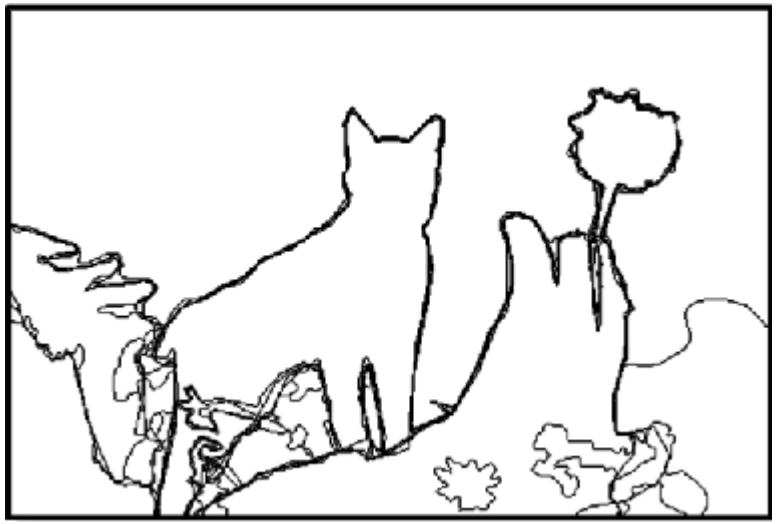
진행 상황

9

목차

- Edge adaptive feature extraction module
  - 배경
  - Method
    - 4색 정리
    - Encoder-decoder 구조
  - Related works
    - AA-net
    - Deformable convolution network
- Soft Argmax의 대안 제시
  - Soft argmax의 이상적인 동작 조건
  - Soft argmax의 대안



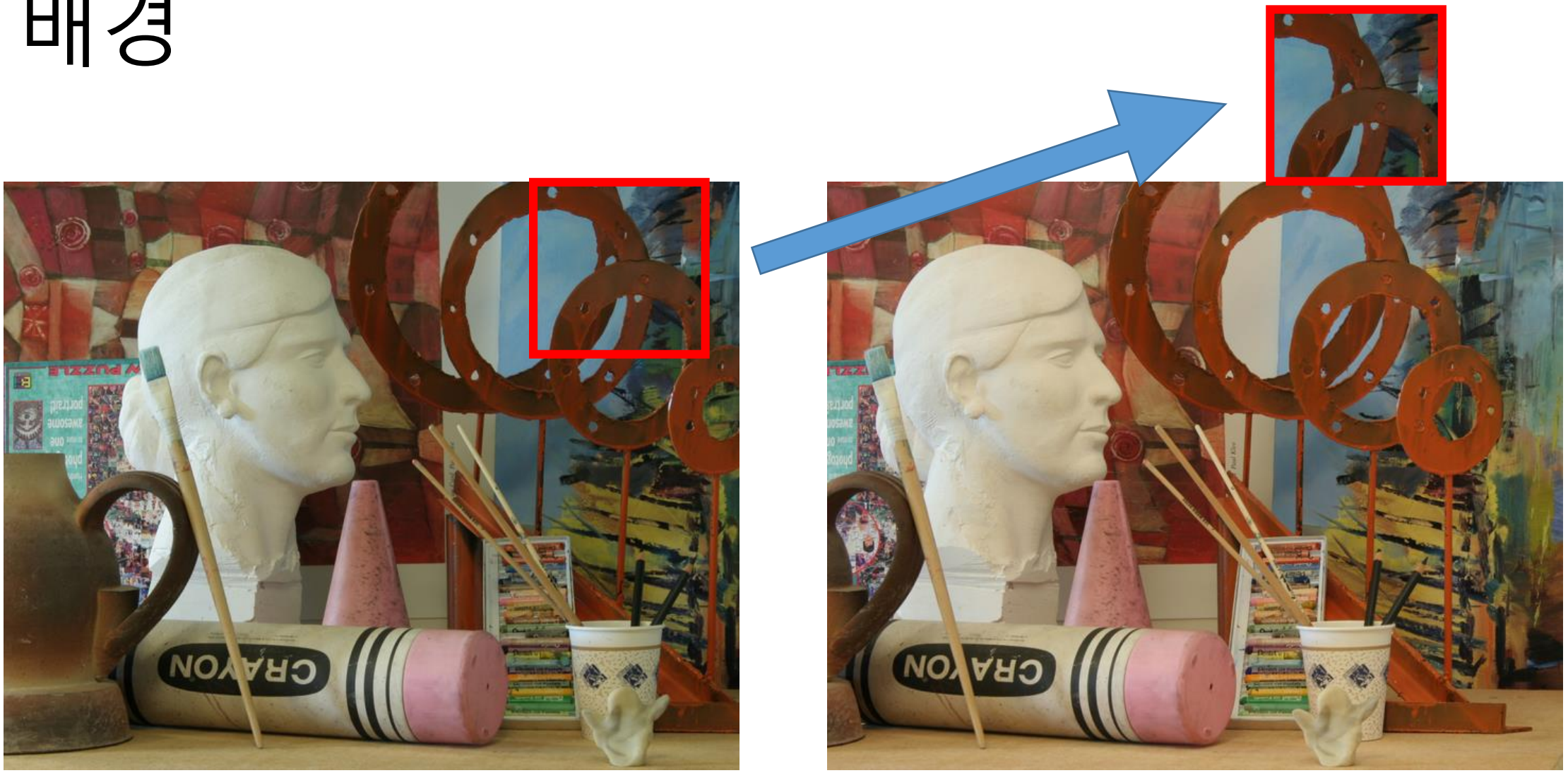
Edge adaptive feature extraction module

배경

- Stereo matching 문제에서 여러 스케일의 윈도우(혹은 receptive field)를 같이 사용하면 이점을 얻을 수 있다. (PSM-net, LW-cnn)

- 하지만 넓은 크기의 윈도우를 사용할 때의 문제점이 있다.

