Pattern Recognition Letters 155 (2022) 69–76

Contents lists available at ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

InfraGAN: A GAN architecture to transfer visible images to infrared domain

Mehmet Akif Özkanoğlu[a,b], Sedat Ozer[a,*]

[a] Ozer Lab, Dept. of Computer Science, Ozyegin University, Turkey
[b] Ozer Lab, Dept. of Computer Science, Bilkent University, Turkey

ARTICLE INFO

Article history:
Received 14 July 2021
Revised 31 December 2021
Accepted 29 January 2022
Available online 3 February 2022

Edited by: Jiwen Lu

Keywords:
Domain transfer
GANs
Infrared image generation

ABSTRACT

Utilizing both visible and infrared (IR) images in various deep learning based computer vision tasks has been a recent trend. Consequently, datasets having both visible and IR image pairs are desired in many applications. However, while large image datasets taken at the visible spectrum can be found in many domains, large IR-based datasets are not easily available in many domains. The lack of IR counterparts of the available visible image datasets limits existing deep algorithms to perform on IR images effectively. In this paper, to overcome with that challenge, we introduce a generative adversarial network (GAN) based solution and generate the IR equivalent of a given visible image by training our deep network to learn the relation between visible and IR modalities. In our proposed GAN architecture (InfraGAN), we introduce using structural similarity as an additional loss function. Furthermore, in our discriminator, we do not only consider the entire image being fake or real but also each pixel being fake or real. We evaluate our comparative results on three different datasets and report the state of the art results over five metrics when compared to Pix2Pix and ThermalGAN architectures from the literature. We report up to +16% better performance in Structural Similarity Index Measure (SSIM) over Pix2Pix and +8% better performance over ThermalGAN for VEDAI dataset. Further gains on different metrics and on different datasets are also reported in our experiments section.

1. Introduction

Recent developments in sensor types and the increasing demand in autonomous systems, (as in [1–3]), have moved many researchers' attention to the data fusion algorithms where using more than one modality is desired. Sample areas and works can be found in [4–7]. Many autonomous systems use images coming from both visible and infrared (IR) spectrum. Since both IR image (typically containing information from the thermal spectrum) and visible image (containing information from the visible spectrum) summarize different properties of a scene, acquired IR and visible images of the same scene would look similar but not the same. A typical example is text images. For example, reading the text information on a posted sign might be an easy task in a visible image, however, that can easily become a troublesome task if the reading is done in the IR spectrum, since the text might not appear on the IR image.

Infrared images have primary importance for various vision tasks such as object detection and object tracking especially in the night mode. Infrared cameras have the ability to distinguish an object in difficult lighting conditions such as in night or in cloudy weather. On the other hand, because of perceiving the thermal temperature level, IR images can also help to differentiate objects based on their body temperature, i.e., they provide extra information in addition to the visible spectrum. As a result, IR images can help to improve the understanding of the surroundings and making miscellaneous vision tasks such as detection, tracking, and self-driving more feasible.

Making a deep *fusion* algorithm to work better than the existing single-modality based algorithms, usually a large data set which is obtained by multiple sensors is needed. However, the typical trend is using single-modality based large public datasets in many vision and pattern recognition applications since major large public datasets (such as ImageNet [8], PASCAL VOC [9] or MS COCO [10]) contain images from only the visible spectrum. That creates a problem to use such datasets in multi-sensor applications since they provide only visible images. Alternative solutions to still using such existing public datasets in fusion algorithms include collecting new datasets with multiple sensors or using domain transfer based techniques to obtain (generate) the missing

* Corresponding author.
E-mail address: sedat.ozer@ozyegin.edu.tr (S. Ozer).

https://doi.org/10.1016/j.patrec.2022.01.026
0167-8655/

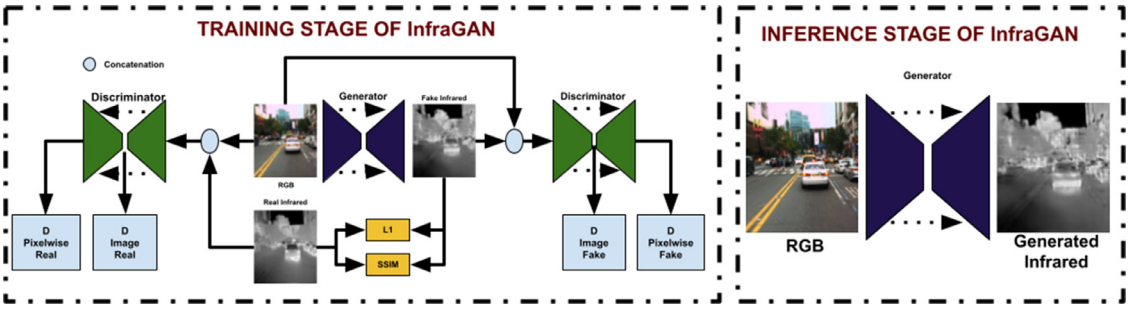


Fig. 1. Overview of our proposed InfraGAN architecture. Both discriminators located on leftmost and rightmost side are identical in the training stage. The details of our generator are given in Fig. 2 and the details of our discriminator are given in Fig. 3.

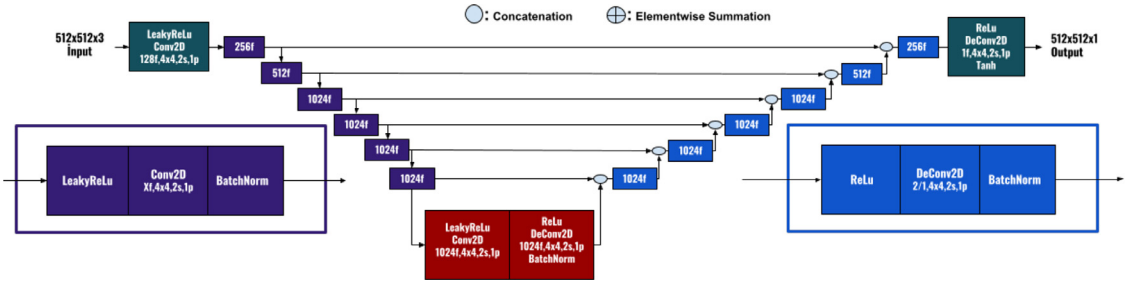
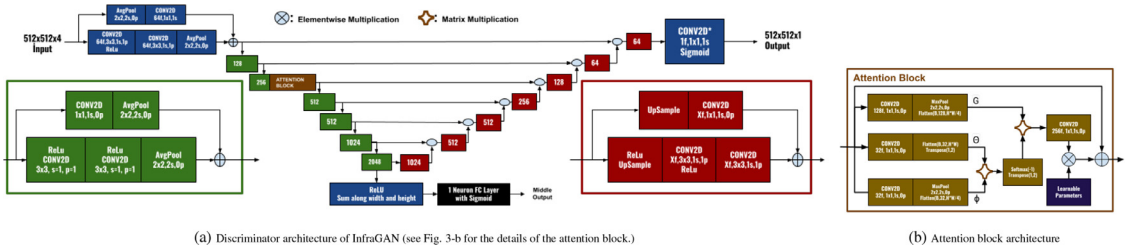


