딥러닝 기반 저조도 CCTV 개선

2020. 10. 24

지도교수 : 양정모 교수님

1 조 안지현

목 차

연구 발표자

이 름	연구 주제
안지현	딥러닝 기반 저조도 CCTV 화질 개선 연구

Artificial Intelligence과 deep learning ⇨ AI란 사고나 학습 등 인간이 가진 지적 능력을 컴퓨터를 통해 구현하는 기술. 이것을 구현하는 방법의 하나인 딥러닝은 인간의 뉴런과 비슷한 인공신경망 방식으로 컴퓨터가 스스로 데이터의 특징을 학습하고 인식하는 기술

주제 선정

CCTV로 잡았다! 절도범부터 마약사범까지…그 비결은?

시민기자 김진형

Visit 970 Date

[기고] CCTV 카메라의 야간촬영 발전 방향

이길주 기자 | 승인 2018.12.05 17:56 | 댓글 0

(주)현명 김철현 대표

글로벌 시장조사기관 (Frost&Sullivan)에 따르면 201 24.1억 달러에서 연평균 4.1% 30.6억 달러로 전 세계적으로 다.

또한 지능형 영상 감시 시스템 상이 접목되면서 주·야간 고품 요는 증가할 것으로 보인다.

초기에는 난관도 많아…인프라 부족, 비용부담, 인원구축 문제

스마트도시통합운영센터의 필요성은 개소 이전부터 제기됐다. 디지털 기술이 발달하면서 CCTV의 쓰임이 많아졌다. 교통관제, 무단 투기 단속 등 각각의 필요성에 의해 구청 곳곳에 모니터를 설치했다. 목적별로 사용하는 CCTV가 많아지면서 중복 투자, 예산 낭비 논란이 일었다. 그런 와중에 나온 아이디어가 통합해서 관제하는 시스템을 구축해 효율적으로 운영하자는 것이었다.

CNN 모델을 사용한 수준 높은 저조도 영상 개선 시스템 개발 추진

시스템 구상도(1/2)

기능적 측면

3-tier Low-Light video Enhancement

1-tier

- 사용자 입력
- 전처리 단계
- Video to Frame

2-tier

- Retinex 이론 기반의 딥러닝 저조도 이미지 개선 model (KinD-Network)