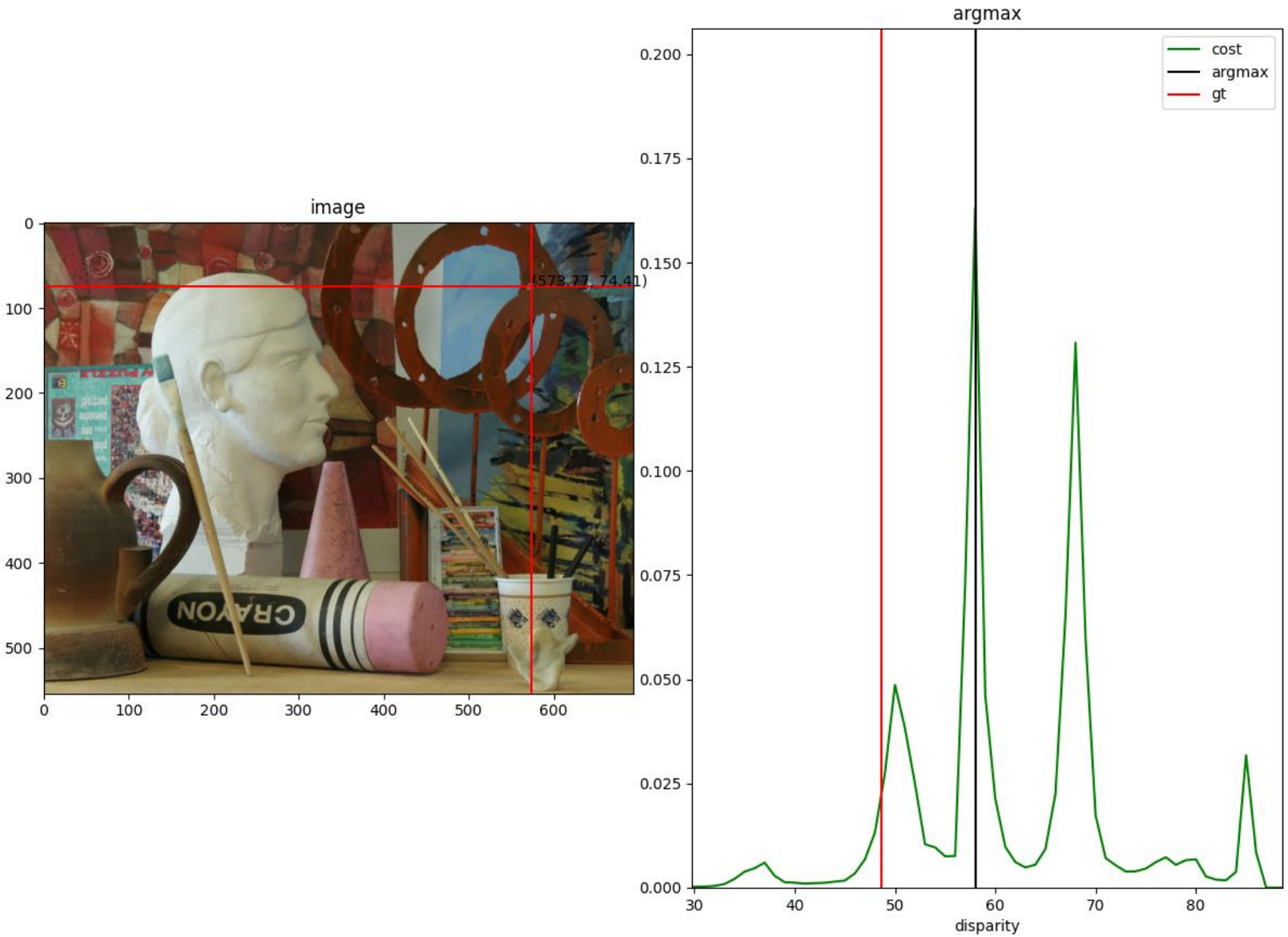
배경



배경

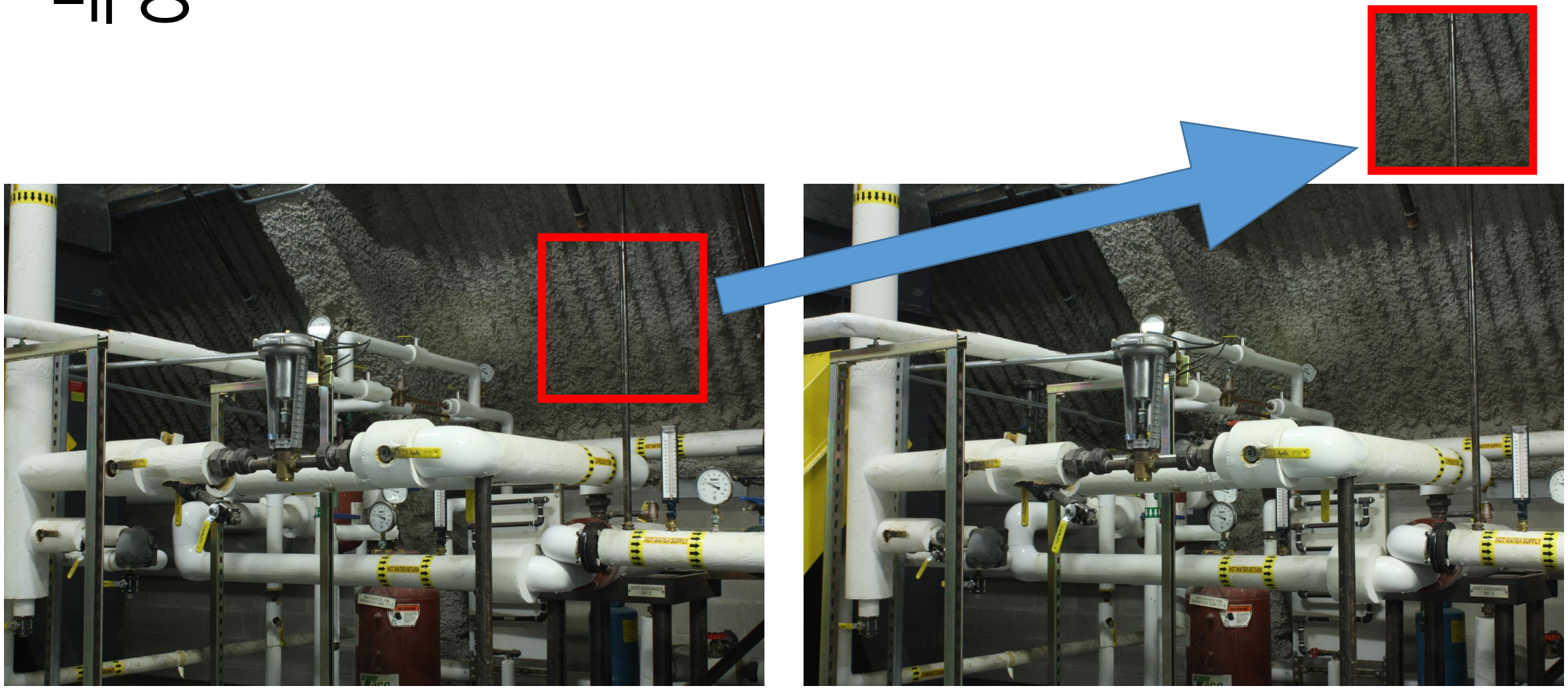
- Large scale의 윈도우(혹은 receptive field)를 사용하여 여러 물체들이 한 윈도우에 같이 들어온 경우 각 물체마다 disparity가 달라 반대편 이미지에서 동일한 정도로 이동해있지 않기 때문에 유사한 픽셀을 찾기 어려워진다.

배경



주변에 물체가 여러 개 있는 경우 직접 설계하고 훈련시킨 신경망에서의 유사도.
두 번째 그림은 첫 번째 그림의 빨간 선이 교차하는 픽셀에서의 유사도 그래프.
(x축은 disparity, y축은 확률을 나타냄.)