Fig. 2. Our U-Net based generator architecture. In the figure, Xf is used to represent the total number of filters in each block where X is the total number of filters.



(a) Discriminator architecture of InfraGAN (see Fig. 3-b for the details of the attention block.) (b) Attention block architecture

Fig. 3. Our U-Net based discriminator architecture. (a) shows the discriminator architecture where the symbol ⊕ means element-wise summation and the symbol O means concatenation. (b) shows the attention block architecture as used in the discriminator.

modalities. In this paper, we look for an efficient transfer technique to specifically obtain the IR equivalent of a given visible image.

In this paper, we propose a generative adversarial networks (GAN) based deep architecture to generate the IR equivalent image of a given (input) visible image to help facilitate large and single modality based datasets in multi-sensor based applications. We call our solution *InfraGAN*. InfraGAN uses Structural Similarity Index Measure (SSIM) in its architecture to focus on learning certain structural similarities between the IR and visible domains while pixel-based L1 norm enforces the architecture to look like an IR image. The overview of our architecture is given in Fig. 1. We compare and evaluate our proposed architecture's performance on three different datasets: on one aerial dataset and on two ground taken datasets (having both visible and IR pairs). We use five metrics to compare our algorithm's performance to two other baseline networks. Our experimental and comparative results show that our proposed architecture generates better IR equivalents when compared to both recently proposed ThermalGAN [11] and Pix2Pix [12] architectures in all of those five metrics.

In this paper, our contributions include:

1. introduction of a GAN-based domain-transfer technique to obtain the IR equivalent of a given visible image where *both* generator and discriminator parts use an encoder-decoder architecture;
2. use of an additional loss term based on SSIM, (see Eq. (8)) for improved results;
3. experiments on three different large benchmark data sets.

2. Related work

Generating infrared images from a given visible image has not been widely studied in the recent literature and there are only a few published earlier works available (as in [11,13–15]) reporting statistical and comparative results. There are also a relevant class of works that focus on fusing both IR and visible images to obtain a single fused images as in [16]. However, our line of work differs from such fusion-related literature as those works take both IR and visible image pairs as input, while we take only visible image as input and generate its IR equivalent as output. Most of those relevant literature to generate IR images introduces GAN based architectures as in [11,14,15]. Among those, ThermalGAN [11] uses a U-Net based architecture in its generator and uses a standard convolutional network (a basic discriminator) in its discriminator to classify the entire image being fake or real. In that work, the authors used ThermalWorld dataset [11] which consists of 5098 of visible and thermal image pairs along with their segmentation annotations for ten classes: truck, car, van, person, boat, bus, cat, building, tram, dog. In another work [14], the authors generated IR images to track objects. There, the authors studied the performance of Cycle-GAN [17] and Pix2Pix [12] based networks for generating IR images. They used both paired and unpaired datasets in their experiments. In [15], the authors introduced using multiple generators. Each generator would focus on learning the properties of a different scene. Another ResNet [18] based classifier was used to choose which generators' out-

put would be most suitable for the given input image. They used a dataset[1] containing 40479 image pairs.

Our work differs from the above-mentioned works by introducing a novel GAN based solution where our discriminator uses an encoder-decoder based architecture and our generator uses an additional loss term based on SSIM in that architecture. Furthermore, while there are not many works in the literature presenting statistical results on large datasets, we utilize three different datasets in our experiments to compare our proposed solution to two recently proposed GAN-based architectures from the literature (Pix2Pix and ThermalGAN), in addition to comparing our results to simple U-Net architecture.

The main differences of our network from the most relevant literature include: (i) the use of pixel-based loss using SSIM in the loss function and (ii) the inclusion of a decoder part in the discriminator architecture using residual blocks for the generation of IR images.

3. Our proposed architecture: InfraGAN

GANs basically consist of a generator and a discriminator network where each of those networks try to win over the other network in an adversarial manner [19]. The goal of the discriminator is to determine whether the input image is real or fake by discriminating the output of the generator from the real ones, and the goal of the generator is to produce fake images imitating the real ones so that it can fool the discriminator.

Our network uses two distinct U-Net based architectures. The first one is used in our generator, inspired from [11], (see Fig. 2) and the second one is used in our discriminator (see Fig. 3) which is similar to the U-Net architecture in [20]. In both figures, we use block structures in encoder and decoder sides of the networks and each of those blocks are color coded based on their type. In contrast to the generator network, discriminator's upsampling layer is composed of bi-linear interpolation with scale rate 2 as shown in Fig. 3. Furthermore, its down sampling layer is made up of average pooling with stride 2 to downscale the input into 1/2 resolution. In the figures, s is used for the stride value, f for the number of used filters, p for the padding value, 1×1, 2×2, 3×3 and 4×4 are the kernel sizes.

Generator Architecture: Our generator downscales images by using 2D convolutional layers with stride 2 and uses deconvolutional layers without skip connections (instead of using interpolation). These layers manage to increase quality of IR images in generator block thanks to learned parameters. On the other hand, the input image is normalized between 1 and -1 in our network and therefore, the generator uses tangent hyperbolic activation function at the end to generate pixel values within the same input range.