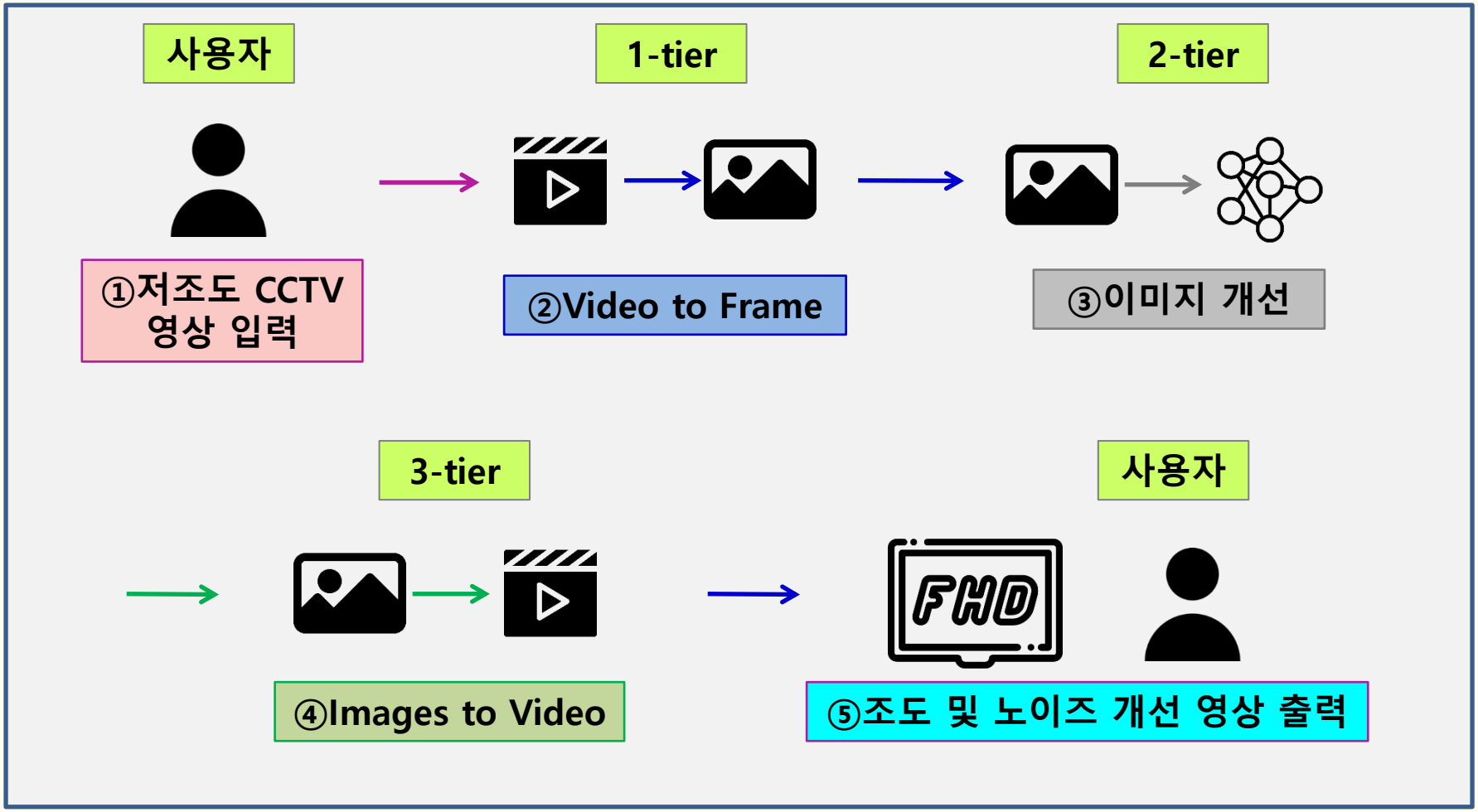
3-tier

- 후처리 단계
- 개선 이미지 to Video
- 최종 결과물 확인

사용자 입력 저조도 영상을 CNN 모델로 조도 및 노이즈를 개선 출력

시스템 구상도(2/2)