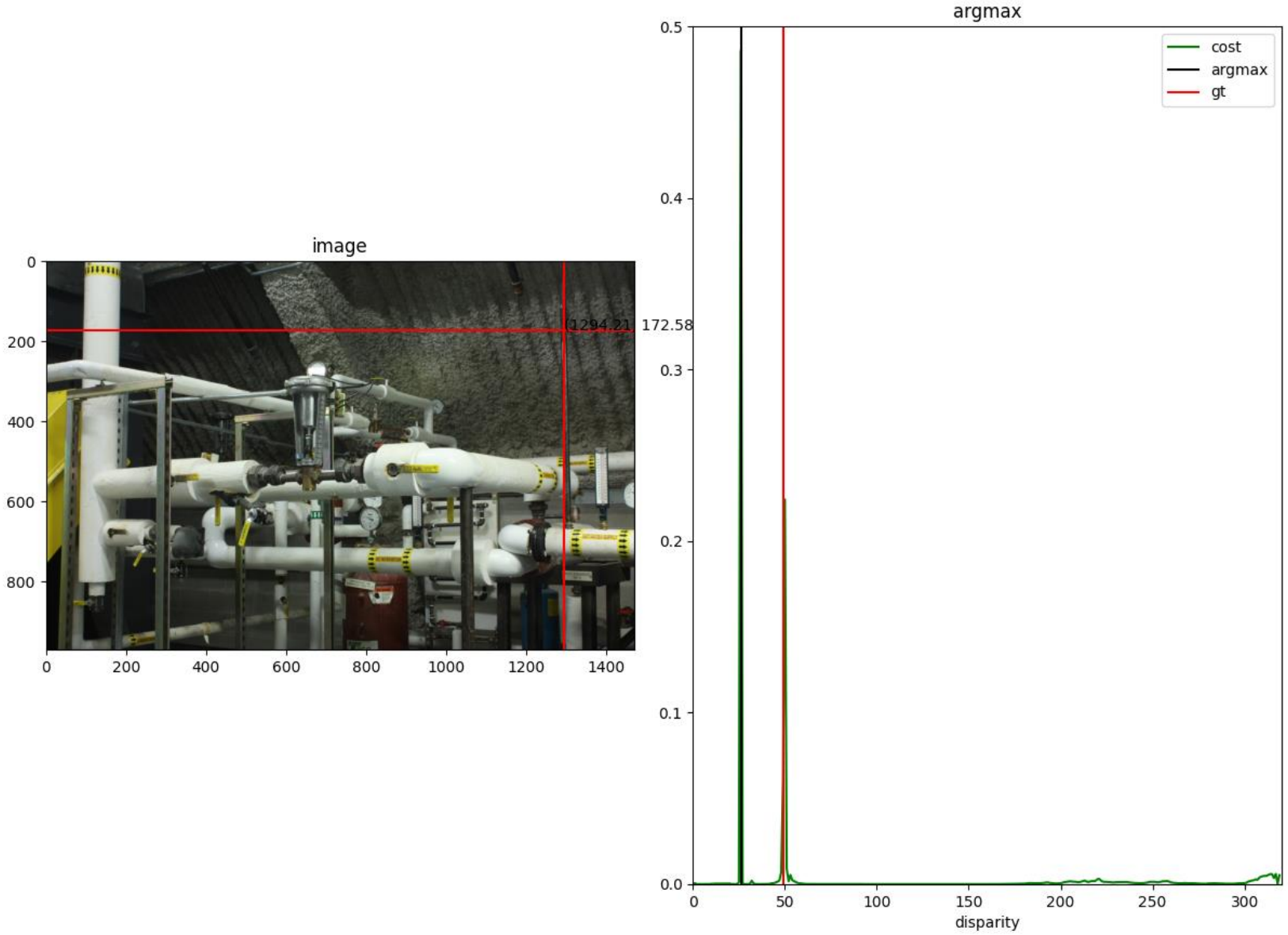
배경



배경

- 넓은 크기의 윈도우를 사용하는 경우, 가느다란 물체는 윈도우에서 차지하는 비율이 매우 적기 때문에 유사한 픽셀을 찾을 때 무시될 수 있다.

배경



가느다란 물체의 경우 직접 설계하고 훈련시킨 신경망에서의 유사도.
두 번째 그림은 첫 번째 그림의 빨간 선이 교차하는 픽셀에서의 유사도 그래프.
(x축은 disparity, y축은 확률을 나타냄.)

배경

- Large scale window에서도 단 하나의 물체에만 집중할 수 있다면 문제점을 해결할 수 있음.

배경

- Edge에 적응적으로 특징을 추출하는 네트워크는 edge를 경계로 배경이나 주변 사물 배치로부터 특징이 섞여 들어가는 것을 억제하려는 목적으로 설계 중.

- Stereo matching의 cost computation 단계에 사용.

Method

4색 정리

- 평면을 유한 개의 부분으로 나누어 각 부분에 색을 칠할 때, **서로 맞닿은 부분**을 다른 색으로 칠한다면 네 가지 색으로 충분하다는 정리.

4색 정리

- 라벨의 집합 L = {a, b, c, d}
- 픽셀에서 라벨로의 함수 F: (p) → L
- 픽셀에서 intensity로의 함수 G: (p) → R^n
- 픽셀에서 이웃 픽셀의 집합으로의 함수 N(p)

- 픽셀의 intensity를 다음 집합을 사용하여 업데이트
- $G^{n+1}(p) \leftarrow \{G^n(e) \mid e \in N(p) \wedge F(e) = F(p)\}$

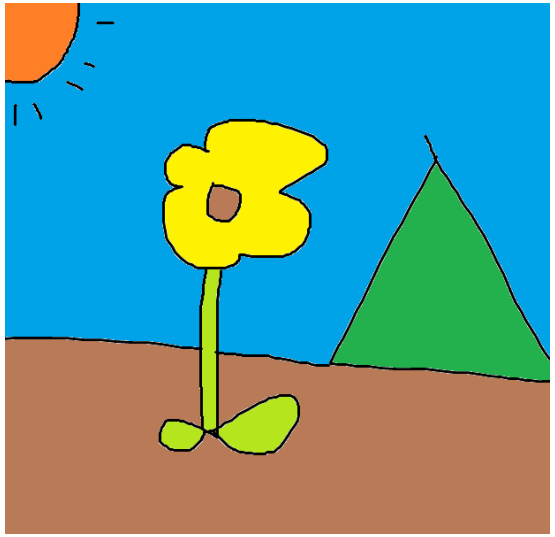
4색 정리

- 구현 (픽셀에서 라벨로의 함수 F 역할을 하는 신경망)
  - U-net 사용
    - 입력 텐서 shape: (batch_num, height, width, 3), RGB 3채널
    - 출력 텐서 shape: (batch_num, height, width, 4)
    - flying things dataset으로 훈련.

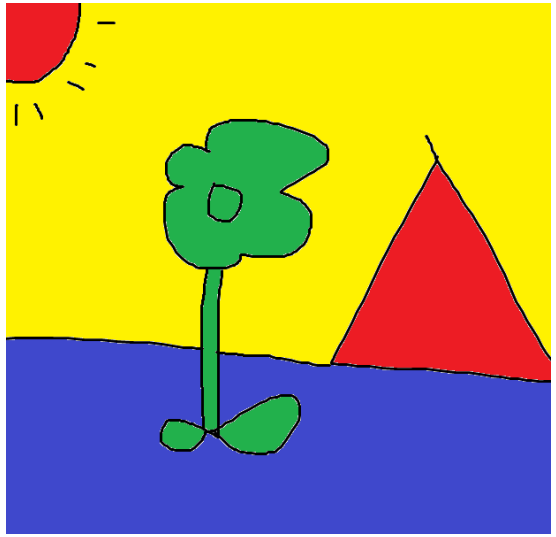
  - Loss:
    - Flying things disparity GT에 라플라시안 필터를 적용하여 엣지 이미지를 만들고 cv2.connectedComponent 사용.
    - Component의 집합 C = {1st, 2nd, 3rd, … }
    - Component에서 neighbor component의 집합으로의 함수 N(c)
    - loss = $\sum_{c\in C}\sum_{c_i\in N(c)}\sum_{q\in c_i}\|E(F(p)|p\in c), F(q)\|$
      - (p, q는 픽셀, F는 픽셀에서 라벨로의 함수, E는 평균)