Discriminator Architecture: In general, the discriminator architectures (based on the output format) can be classified under two main types: basic discriminators that classify only the entire input image being fake or real, and pixel-based discriminators that can classify the entire (input) image being fake or real as well as its individual pixels being fake or real. See Fig. 4 for illustrative comparisons of those two types. Recent literature in [20] reports that utilizing a U-Net architecture in discriminator (by classifying each pixel being fake or real) would improve the performance of overall GAN architecture. Inspired from that work, we use an architecture allowing us to classify each pixel separately in our discriminator to generate IR images. In our discriminator, we used bilinear interpolation for upsampling stages and maxpooling layer for downsampling stages to limit the number of learnable parameters, since, using upsampling and maxpooling functions reduce the required memory consumption when compared to the convolutional and deconvolutional layers (deconv with scale rate 2 and conv2d with stride 2). 1 neuron FCN is used as a normalization layer to limit the output values being between 0 and 1. Furthermore, a single attention block is used (see Fig. 3 for the details of the attention block).

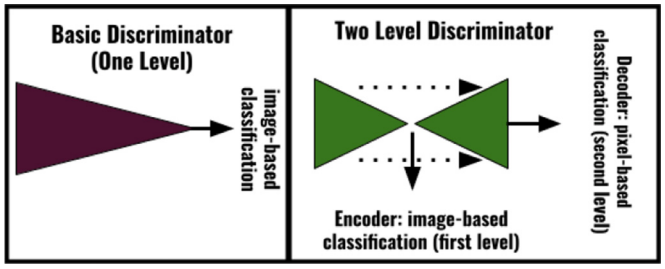


Fig. 4. Comparison of two discriminator types: one-level (on the left) and two-level discriminators (on the right). One-level discriminator only classifies the entire image while the two-level one classifies the image and its pixels as being fake or real at the same time.

A residual block is a collection of multiple layers with skip connection(s). In our discriminator, we utilize two different residual block types. The first residual block type is used to downscale its input (green boxes in Fig. 3), while the second type is used to upscale its input (the red boxes in Fig. 3). Both residual blocks include, 2D convolutional layers with 3×3, 1×1 kernel size, and same padding. The red block upscales its input via interpolation before applying convolutional layers. The green block down-scales its input via average pooling after applying convolutional layers. At the end, both blocks perform channel-wise summation on two parallel branches and pass the output to next block. Next we describe the used loss functions.

Discriminator Loss: Our discriminator uses an encoder-decoder architecture and contains two loss terms as given in Eq. (1). One of those terms focuses on the entire image being fake or real (see Eq. (2)), and the other one focuses on each pixel being fake or real (see Eq. (3)).

$$Loss_{D^U} = Loss_{D^U_{enc}} + Loss_{D^U_{dec}} \tag{1}$$

where $Loss_{D^U_{enc}}$ is the loss obtained at the end of the encoder for image based loss, and $Loss_{D^U_{dec}}$ is the loss at the end of the decoder architecture for pixel based loss. Their definitions are given below:

$$Loss_{D^U_{enc}} = -\mathbb{E}_{x,y}[\log D^U_{enc}(X,Y)] - \mathbb{E}_x[\log(1 - D^U_{enc}(X, G(X)))] \tag{2}$$

$$Loss_{D^U_{dec}} = -\mathbb{E}_{x,y}\left[\sum_{i,j} \log[D^U_{dec}(X,Y)]_{i,j}\right] - \mathbb{E}_x\left[\sum_{i,j} \log(1 - [D^U_{dec}(X, G(X))]_{i,j})\right] \tag{3}$$

where X is the input image in visible domain, Y the ground truth IR image, $E(.)$ refers to the expected value, $G(X)$ the generator's IR output, $D(X,Y)$ and $D(X,G(Y))$ are the binary outputs of the discriminator, (i,j) refers to a pixel's coordinate in both images (where both input and output has the same dimensions). The discriminator's decoder output yields probability of each pixel being real or fake separately.

Generator Loss: Our generator uses three different terms in its loss ($Loss_G$). The first term is the standard conditional GAN loss ($Loss_{cGAN}$). Additionally, we use two more terms: structural similarity index (SSIM) based loss ($Loss_{SSIM}$) and L1 loss ($Loss_{L1}$). The reason for us to add the additional $Loss_{SSIM}$ is to ensure that the

[1] https://www.kaggle.com/c/planet-understanding-the-amazon-from-space.

generated output will look similar to the given input image structurally.

$$Loss_G = Loss_{cGAN} + \lambda_1 * Loss_{L1} + \lambda_2 * Loss_{SSIM} \quad (4)$$

where λ_1 and λ_2 are the hyperparameters for loss function of generator network, and each used term is defined below. For a single image, the $Loss_{L1}$ is defined as:

$$Loss_{L1} = \frac{1}{N}\sum_{i,j}|G(X)_{i,j} - Y_{i,j}| \quad (5)$$

where $G(X)_{i,j}$ is the pixel value of the generated image at (i, j) and N is the total number of pixels in the image. $Y_{i,j}$ is the pixel value of ground truth (real) IR image at (i, j). L_1 is the absolute value of the difference between two pixels.

$$Loss_{cGAN} = -\mathbb{E}_x\left[\sum_{i,j}\log([D^U_{dec}(X, G(X))]_{i,j})\right] - \mathbb{E}_x[\log(D^U_{enc}(X, G(X)))] \quad (6)$$

$$Loss_{SSIM} = \frac{1}{m}\sum_{i=0}^{m-1}(1 - SSIM(G(X_i), Y_i)) \quad (7)$$

where m is the batch-size. The Structural Similarity Index (SSIM) is a metric measuring the similarity between two images: Y and $G(X)$. SSIM is defined as follows [21]:

$$SSIM(Y', Y) = \frac{2\mu_{y'}\mu_y + C_1}{\mu^2_{y'} + \mu^2_y + C_1}\cdot\frac{2\sigma_{y'}\sigma_y + C_2}{\sigma^2_{y'} + \sigma^2_y + C_2}\cdot\frac{\sigma_{y'y} + C_3}{\sigma_{y'}\sigma_y + C_3} = \frac{2\mu_{y'}\mu_y + C_1}{\mu^2_{y'} + \mu^2_y + C_1}\cdot\frac{\sigma_{y'y} + C_3}{\sigma^2_{y'} + \sigma^2_y + C_2} \quad (8)$$

where C_1, C_2 and C_3 are the parameters to ensure the stability of divisions [22]. We set $C_3 = C_2/2$ while $C_1 = (0.01 * L)^2$ and $C_2 = (0.03 * L)^2$ where L is the range of the image pixels. As described above, Y' is the generated output (IR) image $G(X)$ while Y is the ground truth (IR image). $\sigma_{y'}$ and σ_y are modified standard deviations for Y' and Y; $\mu_{y'}$ and μ_y are the modified mean values of their respective images. Finally, $\sigma_{y'y}$ refers to modified co-variance. Further details on computing SSIM can be found in [22].