동작 원리

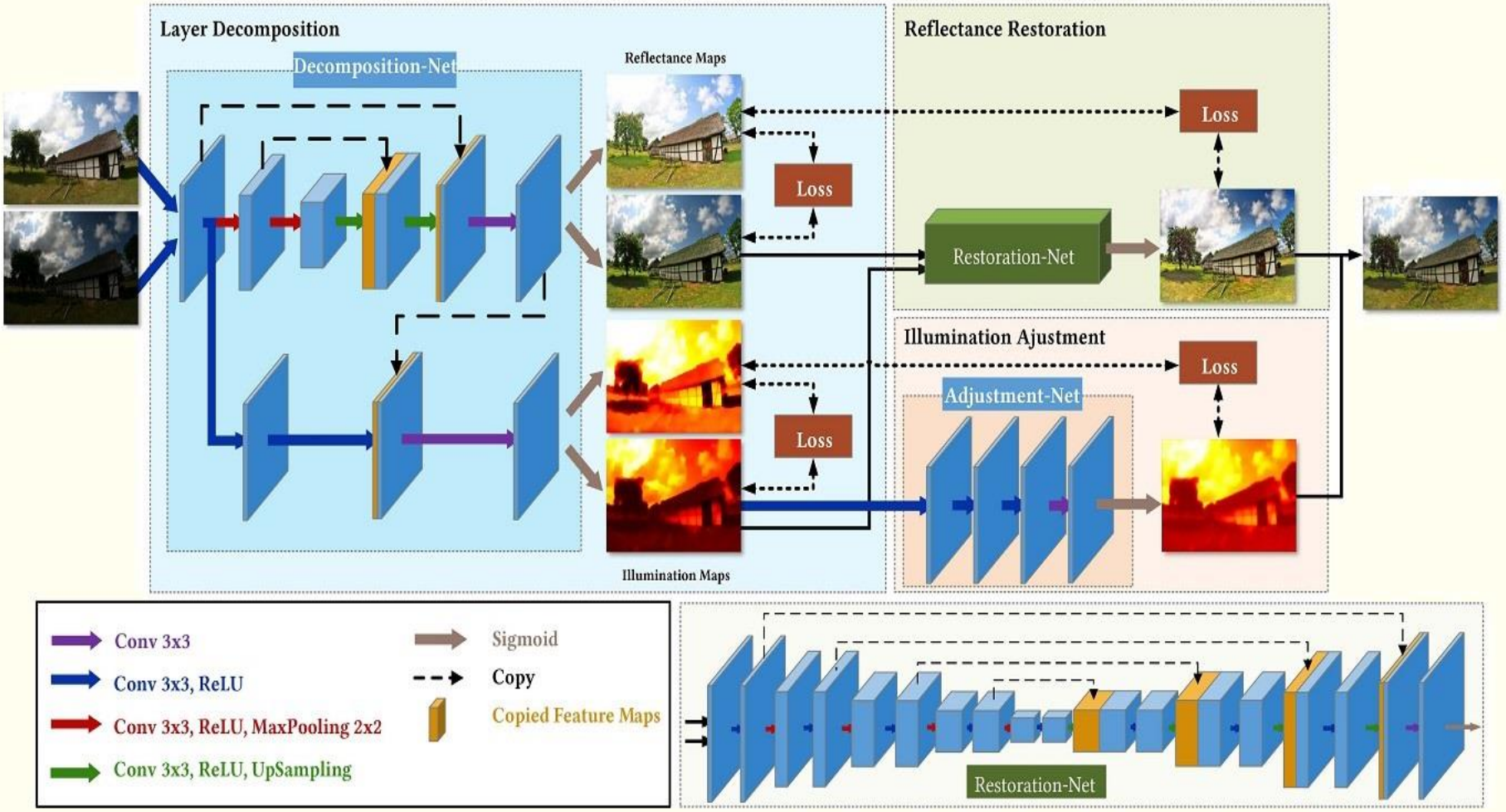


추진 경과

추진기간(2020년) / 수행 업무	6월	7월	8월	9월	10월
자료 조사 및 연구					
저조도 개선 알고리즘 연구					
전처리 시스템 구현					
후처리 시스템 구현					
시스템 검증 및 보완					

KinD Network(1/2)

2-tier 동작 원리



KinD Network(2/2)

2-tier test

low-light test image

Decomposition net 출력

Reflrctance Restoration net 출력

Illumination Adjust net 출력

KinD result

연구 및 평가(1/9)

연구 환경

Tools

- 영상처리 : OpenCV, PIL, numpy, skimage, scipy
- 딥러닝 : Tensorflow

Algorithm

- Model Structure : CNN
- Loss Function : MSE

OS

Windows 10
Anaconda 3
Conda 4.8.3

Development Language

Python 3.6.12

연구 및 평가(2/9)

연구 시스템 구현

○ 비디오 프레임 분할 저장

동영상 읽기

프레임 분할

```
eval_low_data = []
eval_img_name =[]
eva
eva

for
def video_to_image(file):
    cap = cv2.VideoCapture(file)
    count = 0
    eval_low_im = []
    while(cap.isOpened()):
        ret, frame = cap.read()
        if frame is None:
            break
        else:
            count = str(count)
            count2 = count.zfill(4)
            cv2.imwrite('./frame_pre/{0}.png'.format(count2), frame)
            count = int(count)
            print('frame decomposition to png file in [frame_pre] dir ... %d'%count)
            count+=1
```

① 사용자 입력 동영상 읽기 ② 프레임 분할하여 저장

연구 및 평가(3/9)

연구 시스템 구현

❍ Image Load

프레임 읽기

```
eval_low_data2 = []
eval_img_name2 =[]
eval_low_data_name2 = glob('./frame_pre/*.png')
eval_low_data_name2.sort()
```

