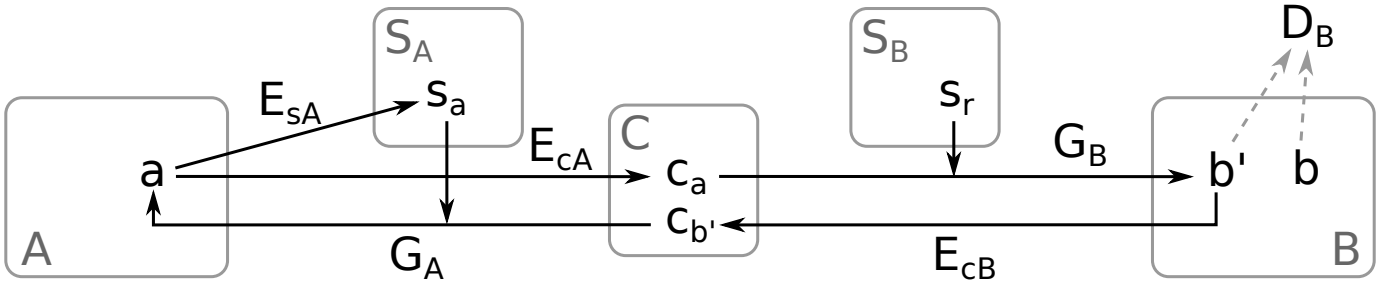

Fig. 1: Cycle consistency in the MUNIT framework. An image a drawn from A is encoded to a style code s_a and content code c_a by the style encoder E_{sA} and content encoder E_{cA}, respectively. The content code is then translated to B using a randomly drawn style vector s_r, resulting in a translated image b'. The image is then re-encoded to the content again, and finally the original image is reconstructed using the style s_a which was previously extracted. The Mean Absolute Error is used to force the reconstruction to correspond to the originally drawn image a. At the same time, a discriminator D_B attempts to distinguish the translation result from original images drawn from B. The whole process is repeated in the opposite direction for an image drawn from B. The cycle consistency, together with the discriminators, is responsible for the ability of the networks to learn unpaired image-to-image translations. By injecting different random style vectors, MUNIT enables multi-modal translation. Our implementation uses no style in A but is otherwise the same (please compare with Fig. 2 in the original article). The cycle consistency is also described in Table I in the ”Cycle consistency” row.