By using Eq. (6), the objective function of conditional GAN loss (cGAN) forces $D^U(X, G(X))$ to maximize in order to deceive the discriminator. That means: the higher realness score we obtained from $D^U(X, G(X))$, the more realistic image the generator can produce. The generator's loss is computed as if the generated images are classified as real by the discriminator.

4. Experiments

In this section, we first describe the used metrics and datasets and then present our experimental results. All our experiments were running on Intel(R) Core(TM) i9-10920X CPU @ 3.50GHz with 2 Titan RTX (24GB) GPUs.

4.1. Datasets and used metrics

We utilized three different datasets containing both IR and visible image pairs: VEDAI [23], FLIR[2] and KAIST [24] datasets. VEDAI [23] is a dataset for vehicle detection in aerial imagery. It contains 1268 image pairs (visible and IR images). We use 1068 pairs of those for training and use the remaining 200 pairs for testing. FLIR is another dataset containing visible-thermal image pairs. We use 12795 image pairs for training and 839 pairs for testing in FLIR. KAIST [24] is another dataset that provides 11 video sequences in different circumstances such as in night mode and in sunny mode. In our experiments, we use 12538 image pairs for training and 2252 image pairs for testing from the KAIST dataset. While in both VEDAI and KAIST datasets, the visible image comes in RGB format, in FLIR dataset the visible image comes in grayscale (single channel) format. In all of our networks, we use Adam optimization with learning rate of generator being 2.10^{-4} and learning rate of discriminator being 2.10^{-6}. Batch-size is set to 8. All networks are trained from scratch with random initialization. We set $\lambda_1 = 100$ and $\lambda_2 = 100$ in our experiments.

We evaluate the performance of each network over five different metrics including Structural Similarity Index Measure (SSIM, see Eq. (8)), Mean Structural Similarity Index Measure (MSSIM), Learned Perceptual Image Patch Similarity (LPIPS), L1 (pixel by pixel, see Eq. (5)) norm and Peak Signal-to-Noise Ratio (PSNR). Those metrics are briefly explained below.

$$MSSIM(Y', Y) = \frac{1}{K}\sum_{y'_i, y_i}^{K} SSIM(y'_i, y_i) \quad (9)$$

MSSIM is the mean SSIM value over the downscaled versions of the images [22] where y'_i and y_i are downscaled versions of generated (Y') and real (Y) infrared images. K is the number of downscaling factors (we used $2^1, 2^2, 2^3, 2^4, 2^5$ values for K=5 in this work).

Learned Perceptual Image Patch Similarity (LPIPS) [25] measures the Euclidian distance between the feature vectors of both images. To compute the metric, the comparative features are obtained from a CNN-based backbone pretrained on ImageNet [8] (in our experiments we used AlexNet [26]). The work in [25] reported that LPIPS manages to favorably evaluate closeness rate between target patch and reference patch in terms of human judgment compared to other traditional metrics such as SSIM and PSNR by using deep networks and their deep features.

$$LPIPS(Y', Y) = \sum_{l}^{L}\frac{1}{H_l W_l}\sum_{h_l, w_l}[f_l(Y')_{h_l, w_l} - f_l(Y)_{h_l, w_l}]^2 * \omega_l \quad (10)$$

where $f_l(Y'), f_l(Y) \in \mathbb{R}^{H_l x W_l x C_l}$ denote the normalized features derived from the lth layer of the pre-trained CNN backbone. C_l is the channel size for lth layer. H, W are the height and width, respectively. $\frac{1}{H_l W_l}\sum_{h_l, w_l}$ corresponds to the spatial average function as defined in [25]. h_l, w_l show pixel's coordinate for lth layer. Moreover, $\omega_l \in \mathbb{R}^{C_l x 1}$ refers to the trained weights of LPIPS and its main purpose is reducing C_l value to one, before computing spatial average. The multiplication operator ($*$) is used as linear matrix multiplication. Finally, L is the number of used layers for evaluation. Next metric: Peak Signal-to-Noise Ratio (PSNR) is defined as follows:

$$PSNR(Y', Y) = 10\log\frac{MAXVALUE}{MSE(Y', Y)},$$

where MSE is defined as:

$$MSE(Y', Y) = \frac{1}{HW}\sum_{i=0}^{H}\sum_{j=0}^{W}(Y'(i, j) - Y(i, j))^2 \quad (11)$$

MAXVALUE is set to 1 (normalized maximum), as the ground truth and the pixel values in the generated image are normalized between -1 and 1. PSNR indicates the quality of the reconstruction of the IR images from the visible images [27]. H, W are the height and width of the image, respectively.

4.2. Results

In this section, we compare our InfraGAN's performance to Pix2Pix [12] using residual blocks in its generator, to UNet, and

[2] FLIR dataset, https://www.flir.com/oem/adas/adas-dataset-form/.

Table 1
Comparison of different algorithms on three different datasets. Best result is shown in bold for each metric over all classes for each dataset.