이미지 전처리

```
def load_images(file):
    im = Image.open(file)
    img = np.array(im, dtype="float32") / 255.0
    img_max = np.max(img)
    img_min = np.min(img)
    img_norm = np.float32((img - img_min) / np.maximum((img_max - img_min), 0.001))
    return img_norm

def tensor_size(tensor):
    from operator import mul
    return reduce(mul, (d.value for d in tensor.get_shape()[1:]), 1)
```

① 프레임 읽기 ② python list 자료구조로 입력받은 이미지 data를 numpy를 이용하여 배열처리 후 이미지 전처리(정규화)

연구 및 평가 (4/9)

연구 시스템 구현

❍ KinD-Net 구현(1/4) – 네트워크 실행 및 CNN 함수

네트워크 실행 코드

```
print("Start evalating!")
print(time.strftime('%Y-%m-%d %p %I:%M:%S', time.localtime(time.time())))
for idx in range(len(eval_low_data2)):
    print(idx)
    name = eval_img_name2[idx]
    input_low = eval_low_data2[idx]
    input_low_eval = np.expand_dims(input_low, axis=0)
    h,
    de
    re
    ra
    i
    i
    i
    ac

    decom_r_sq = np.squeeze(decom_r_low)
    r_gray = color.rgb2gray(decom_r_sq)
```

CNN 함수 구현

```
def lrelu(x, trainbable=None):
    return tf.maximum(x*0.2,x)

def upsample_and_concat(x1, x2, output_channels, in_channels, scope_name, trainable=True):
    with tf.compat.v1.variable_scope(scope_name, reuse=tf.compat.v1.AUTO_REUSE) as scope:
        pool_size = 2
        deconv_filter = tf.compat.v1.get_variable('weights', [pool_size, pool_size, output_channels, in_channels], trainable=
        deconv = tf.nn.conv2d_transpose(x1, deconv_filter, tf.shape(x2) , strides=[1, pool_size, pool_size, 1], name=scope_na

        deconv_output =  tf.concat([deconv, x2],3)
        deconv_output.set_shape([None, None, None, output_channels*2])

        return deconv_output
```

① 전처리한 이미지들을 KinD-Net 모델 각각의 sub-Net(3개) 통과
② 사용자 임의의 밝기 조절 변수인 ratio = 4.0(default)

연구 및 평가(5/9)

연구 시스템 구현

❍ KinD-Net 구현(2/4) – 이미지 분해

```
def DecomNet_simple(input):
    with tf.compat.v1.variable_scope('DecomNet', reuse=tf.
        conv1=slim.conv2d(input,32,[3,3], rate=1, activation_fn=lrelu,scope='g_conv1_1')
        pool1=slim.max_pool2d(conv1, [2, 2], stride = 2, padding='SAME' )
        conv2=slim.conv2d(pool1,64,[3,3], rate=1, activation_fn=lrelu,scope='g_conv2_1')
        pool2=slim.max_pool2d(conv2, [2, 2], stride = 2, padding='SAME' )
        conv3=slim.conv2d(pool2,128,[3,3], rate=1, activation_fn=lrelu,scope='g_conv3_1')
        up8 =  upsample_and_concat( conv3, conv2, 64, 128 , 'g_up_1')
        conv8=slim.conv2d(up8,  64,[3,3], rate=1, activation_fn=lrelu,scope='g_conv8_1')
        up9 =  upsample_and_concat( conv8, conv1, 32, 64 , 'g_up_2')
        conv9=slim.conv2d(up9,  32,[3,3], rate=1, activation_fn=lrelu,scope='g_conv9_1')
        # Here, we use 1*1 kernel to replace the 3*3 ones in the paper to get better results.
        conv10=slim.conv2d(conv9,3,[1,1], rate=1, activation_fn=None, scope='g_conv10')
        R_out = tf.sigmoid(conv10)

        l_conv2=slim.conv2d(conv1,32,[3,3], rate=1, activation_fn=lrelu,scope='l_conv1_2')
        l_conv3=tf.concat([l_conv2, conv9],3)
        # Here, we use 1*1 kernel to replace the 3*3 ones in the paper to get better results.
        l_conv4=slim.conv2d(l_conv3,1,[1,1], rate=1, activation_fn=None,scope='l_conv1_4')
        L_out = tf.sigmoid(l_conv4)

    return R_out, L_out
```