4색 정리

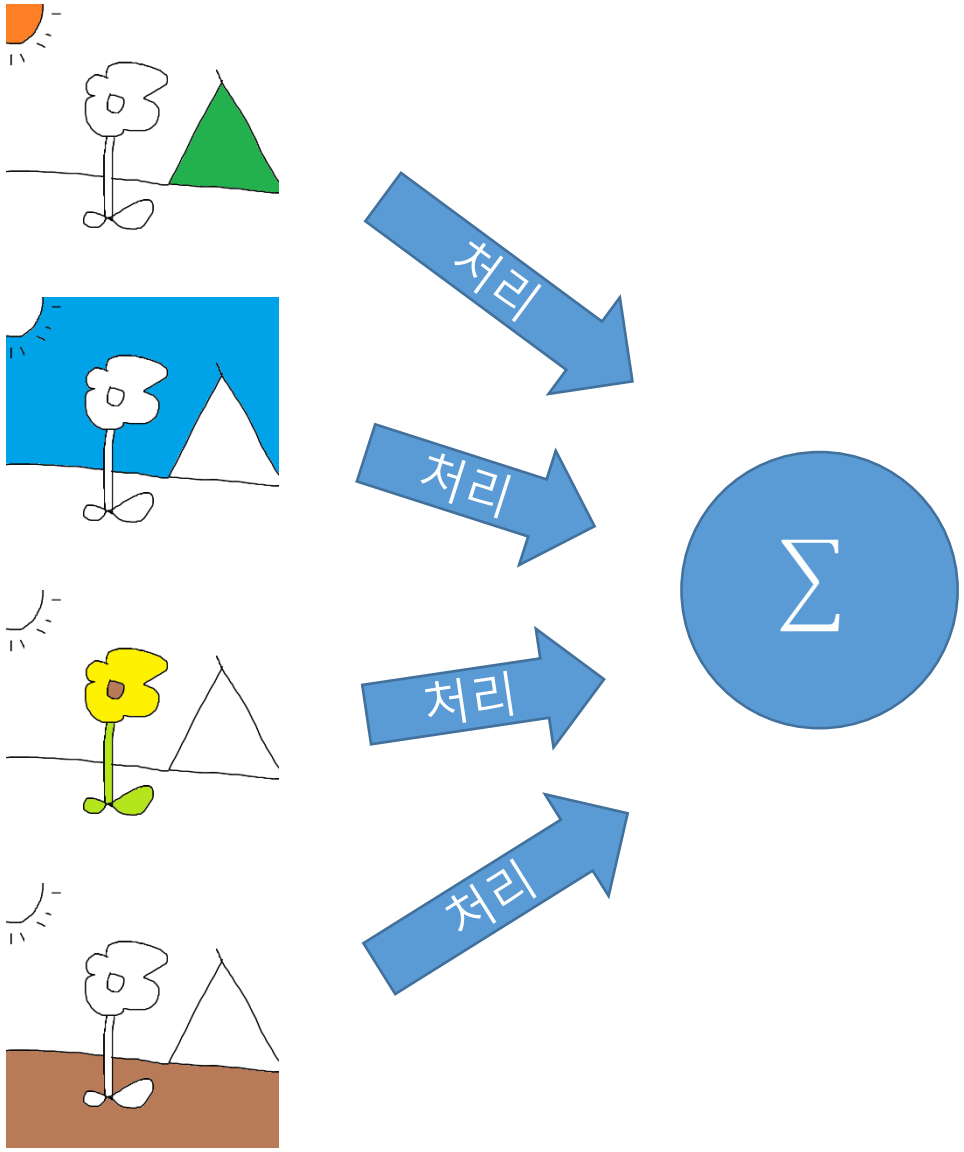
- 구현 (픽셀 intensity 업데이트)



이미지



마스크



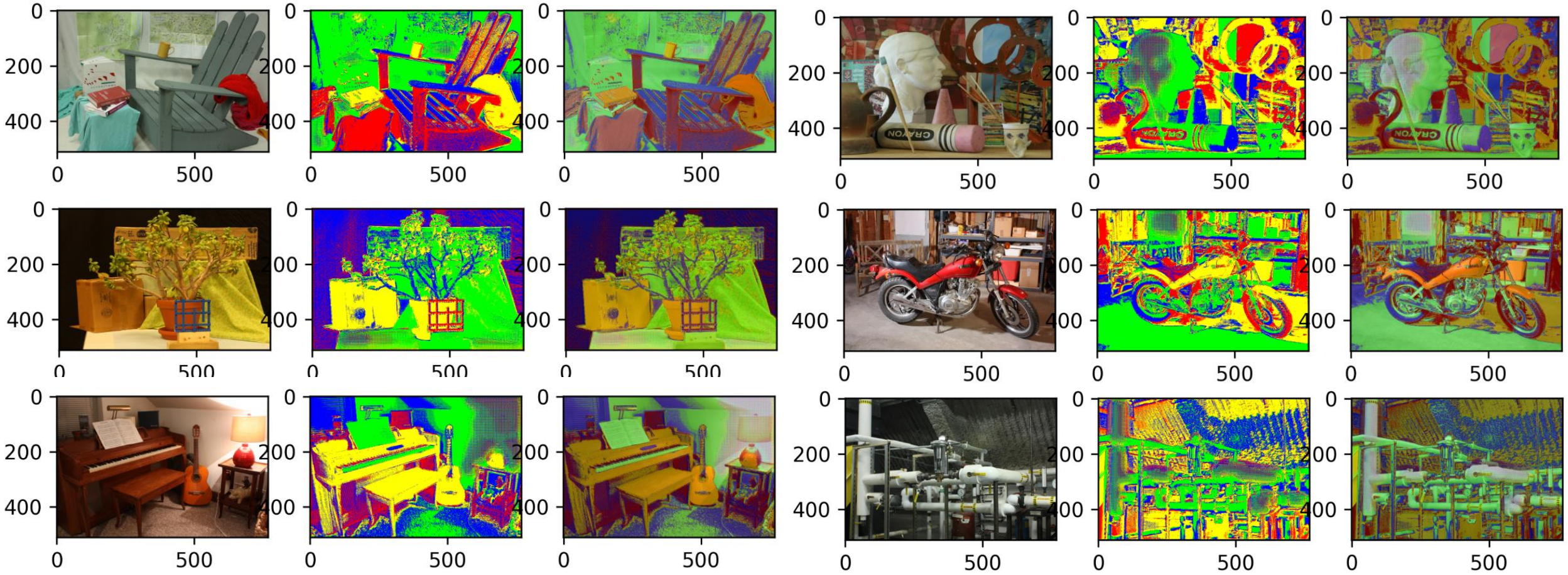
Masked 이미지

4색 정리

- 구현 (픽셀 intensity 업데이트)
  - U-net의 output 텐서 (batch_num, height, width, 4)를 마스크로 사용.
  - 기존 이미지 텐서 (batch_num, height, width, feature_num)을 4개로 복제한 후 각각의 텐서에 마스크 필터 적용
    - image: (batch_num, height, width, feature_num) → (batch_num, 4, height, width, feature_num)
    - mask: (batch_num, height, width, 4) → (batch_num, 4, height, width, 1)
      - 전치 연산 및 차원 확장 연산
    - Masked image = (batch_num, 4, height, width, feature_num) * (batch_num, 4, height, width, 1)
      - Hadamard product (broadcasting 룰 적용)
  - 마스크 처리된 이미지를 각각 처리한 후 결과를 합침. (미구현)
    - (batch_num, 4, height, width, feature_num) → (batch_num * 4, height, width, feature_num) → 처리 → (batch_num, 4, height, width, feature_num) → (batch_num, height, width, feature_num)