VEDAI dataset [23]	SSIM	MSSIM	LPIPS	L1	PSNR
Pix2Pix [12]	0.72	0.66	0.066	0.175	19.84
ThermalGAN [11]	0.80	0.83	0.019	0.068	27.44
U-Net	**0.89**	0.87	0.013	0.066	27.96
InfraGANv1	0.86	0.87	0.013	0.058	28.74
InfraGANv2	0.79	0.84	0.030	0.065	27.64
InfraGANv3	0.88	**0.90**	**0.011**	**0.055**	29.24
InfraGANv4	0.88	**0.90**	**0.011**	**0.055**	**29.26**
InfraGANv5	0.86	0.88	0.013	0.056	28.98
InfraGANv6	0.88	**0.90**	**0.011**	0.056	29.21
KAIST dataset [24]	SSIM	MSSIM	LPIPS	L1	PSNR
Pix2Pix [12]	0.69	0.55	0.196	0.137	21.25
ThermalGAN [11]	0.66	0.51	0.242	0.165	19.74
U-Net	**0.78**	0.64	0.172	0.137	21.75
InfraGANv2	0.76	0.67	0.167	0.123	22.81
InfraGANv3	0.76	**0.68**	**0.157**	0.123	22.89
InfraGANv5	0.77	**0.68**	0.165	0.122	22.87
InfraGANv6	0.76	0.67	0.159	**0.121**	**22.97**
FLIR dataset	SSIM	MSSIM	LPIPS	L1	PSNR
Pix2Pix [12]	0.13	0.38	0.203	0.228	16.51
ThermalGAN [11]	0.17	0.43	0.192	0.209	17.30
U-Net	**0.31**	**0.54**	0.188	0.169	**19.29**
InfraGANv1	0.24	**0.54**	0.164	0.171	19.23
InfraGANv2	0.25	0.53	0.173	0.171	19.10
InfraGANv3	0.26	**0.54**	0.159	0.169	19.24
InfraGANv4	0.26	**0.54**	**0.158**	0.170	19.21
InfraGANv5	0.26	**0.54**	0.175	**0.168**	19.28

to the ThermalGAN implementation[3] from [11]. In ThermalGAN, there is an extra input which is called as temperature vector T, (see [11] for the definition of vector T), which helps network to learn thermal color spectrum better. However, to have fair comparison in our algorithms (since none of the other algorithms use additional input vector), and since we do not have ground truth T vectors in our used datasets, we used the version that does not consider the input vector T for ThermalGAN in our experiments.

Table 1 compares overall performance of those four different architectures (ThermalGAN, Pix2Pix, UNet and InfraGAN) in five different metrics over three different datasets. Furthermore, we also include different versions of InfraGAN in the table (see Table 4 for the difference between different InfraGAN versions). As shown in Table 1, the VEDAI and KAIST results for each algorithm are higher on average than the results obtained for the FLIR dataset. We think that is mainly because of the format difference of the datasets since the FLIR dataset is the only dataset we used where the visible image was present in grayscale, while the other two datasets provide the visible image in the RGB format. Consequently, we believe that all the networks could not use the color information in FLIR experiments yielding lower results on average when compared to the other two datasets. Overall, in all five metrics, our InfraGAN architecture yielded the best results on both KAIST and VEDAI datasets when compared to the other GAN-based networks. While simple U-Net architecture (that is trained without using a GAN loss) yielded slightly higher results on SSIM, it yielded lower results on the other metrics on average. The table also demonstrates that using also the SSIM loss in our network helped gaining higher performance (compare InfraGANv2 results to InfraGANv5 results).

Table 2 shows individual class performances of each algorithm on the KAIST dataset. The test dataset in KAIST has three classes: Cyclist, People and Person. Each entry in the table shows the five metric in the format of: SSIM / MSSIM / LPIPS / L1 / PSNR for each class.

Table 3 shows individual class performances of each algorithm on the VEDAI dataset. The test dataset in VEDAI has eight classes: car, truck, pickup, tractor, camping car, boat, van and other. Each entry in the table shows the five metric in the format of: SSIM / MSSIM / LPIPS / L1 / PSNR for each class.

Table 4 summarizes our ablation study on computation time and on the use of activation function. The term FPS means frame per second and computed as the time required by algorithm to test a given image (where image resolution is 512×512). Sec/Epoch means the training time (in seconds) per epoch on KAIST dataset. Since we used different versions of our InfraGAN in the table where we changed the hyperparameters including the used activation function and the used loss functions, we named each used version seperately. In total, we listed six different versions of InfraGAN in the table. Additionally, Table 5 shows comparative results of using L1 vs. using Mean Square Error (MSE) in InfraGANv2 on two different datasets.

Figure 5 compares qualitative results obtained from three different algorithms, namely Pix2Pix, ThermalGAN and InfraGAN (ours) for selected (sample) two pairs from each dataset. In the figure, the first two rows show the results of each compared algorithm in this paper for the image pairs sampled from VEDAI, the third and fourth rows show results for the sample image pairs from KAIST and the last two rows show the results for the sample image pairs from FLIR datasets, respectively. Some of our results yield checkerboard artifact and we think one potential reason might be due to the use of trainable deconvolutional layers that are used instead of upscaling layers.

Figure 6 qualitatively compares the results of UNet to GAN-based networks on three sample images from KAIST. As it can be seen in the figure, UNet results are more blurry when compared to the results of GAN-based networks.

[3] https://github.com/vlkniaz/ThermalGAN.