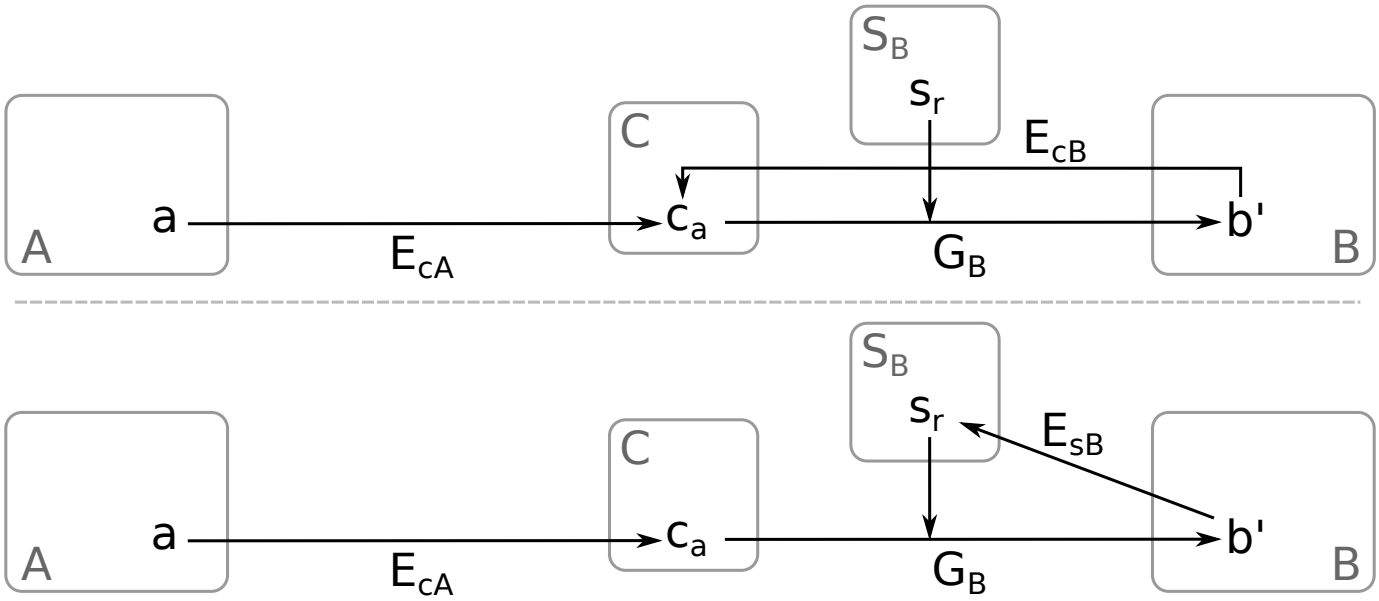

Fig. 2: Latent code reconstruction in the MUNIT framework. Top: Content code reconstruction. First, an image a is encoded to a content code c_a. This is then translated to an image b' using a random style vector s_r. The content code reconstruction states that re-encoding this translated image with E_{cB} should yield the original content code c_a. Bottom: Style code reconstruction. Again, an image a is encoded to a content code c_a and translated to an image b' using a random style code s_r. The style code reconstruction states that re-encoding this translated image with E_{sB} should yield the original style code s_r. Both reconstructions are also run in the same way in the opposite direction for images drawn from B. These reconstruction losses ensure the correct extraction of styles and content from images, since the networks are forced to use the latent codes in image translation and then extract them again from the translated images. Our framework uses the same method, except that we do not have style reconstruction in A (as there is no style).

TABLE I: Overview of symbols and formulas and how they differ between MUNIT and our work. Note that the main difference is that we do not have a style in A. Also, we have the additional MS-SSIM loss acting on the image brightness. For reconstruction losses, the mean absolute error (MAE) is used. Style vectors s_a and s_b can be extracted from images using the encoders or, alternatively, sampled randomly (s_r). The overall loss is computed by adding all losses together.

MUNIT	Ours	Comment
A	A	First image domain. In our case, these are renderings of a simulated laparoscopic scene.
B	B	Second image domain. In our case, these are laparoscopic images from Cholec80.
a, b	a, b	Input images drawn from A and B respectively.
$c_a = E_{cA}(a)$ $c_b = E_{cB}(b)$	$c_a = E_{cA}(a)$ $c_b = E_{cB}(b)$	Content encoders E_{cA} and E_{cB} encode images into latent content codes c_a and c_b
$s_a = E_{sA}(a)$ $s_b = E_{sB}(b)$	$s_b = E_{sB}(b)$	Style encoders extract style codes from images. We have no style in A, so we have no encoder E_{sA}.
$b' = G_B(c_a, s_b)$ $a' = G_A(c_b, s_a)$	$b' = G_B(c_a, s_b)$ $a' = G_A(c_b)$	Generator networks generate images given content and style codes
D_A D_B	D_A D_B	Discriminators. D_A learns to differentiate between images a drawn from A and images a' generated by G_A. D_B works analogously. (GAN-Loss)
$a \approx G_A(E_{cB}(b'), s_a)$ $b \approx G_B(E_{cA}(a'), s_b)$	$a \approx G_A(E_{cB}(b'))$ $b \approx G_B(E_{cA}(a'), s_b)$	Cycle consistency (MAE-Loss)
$a \approx G_A(c_a, s_a)$ $b \approx G_B(c_b, s_b)$	$a \approx G_A(c_a)$ $b \approx G_B(c_b, s_b)$	Same domain image reconstruction (MAE-Loss)
$c_b \approx E_{cA}(G_A(c_b, s_r))$ $c_a \approx E_{cB}(G_B(c_a, s_r))$	$c_b \approx E_{cA}(G_A(c_b))$ $c_a \approx E_{cB}(G_B(c_a, s_r))$	Content reconstruction, using random style vectors s_r. (MAE-Loss)
$s_r \approx E_{sA}(G_A(c_b, s_r))$ $s_r \approx E_{sB}(G_B(c_a, s_r))$	$s_r \approx E_{sB}(G_B(c_a, s_r))$	Style reconstruction (MAE-Loss)
	$\mu(a') \approx \mu(b)$ $\mu(b') \approx \mu(a)$	Brightness consistency during translation. $\mu(x)$ calculates the brightness of the image x (average of red, green and blue channels for every pixel). Enforced by the multi-scale structural similarity loss (MS-SSIM-Loss).