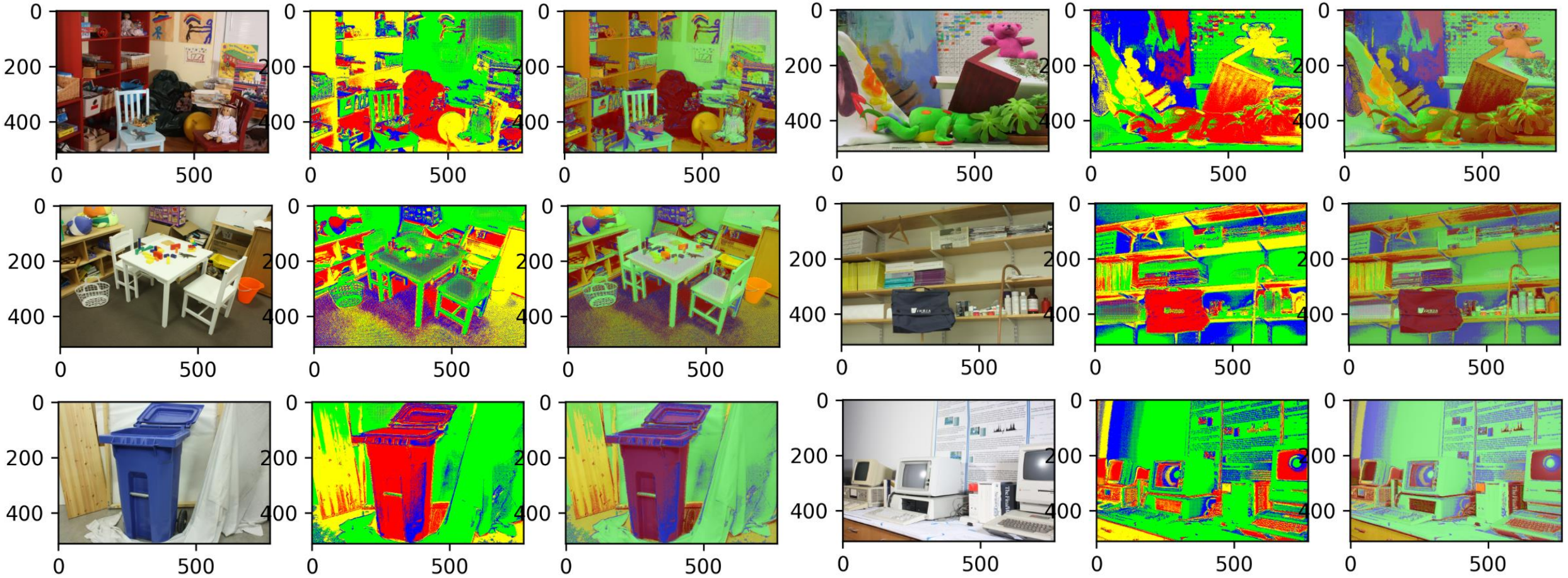
4색 정리

- 분할 결과 (순서대로 원본 이미지, 색칠 이미지, 원본 이미지와 색칠 이미지의 alpha blending)



4색 정리

- 분할 결과



4색 정리

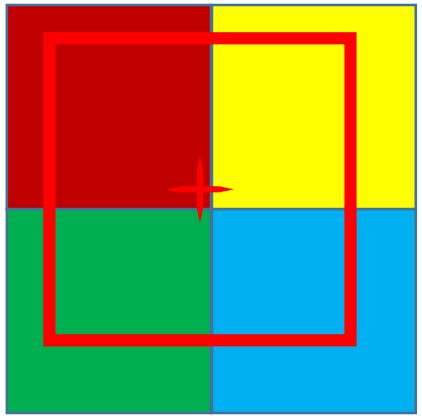
- 분할 품질이 좋지 않아 다음 단계는 진행하지 않음.

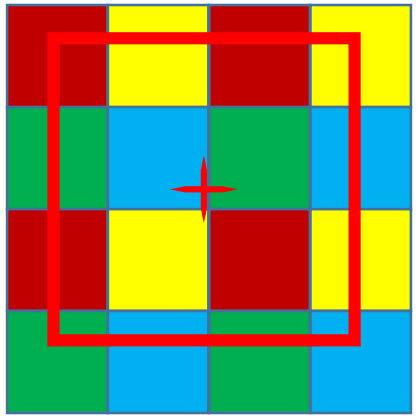
4색 정리

- 분할 품질이 안 좋은 이유에 대한 추측
  - 신경망이 문맥을 학습해야 하지만, 트레이닝 데이터 셋과 테스트 데이터 셋의 특성 차이가 매우 다름
  - U-net은 특정 종류의 오브젝트를 특정 라벨로 대응시키려는 목적으로 설계된 네트워크이지만, 4 coloring은 오브젝트가 무엇인지는 관계없이 오브젝트의 배치에 의해 라벨을 결정하는 문제이므로, U-net은 이 문제를 해결하기에 적합한 네트워크가 아님.