이미지 밝기/색상 분리

입력 이미지의 밝기와 색상을 분리하여 출력

연구 및 평가(6/9)

연구 시스템 구현

❍ KinD-Net 구현(3/4) – 반사율 복원 네트워크

색상 복원

색상 복원2

```
def Restoration_net(input_r, input_i):
    with tf.compat.v1.variable_scope('Restoration_net', reuse=tf.compat.v1.AU
        input_all = tf.concat([input_r,input_i], 3)

        conv1=slim.conv2d(input_all,32,[3,3], rate=1, activation_fn=lrelu,scope='de_conv1_1')
        conv1=slim.
        pool1=slim.
                conv6=slim.conv2d(up6,  256,[3,3], rate=1, activation_fn=lrelu,scope
                conv6=slim.conv2d(conv6,256,[3,3], rate=1, activation_fn=lrelu,scope
        conv2=slim.
        conv2=slim.
        pool2=slim.
                up7 =  upsample_and_concat( conv6, conv3, 128, 256, 'up_7'  )
                conv7=slim.conv2d(up7,  128,[3,3], rate=1, activation_fn=lrelu,scope='de_conv7_1')
                conv7=slim.conv2d(conv7,128,[3,3], rate=1, activation_fn=lrelu,scope='de_conv7_2')
        conv3=slim.
        conv3=slim.
        pool3=slim.
                up8 =  upsample_and_concat( conv7, conv2, 64, 128, 'up_8' )
                conv8=slim.conv2d(up8,  64,[3,3], rate=1, activation_fn=lrelu,scope='de_conv8_1')
                conv8=slim.conv2d(conv8,64,[3,3], rate=1, activation_fn=lrelu,scope='de_conv8_2')
        conv4=slim.
        conv4=slim.
        pool4=slim.
                up9 =  upsample_and_concat( conv8, conv1, 32, 64, 'up_9' )
                conv9=slim.conv2d(up9,  32,[3,3], rate=1, activation_fn=lrelu,scope='de_conv9_1')
                conv9=slim.conv2d(conv9,32,[3,3], rate=1, activation_fn=lrelu,scope='de_conv9_2')
        conv5=slim.
        conv5=slim.
                conv10=slim.conv2d(conv9,3,[3,3], rate=1, activation_fn=None, scope='de_conv10')

        up6 = upsa
                out = tf.sigmoid(conv10)
```

주간(낮) 인간의 지각력과 유사한 색상으로 복원하여 출력

연구 및 평가(7/9)

연구 시스템 구현

❍ KinD-Net 구현(4/4) – 빛 조정 네트워크

빛 조정

```
def Illumination_adjust_net(input_i, input_ratio):
    with tf.compat.v1.variable_scope('Illumination_adjust_net', reuse=tf.compat.v1.AUTO_REUSE):
        input_all = tf.concat([input_i, input_ratio], 3)

        conv1=slim.conv2d(input_all,32,[3,3], rate=1, activation_fn=lrelu,scope='en_conv_1')
        conv2=slim.conv2d(conv1,32,[3,3], rate=1, activation_fn=lrelu,scope='en_conv_2')
        conv3=slim.conv2d(conv2,32,[3,3], rate=1, activation_fn=lrelu,scope='en_conv_3')
        conv4=slim.conv2d(conv3,1,[3,3], rate=1, activation_fn=lrelu,scope='en_conv_4')

        L_enhance = tf.sigmoid(conv4)
    return L_enhance

    ratio = 4.0
    i_low_data_ratio = np.ones([h, w]) * (ratio)
    i_low_ratio_expand = np.expand_dims(i_low_data_ratio, axis=2)
    i_low_ratio_expand2 = np.expand_dims(i_low_ratio_expand, axis=0)
    adjust_i = sess.run(output_i, feed_dict={input_low_i: decom_i_low, input_low_i_ratio: i_low_ratio_expand2})

    decom_r_sq = np.squeeze(decom_r_low)
    r_gray = color.rgb2gray(decom_r_sq)
    r_gray_gaussion = filters.gaussian(r_gray, 3)
    low_i = np.minimum((r_gray_gaussion * 2) ** 0.5, 1)
    low_i_expand_0 = np.expand_dims(low_i, axis=0)
    low_i_expand_3 = np.expand_dims(low_i_expand_0, axis=3)
    result_denoise = restoration_r * low_i_expand_3
    fusion4 = result_denoise * adjust_i
```