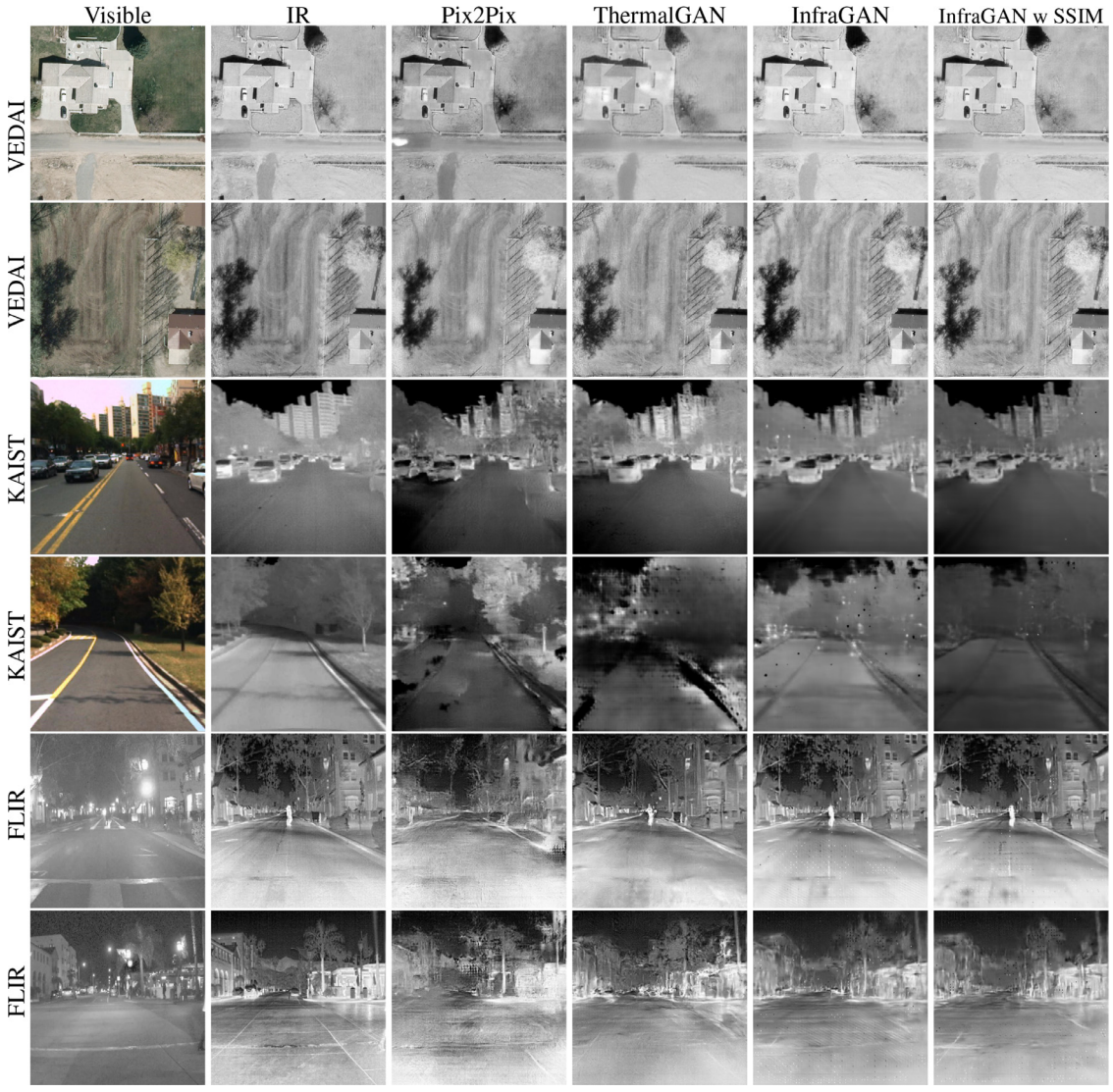


Fig. 5. Comparative results of multiple algorithms on sample image pairs from three datasets (the first two rows are from VEDAI, the third and fourth rows are from KAIST, the last two rows are from FLIR). The first two columns are the real images (visible and IR). The 3rd through 6th column images are the results from Pix2Pix, ThermalGAN, InfraGAN without SSIM loss and InfraGAN with SSIM loss, respectively.

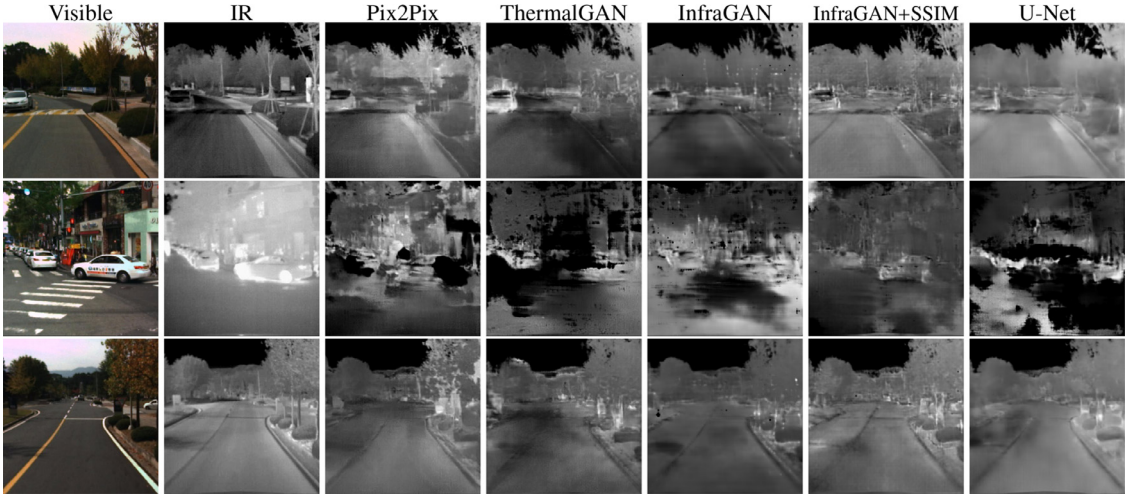


Fig. 6. Comparative results between simple U-Net generator and other GANs to show the difference of using GAN loss. These qualitative results show that using GAN architecture can help improving the generated image quality. In the figure, the first two columns show the original images, from third column to sixth column shows GAN-based networks' results and the last column shows U-Net results on three samples.

Table 2
Comparison of four different algorithms on the KAIST dataset for each class. Each entry shows five different metrics in the order of: SSIM, MSSIM, LPIPS, L1 and PSNR. Best result is shown in bold for each metric.

KAIST dataset [23]	Cyclist	People	Person
Pix2Pix	0.70/0.54/0.196/0.155/20.63	0.69/0.55/0.196/0.144/20.96	0.68/0.54/0.199/0.143/21.00
ThermalGAN	0.66/0.51/0.243/0.176/19.33	0.65/0.52/0.237/0.174/19.43	0.65/0.52/0.235/0.168/19.66
INFRAGAN w/o SSIM	0.78/0.69/0.154/0.125/23.06	0.76/0.68/0.163/0.131/**22.44**	0.75/0.66/0.173/0.131/22.23
INFRAGAN	**0.78**/**0.70**/**0.147**/**0.118**/**23.29**	**0.76**/**0.68**/**0.161**/**0.130**/22.30	**0.75**/**0.66**/**0.170**/**0.128**/**22.40**

Table 3
Comparison of four different algorithms on the VEDAI dataset for each class. Each entry shows five different metrics in the order of: SSIM, MSSIM, LPIPS, L1 and PSNR. Best result is shown in bold for each metric.