4색 정리

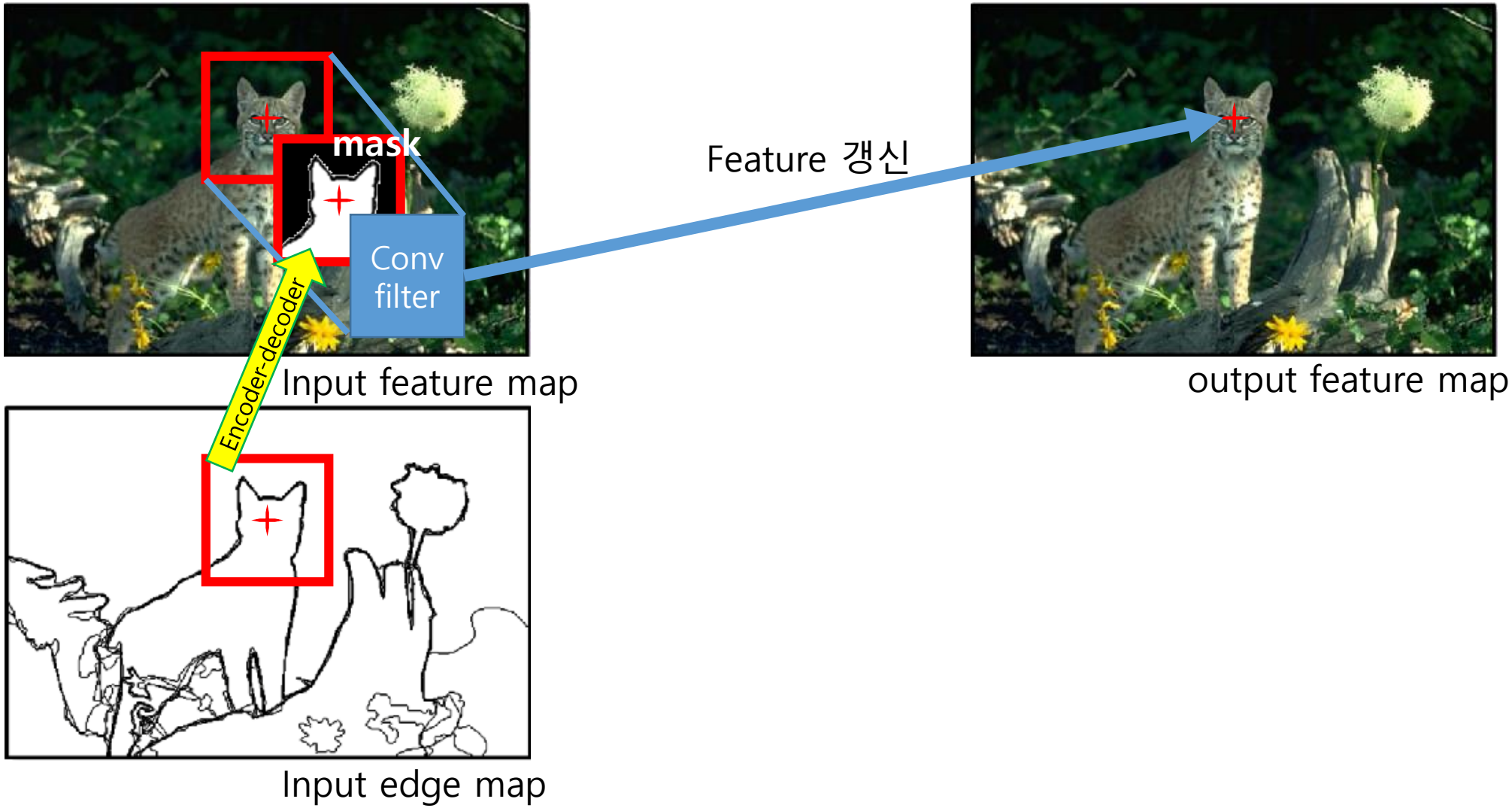
- 한계





- 라벨의 수가 적기 때문에 층이 깊어지며 receptive field의 크기가 커질수록 같은 라벨이지만 다른 오브젝트에 속하는 픽셀이 동시에 포함되는 경우가 많아질 것이다.

Encoder-decoder 구조



Encoder-decoder 구조

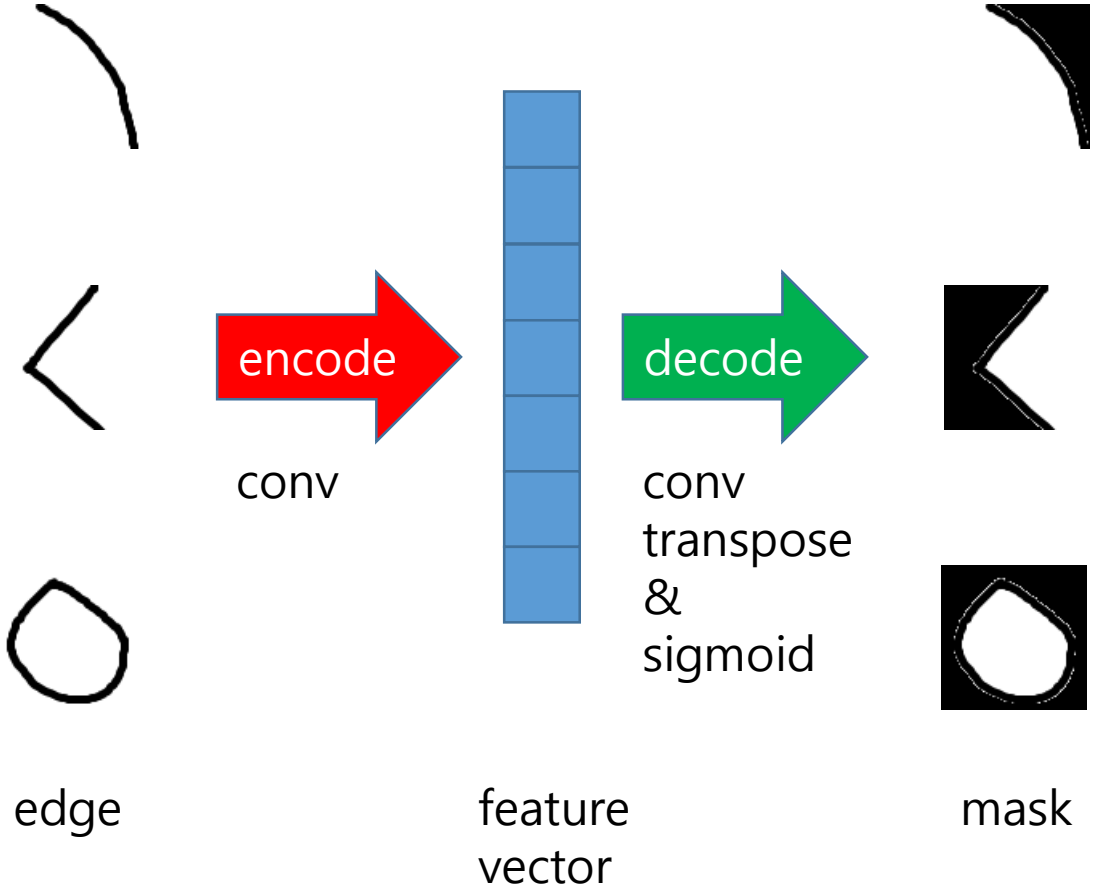
- 미리 획득한 엣지 이미지를 이용하여 마스크 생성
- 컨볼루션 필터를 적용하기 전에 마스크를 적용
- 컨볼루션 연산이 tensorflow에서 atomic 연산이므로 연산을 직접 구현해야 할 것으로 생각됨.

Encoder-decoder 구조

- 수식(?)
  - O(p)는 original 이미지의 픽셀 intensity
  - E(p)는 edge 이미지의 픽셀 intensity
  - N(p)는 픽셀 p의 이웃
  - Mask는 픽셀의 intensity의 집합에서 마스크를 생성하는 함수

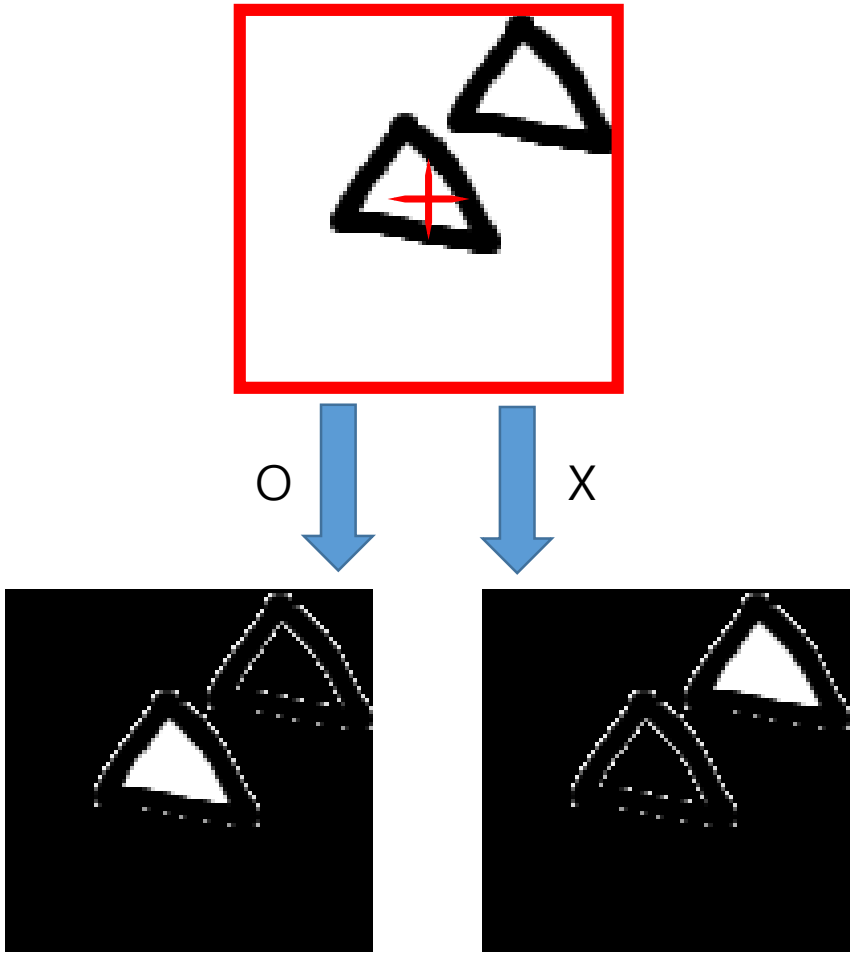
  - 픽셀의 intensity를 다음 집합을 사용하여 업데이트
    - $O^{n+1}(p) \leftarrow \{O^n(q) | q \in N(p) \wedge q \notin Mask(\{E(k) | k \in N(p)\})\}$