Layer Decomposition Net에서 얻은 출력의 밝기 조정

연구 및 평가(8/9)

연구 시스템 구현

❍ 프레임 합친 후 비디오 저장

모델 출력 읽기

```
inputpath = 'C:/Users/96dks/Desktop/
outpath =  './test2/5.mp4'
fps = 30

eval_low_data = []
eval_img_name =[]
ifiles = glob(inputpath)
ifiles.sort()
ifilesl = len(ifiles)

for idx in range(ifilesl):
    [_, name] = os.path.split
    suffix = name[name.find(
    name = name[:name.find('
    eval_img_name.append(name

    image_to_video(ifiles, ou
```

mp4 포맷 저장

```
def image_to_video(inputpath, output
    img = []
    image_array = []
    size = (1280, 720)

    for i in range(filesl):
        img = cv2.imread(inputpath[i])
        image_array.append(img)
    fourcc = cv2.VideoWriter_fourcc('D', 'I', 'V', 'X')
    out = cv2.VideoWriter(outputpath, fourcc, fps, size)
    for i in range(len(image_array)):
        out.write(image_array[i])
    out.release()
```

① KinD-Net 처리된 Frame 읽기
② 기존 FPS와 frame size 유지해 mp4 포맷으로 저장

연구 및 평가(9/9)

효과 측정 및 평가

○ 조도 및 노이즈 개선 평가

3-tier Network 전

3-tier Network 후

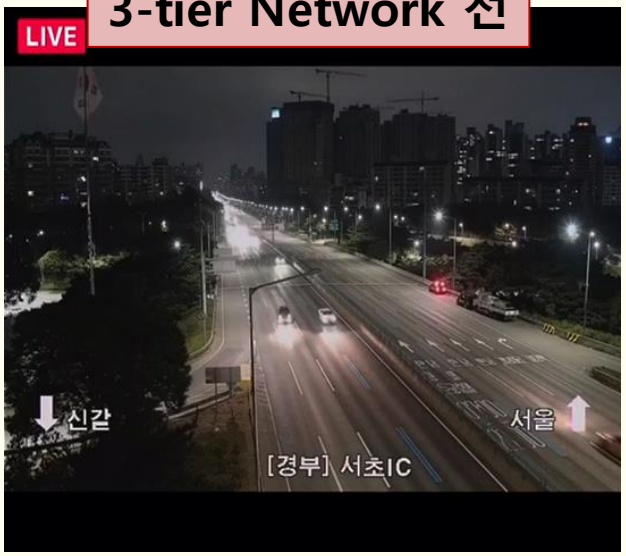
1280x720(HD)해상도 Low-Light Video의『3-tier LoL Video Enhancement Network』출력 결과 조도 및 노이즈 개선

연구 및 평가(9/9)