VEDAI dataset [23]	Car	Truck	Pickup	Tractor	Camping Car	Boat	Van	Other
Pix2Pix [12]:	0.72/0.67/0.065 /0.161/20.39	0.74/0.70/0.056 /0.173/20.09	0.73/0.69/0.061 /0.169/20.12	0.71/0.67/0.065 /0.162/20.23	0.72/0.66/0.060 /0.166/20.10	0.75/0.79/0.050 /0.109/23.45	0.68/0.60/0.069 /0.167/19.64	0.73/0.65/0.075 /0.266/17.05
ThermalGAN [11]:	0.81/0.85/0.019 /0.064/27.98	0.82/0.86/0.017 /0.065/27.56	0.82/0.84/0.019 /0.067/27.51	0.80/0.83/0.020 /0.70/27.21	0.80/0.84/0.018 /0.065/27.81	0.82/0.86/0.019 /0.063/27.69	0.77/0.78/0.030 /0.104/23.76	0.84/0.85/0.015 /0.070/27.88
InfraGAN w/o SSIM:	0.79/0.85/0.029 /0.061/28.11	0.81/0.87/0.026 /0.061/27.90	0.80/0.85/0.029 /0.066/27.51	0.78/0.84/0.029 /0.064/27.51	0.79/0.85/0.029 /0.064/27.77	0.80/0.86/0.030 /0.067/26.64	0.76/0.81/0.039 /0.094/24.27	0.82/0.86/0.022 /0.055/29.08
InfraGAN:	**0.86**/**0.89**/**0.013** /**0.053**/**29.43**	**0.87**/**0.90**/**0.013** /**0.057**/**28.72**	**0.86**/**0.89**/**0.014** /**0.057**/**28.87**	**0.85**/**0.88**/**0.014** /**0.056**/**28.97**	**0.86**/**0.89**/**0.013** /**0.054**/**29.49**	**0.86**/**0.86**/**0.018** /**0.063**/**27.53**	**0.82**/**0.83**/**0.024** /**0.081**/**25.21**	**0.88**/**0.90**/**0.011** /**0.052**/**29.85**

Table 4
Network definitions based on their configurations, where Łrefers to the used loss function and A.F. means activation function. Encoder and Decoder is listed for the generator. This table also compares run times of different networks on KAIST dataset.

Algorithms	$Ł_{cGAN}$	$Ł_{L1}$	$Ł_{MSE}$	$Ł_{SSIM}$	A.F. in Encoder	A.F. in Decoder	A.F. in Discriminator	Sec/Epoch	FPS
ThermalGAN [11]	✓	✓			LeakyReLU	ReLU	ReLU	444 sec	328.06
UNet		✓		✓	LeakyReLU	ReLU	ReLU	936 sec	341.36
Pix2Pix [12]	✓	✓			ReLU	ReLU	ReLU	1104 sec	264.95
INFRAGANv1	✓		✓		LeakyReLU	ReLU	ReLU	2148 sec	340.37
INFRAGANv2	✓	✓			LeakyReLU	ReLU	ReLU	2172 sec	342.12
INFRAGANv3	✓	✓			LeakyReLU	ReLU	LeakyReLU	2220 sec	340.76
INFRAGANv4	✓	✓			LeakyReLU	LeakyReLU	ReLU	2244 sec	341.56
INFRAGANv5	✓	✓		✓	LeakyReLU	ReLU	ReLU	2256 sec	342.29
INFRAGANv6	✓	✓			LeakyReLU	LeakyReLU	LeakyReLU	2252 sec	341.56

Table 5
Comparison of using L1 (InfraGANv2) and using Mean Squared Error, MSE, (InfraGANv1) for pixel by pixel loss on VEDAI and FLIR.

VEDAI [23]	SSIM	MSSIM	LPIPS	L1	PSNR
L1	0.79	0.84	0.030	0.065	27.64
MSE	**0.86**	**0.87**	**0.013**	**0.058**	**28.74**
FLIR dataset	SSIM	MSSIM	LPIPS	L1	PSNR
L1	**0.25**	0.53	0.173	**0.171**	19.10
MSE	0.24	**0.54**	**0.164**	**0.171**	**19.23**

5. Conclusion

As the use of IR sensors increases, the demand on using algorithms that can run on IR images also increases. The IR images complement RGB images from the thermal spectrum and help algorithms to detect many objects easier. However, the lack of big and public datasets containing IR images, limits what can be obtained from the existing deep object detection algorithms (such as SyNet [28] or YOLO [29]). In this work, we tackle that problem where algorithms can learn the relation between the visible and IR spectrum so that we can artificially generate the IR equivalent of any given visible image. For that purpose, as being the first step, we introduce InfraGAN architecture and study the performance of different InfraGAN versions in this paper. While the literature has some preliminary works reporting qualitative results (see for example [17]), our work is one of the early works that presents a comprehensive analysis using three different benchmarking data sets and five different metrics. Our experimental results (see Table 1) show that our proposed algorithm yields the best overall performance on all five metrics for both VEDAI and KAIST datasets. Additionally, Fig. 5 demonstrates that our generated IR images look similar to the ground truth IR images. That makes our approach of using GAN-based networks utilizing U-Nets with two level discriminator a good alternative solution to obtain the IR equivalent of a given visible image. Since we build additional layers as decoder on top of the existing encoder stage, our implementation takes slightly longer time during the training (see Table 4) when compared to ThermalGAN and Pix2Pix architectures. Figure 6 compares U-Net architecture (as used in our implementation, the U-Net architecture is shown in Fig. 2) to GAN-based networks. As shown in the figure, the U-Net architecture (which is trained by using both SSIM and L1 losses) yields more blurry outputs than GAN based networks, even if it yields slightly higher SSIM results in Table 1.

Declaration of Competing Interest

Sedat Ozer is affiliated with Ozyegin University and Mehmet Akif Ozkanoglu is affiliated with both Ozyegin University and Bilkent University. Both authors are funded by TUBITAK. There is no any other funding agency currently funding our work.

Acknowledgment

This paper has been produced benefiting from the 2232 International Fellowship for Outstanding Researchers Program of TÜBİTAK (Project No:118C356). However, the entire responsibility of the paper belongs to the owner of the paper. The financial support received from TÜBİTAK does not mean that the content of the publication is approved in a scientific sense by TÜBİTAK.

References

[1] R. Valiente, M. Zaman, S. Ozer, Y.P. Fallah, Controlling steering angle for cooperative self-driving vehicles utilizing CNN and LSTM-based deep networks, in: 2019 IEEE Intelligent Vehicles Symposium (IV), IEEE, 2019, pp. 2423–2428.

[2] D. Gözen, S. Ozer, Visual object tracking in drone images with deep reinforcement learning, in: 25th International Conference on Pattern Recognition (ICPR2020), 2020.