Encoder-decoder 구조

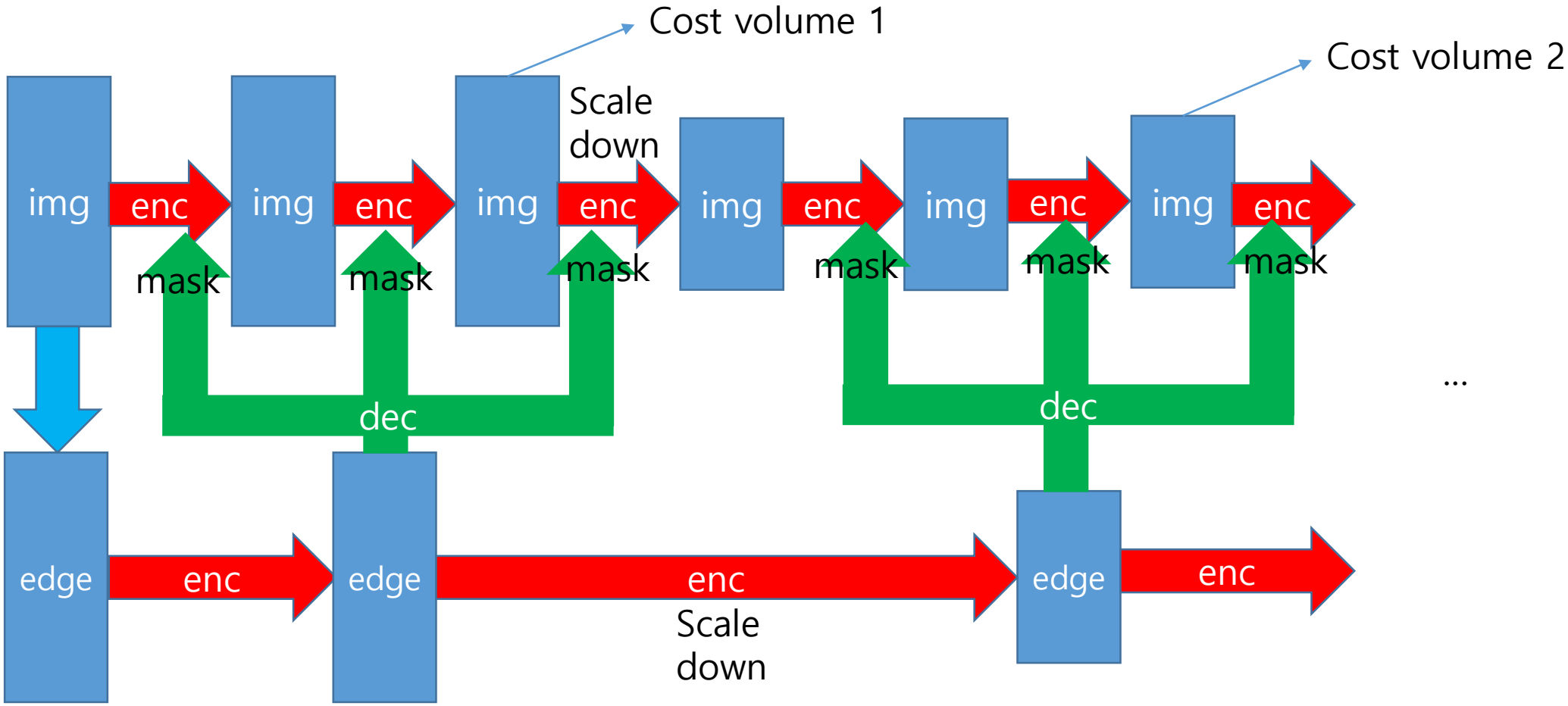


Encoder-decoder 구조

- 네트워크는 edge의 위치도 고려하여 마스크를 만들어야 함.
- 위치 정보가 보존되어야 함.
  - 풀링 사용 X
  - bias 사용
  - height = 1, width = 1 까지 압축 후 디코딩



Encoder-decoder 구조



한계

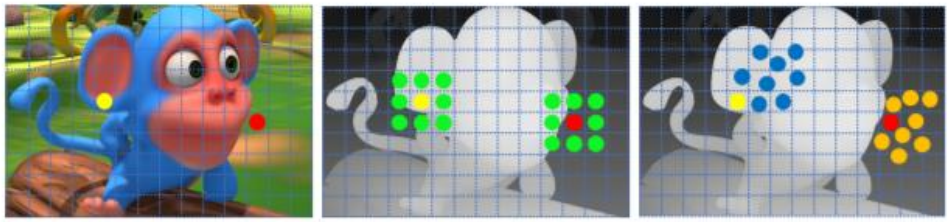
- 오브젝트 A 앞에 오브젝트 B가 있어 edge 이미지에서 오브젝트 A가 분할될 경우 분할된 조각을 매칭할 때 다른 조각의 정보를 사용할 수 없음.

- segmentation에 기반한 특징 추출을 사용하면 위 문제를 해결할 수 있지만, 라벨 개수에 제한이 있는 문제에만 사용 가능할 것으로 생각됨.