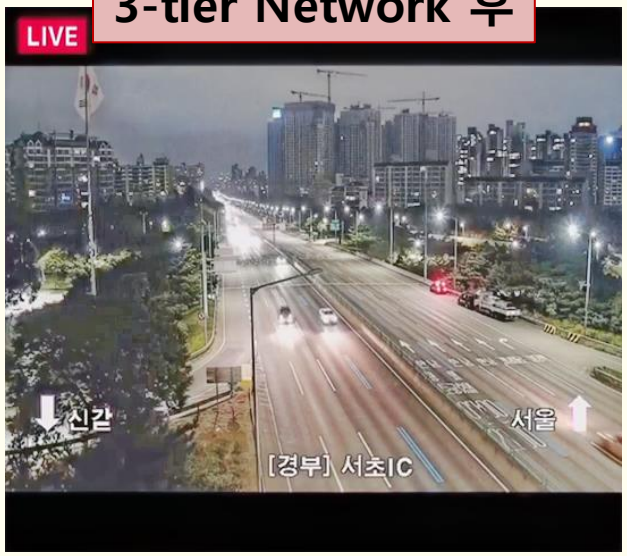
효과 측정 및 평가

❍ 조도 및 노이즈 개선 평가

3-tier Network 전



3-tier Network 후



544x480 해상도 Low-Light Video의『3-tier LoL Video Enhancement Network』출력 결과 조도 및 노이즈 개선

연구 및 평가(9/9)

효과 측정 및 평가

○ 조도 및 노이즈 개선 평가

3-tier Network 전

3-tier Network 후

556x480 해상도 Low-Light Video의『3-tier LoL Video Enhancement Network』출력 결과 조도 및 노이즈 개선

결론 및 기대 효과(1/3)

❍ Low-Light Image Enhancement 해결 과제

- 단순히 밝기를 향상시켰을 경우 증폭되는 노이즈 문제
- 단순히 밝기를 향상시켰을 경우 발생하는 색상 왜곡 문제
- 사용자마다 원하는 빛의 세기가 다른 문제

→ **KinD Network로 개선**

영상 평가 기준으로 정량적 평가 시 기존 알고리즘들보다 성능이 우수

Metrics	BIMEF [33]	CRM [34]	Dong [11]	LIME [16]	MF [14]	RRM [21]
PSNR	13.8753	17.2033	16.7165	16.7586	18.7916	13.8765
SSIM	0.5771	0.6442	0.5824	0.5644	0.6422	0.6577
LOE	1456.1	1757.7	**1283.2**	1909.5	2051.7	2025.5
LOE_{ref}	985.9	**926.1**	1391.5	1342.4	1042.1	958.7
NIQE	7.5150	7.6865	8.3157	8.3777	8.8770	5.8101
Metrics	SRIE [12]	Retinex-Net [30]	MSR [18]	NPE [28]	GLAD [29]	KinD
PSNR	11.8552	16.7740	13.1728	16.9697	19.7182	**20.8665**
SSIM	0.4979	0.5594	0.4787	0.5894	0.7035	**0.8022**
LOE	1745.4	2449.3	2589.4	2076.3	1795.5	2012.2
LOE_{ref}	1199.8	2201.7	2084.8	1643.1	1017.1	977.3
NIQE	7.2869	8.8785	8.1136	8.4390	6.4755	**5.1461**

결론 및 기대 효과(2/3)

❍ 딥러닝 기반 동영상 저조도 개선 시스템 연구 성과

- 자체 기술로 이미지에서 동영상으로 변환 및 역 변환하는 모듈을 구현하고, 이를 기반으로 딥러닝 모델을 사용한 영상 처리 모듈을 구현하는데 성공

- OpenCV, PIL, numpy, Tensorflow 등의 라이브러리를 사용하여 딥러닝 및 영상처리 모듈을 자체 개발

❍ 기대 효과

- 범죄 채증자료로 활용되는 야간 CCTV의 조도를 개선함으로써 자료 활용의 효율성과 정확성을 보장

- Super Resolution 등의 AI 모델과 결합하여 활용할 수 있는 여건 조성

결론 및 기대 효과(3/3)

❍ 향후 개선 과제

- 정보보호 관점의 기술요소 추가

- 딥러닝의 '블랙 박스' 특징 상 3-tier Net에서 데이터가 변형될 가능성은 적지만, 그 이전의 원본 저조도 영상에 대한 변형 가능성과 암호학 관점에서 네트워크 전체 데이터의 무결성에 대한 인증방식에 대한 내용. 끝.

Q & A

감사합니다