II. Further results

After training the translation networks, either the style from real images or random style vectors can be used during translation from A to B. In Fig. 3, we show some results using both real and random style vectors. Fig. 4 shows further samples of the final image translation process where we use random style vectors.

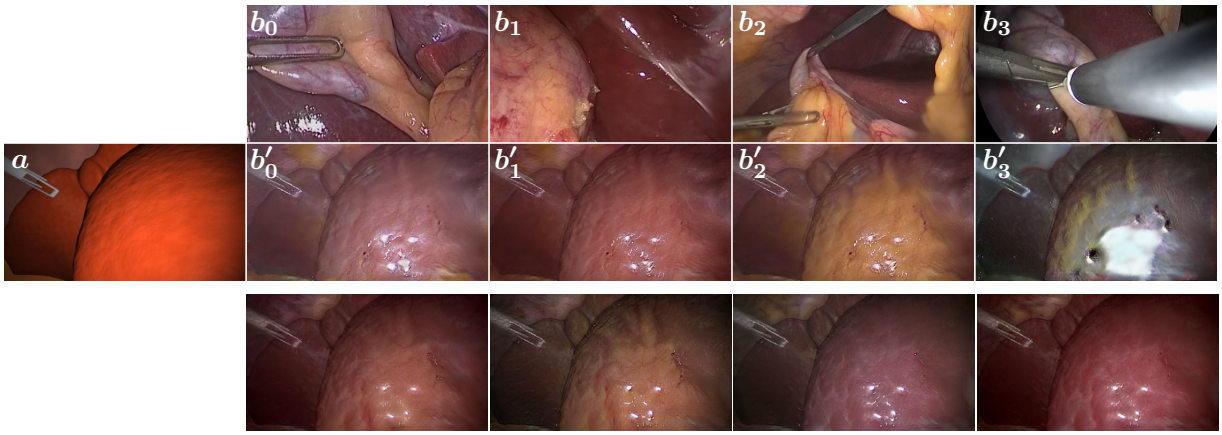

Fig. 3: Difference between using real styles and random styles during translation. Here, the styles from various images (top row, b_0 to b_3) are used to translate input image a to various result images (center row, b'_0 to b'_3). While b'_0 and b'_1 show decent results, b'_2 and b'_3 have clear artifacts. These can be attributed to the fact that the content in the corresponding style images differs too much from the content in the input image a: b_2 shows little liver surface and lots of fat tissue and a large portion of b_3 shows only the tool. Similar issues can arise when using random styles (bottom row), but we have found empirically that this is less likely to happen. In this work, we chose to only use random styles in translation, since choosing good style images from data sets B – in which the viewpoint and image content varies a lot – is far from trivial.