[3] N. Barzilay, T.B. Shalev, R. Giryes, MISS GAN: a multi-illustrator style generative adversarial network for image to illustration translation, Pattern Recognit. Lett. 151 (2021) 140–147.

[4] L. Ran, Y. Hong, S. Zhang, Y. Yang, Y. Zhang, Improving visible-thermal ReID with structural common space embedding and part models, Pattern Recognit. Lett. 142 (2021) 25–31.

[5] J. Miura, M. Demura, K. Nishi, S. Oishi, Thermal comfort measurement using thermal-depth images for robotic monitoring, Pattern Recognit. Lett. 137 (2020) 108–113.

[6] C. Bisogni, L. Cascone, J.-L. Dugelay, C. Pero, Adversarial attacks through architectures and spectra in face recognition, Pattern Recognit. Lett. 147 (2021) 55–62.

[7] P. Wang, F. Su, Z. Zhao, Y. Zhao, L. Yang, Y. Li, Deep hard modality alignment for visible thermal person re-identification, Pattern Recognit. Lett. 133 (2020) 195–201.

[8] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A.C. Berg, L. Fei-Fei, ImageNet large scale visual recognition challenge, Int. J. Comput. Vis. (IJCV) 115 (3) (2015) 211–252, doi:10.1007/s11263-015-0816-y.

[9] M. Everingham, S.M.A. Eslami, L. Van Gool, C.K.I. Williams, J. Winn, A. Zisserman, The pascal visual object classes challenge: a retrospective, Int. J. Comput. Vis. 111 (1) (2015) 98–136.

[10] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, C.L. Zitnick, Microsoft COCO: common objects in context, in: European Conference on Computer Vision, Springer, 2014, pp. 740–755.

[11] V.V. Kniaz, V.A. Knyaz, J. Hladuvka, W.G. Kropatsch, V. Mizginov, ThermalGAN: multimodal color-to-thermal image translation for person re-identification in multispectral dataset, in: Proceedings of the European Conference on Computer Vision (ECCV) Workshops, 2018.

[12] P. Isola, J.-Y. Zhu, T. Zhou, A.A. Efros, Image-to-image translation with conditional adversarial networks, CVPR (2017).

[13] Y. Jiang, Y. Liu, Q. Peng, F. Jie, D. Ming, Infrared image generation method based on visible light remote sensing image, in: 2019 IEEE 3rd Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC), 2019, pp. 63–68, doi:10.1109/IMCEC46724.2019.8984157.

[14] L. Zhang, A. Gonzalez-Garcia, J. van de Weijer, M. Danelljan, F.S. Khan, Synthetic data generation for end-to-end thermal infrared tracking, IEEE Trans. Image Process. 28 (4) (2018) 1837–1850.

[15] L. Li, P. Li, M. Yang, S. Gao, Multi-branch semantic GAN for infrared image generation from optical image, in: Z. Cui, J. Pan, S. Zhang, L. Xiao, J. Yang (Eds.), Intelligence Science and Big Data Engineering. Visual Data Engineering, Springer International Publishing, Cham, 2019, pp. 484–494.

[16] H. Xu, P. Liang, W. Yu, J. Jiang, J. Ma, Learning a generative model for fusing infrared and visible images via conditional generative adversarial network with dual discriminators, in: IJCAI, 2019, pp. 3954–3960.

[17] J.-Y. Zhu, T. Park, P. Isola, A.A. Efros, Unpaired image-to-image translation using cycle-consistent adversarial networks, in: Computer Vision (ICCV), 2017 IEEE International Conference on, 2017.

[18] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.

[19] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, Adv. Neural Inf. Process. Syst. 27 (2014).

[20] E. Schonfeld, B. Schiele, A. Khoreva, A U-Net based discriminator for generative adversarial networks, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 8207–8216.

[21] H.R. Sheikh, M.F. Sabir, A.C. Bovik, A statistical evaluation of recent full reference image quality assessment algorithms, IEEE Trans. Image Process. 15 (11) (2006) 3440–3451.

[22] Zhou Wang, A.C. Bovik, H.R. Sheikh, E.P. Simoncelli, Image quality assessment: from error visibility to structural similarity, IEEE Trans. Image Process. 13 (4) (2004) 600–612, doi:10.1109/TIP.2003.819861.

[23] S. Razakarivony, F. Jurie, Vehicle detection in aerial imagery: a small target detection benchmark, J. Vis. Commun. Image Represent. 34 (2015), doi:10.1016/j.jvcir.2015.11.002.

[24] S. Hwang, J. Park, N. Kim, Y. Choi, I.S. Kweon, Multispectral pedestrian detection: benchmark dataset and baselines, in: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015.

[25] R. Zhang, P. Isola, A.A. Efros, E. Shechtman, O. Wang, The unreasonable effectiveness of deep features as a perceptual metric, CVPR, 2018.

[26] A. Krizhevsky, I. Sutskever, G.E. Hinton, ImageNet classification with deep convolutional neural networks, in: F. Pereira, C.J.C. Burges, L. Bottou, K.Q. Weinberger (Eds.), Advances in Neural Information Processing Systems, vol. 25, Curran Associates, Inc., 2012, pp. 1097–1105. https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[27] A. Horé, D. Ziou, Image quality metrics: PSNR vs. SSIM, in: 2010 20th International Conference on Pattern Recognition, 2010, pp. 2366–2369, doi:10.1109/ICPR.2010.579.

[28] B.M. Albaba, S. Ozer, SyNet: an ensemble network for object detection in UAV images, in: 25th International Conference on Pattern Recognition (ICPR2020), 2020.

[29] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, You only look once: unified, real-time object detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 779–788.

Sedat Ozer received his M.Sc. degree from Univ. of Massachusetts, Dartmouth and his Ph.D. degree from Rutgers University, NJ. He has worked as a research associate at Univ. of Virginia and Massachusetts Institute of Technology. His research interests include object detection & segmentation, object tracking, visual data analysis, geometric and explainable AI algorithms and explainable fusion algorithms. He is currently an Assistant Prof. at Ozyegin University and a recepient of TUBITAK's international outstanding research fellow.

Mehmet Akif Özkanoğlu is currently a M.Sc. student at Bilkent University and is working on designing efficient algorithms for autonomous systems. He received his B.Sc. degree from Istanbul Technical University and his research interests include deep learning, object detection, tracking and robotics.