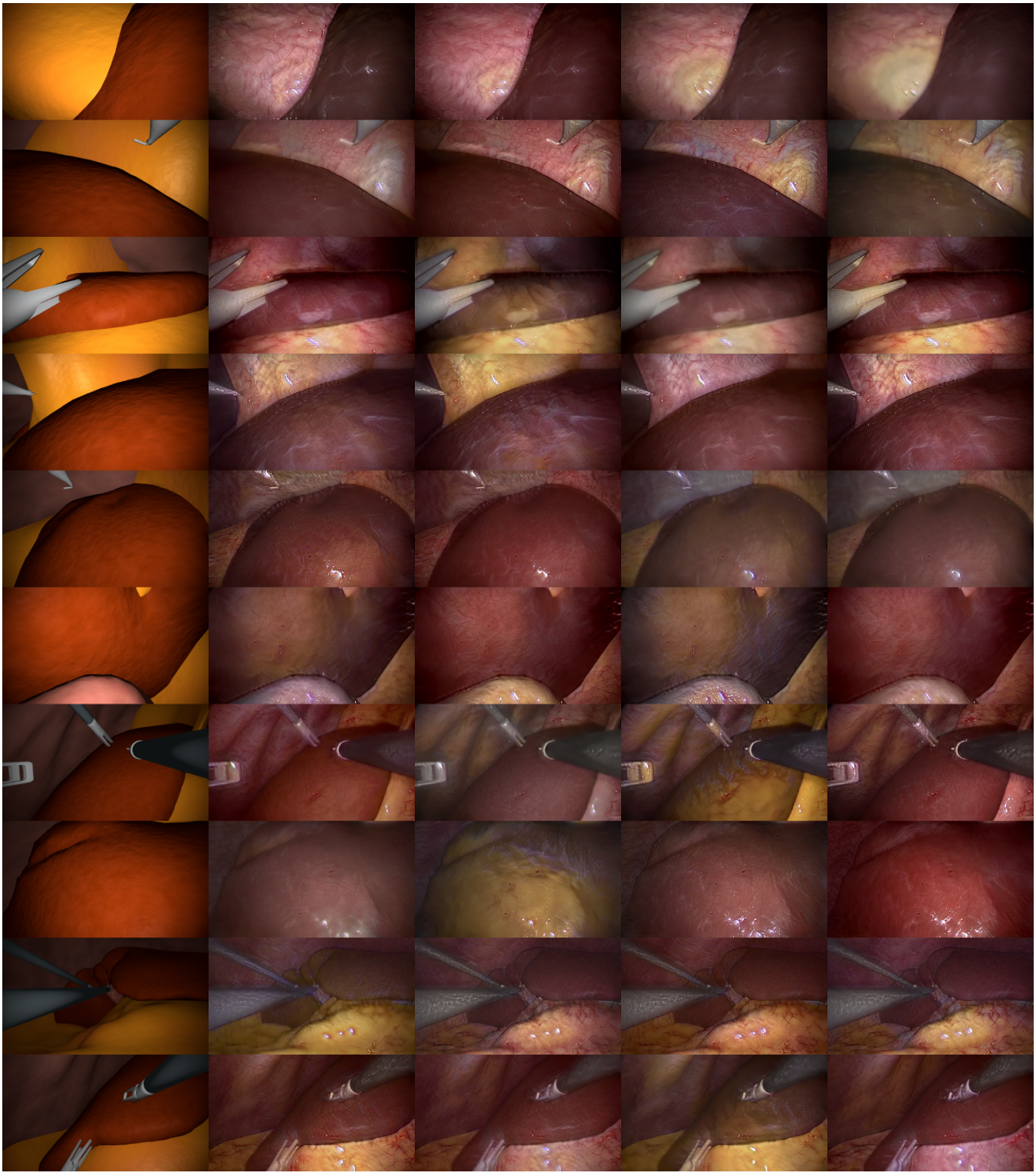

Fig. 4: Translation results with random style vectors. Each input image (left column) is translated with four, randomly drawn style vectors. Input as well as output images are 452×256 pixels in size. Sometimes, the translation networks use very similar textures for fat and gallbladder tissues, which is likely due to the fact that these are often also hard to distinguish in real images. During translation, image structure is preserved for all images, independent of image content and camera viewpoint. The MS-SSIM loss keeps the networks from adding more structures, limiting the freedom the networks have to make the result images look more realistic. Adding more detail to the input images (connective tissues, deformation) would likely enhance realism in the output. Since the ultimate goal of the project was to keep the manual workload to a minimum, we refrained from doing so.