Related works

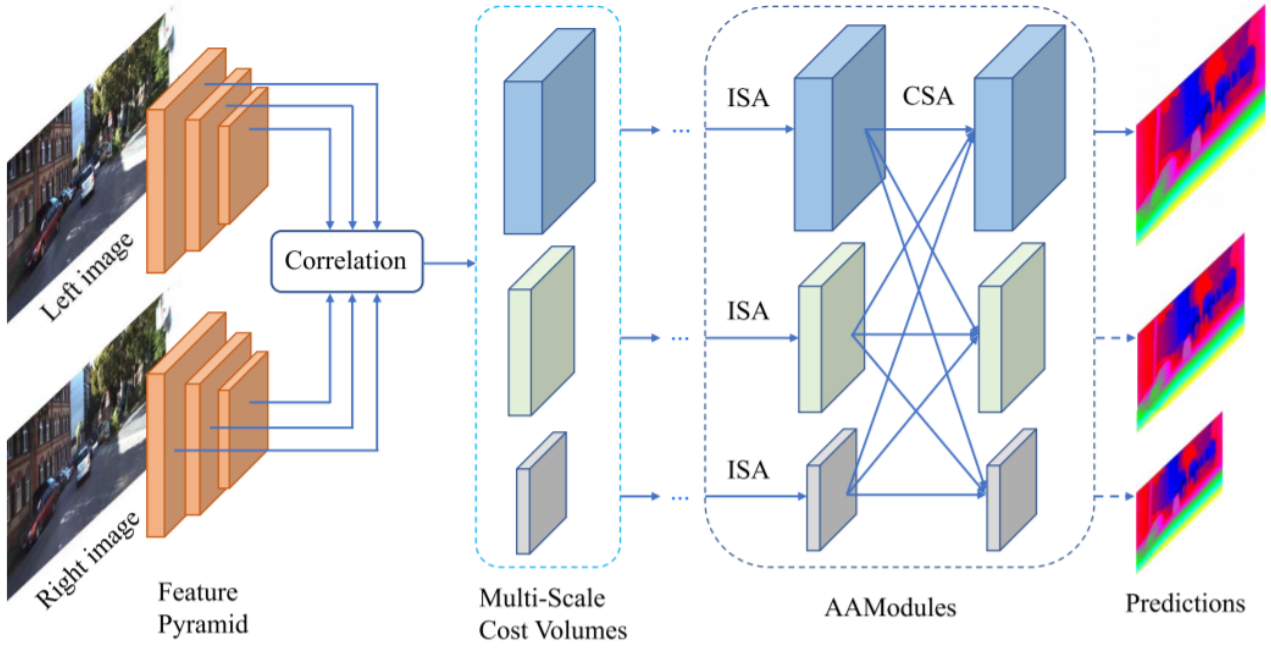
Adaptive Aggregation Network for Efficient Stereo Matching

- https://arxiv.org/pdf/2004.09548.pdf



(a) (b) (c)

Figure 1: Illustration of the sampling locations in regular convolution based cost aggregation methods and our proposed approach, where the yellow and red points represent the locations for aggregation. (a) left image of a stereo pair. (b) fixed sampling locations in regular convolutions, also the aggregation weights are spatially shared. (c) adaptive sampling locations and position-specific aggregation weights in our approach. The background in (b) and (c) is ground truth disparity.



Adaptive Aggregation Network for Efficient Stereo Matching

- 주요 특징
  - Cost volume aggregation 단계(4d volume, 3d-conv)에서 deformable convolution을 사용하여 비용 함수를 갱신. (Intra-Scale Aggregation)
  - 여러 스케일의 코스트 볼륨을 만들고 Cross-Scale Aggregation을 사용.
- Deformable convolution의 특징 상 cost computation 단계에서는 사용할 수 없고, cost aggregation 단계에서만 사용할 수 있을 것으로 생각됨. (다음 부분에서 좀 더 자세히 설명)

Deformable Convolutional Networks

- https://arxiv.org/pdf/1703.06211.pdf

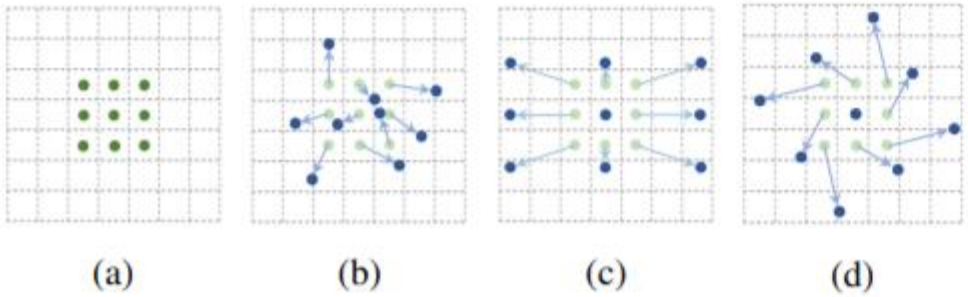


Figure 1: Illustration of the sampling locations in 3×3 standard and deformable convolutions. (a) regular sampling grid (green points) of standard convolution. (b) deformed sampling locations (dark blue points) with augmented offsets (light blue arrows) in deformable convolution. (c)(d) are special cases of (b), showing that the deformable convolution generalizes various transformations for scale, (anisotropic) aspect ratio and rotation.

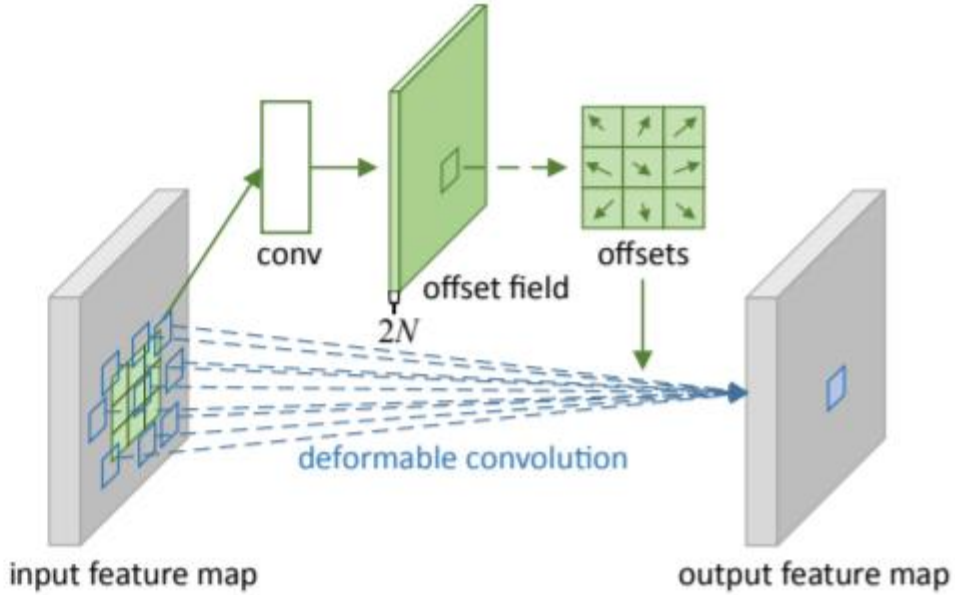


Figure 2: Illustration of 3×3 deformable convolution.

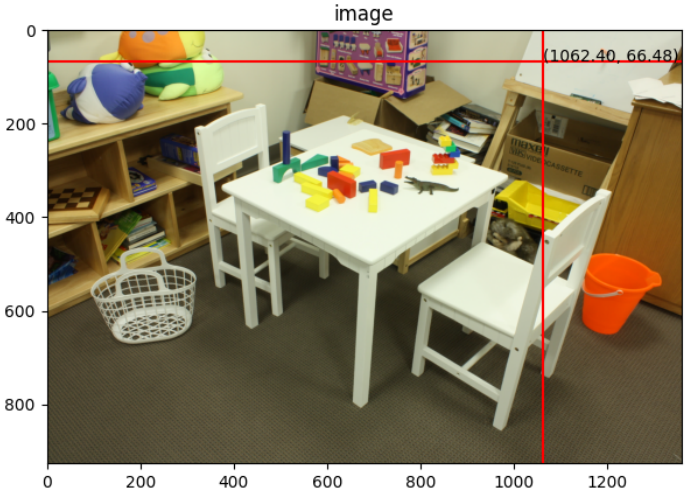
Deformable Convolutional Networks

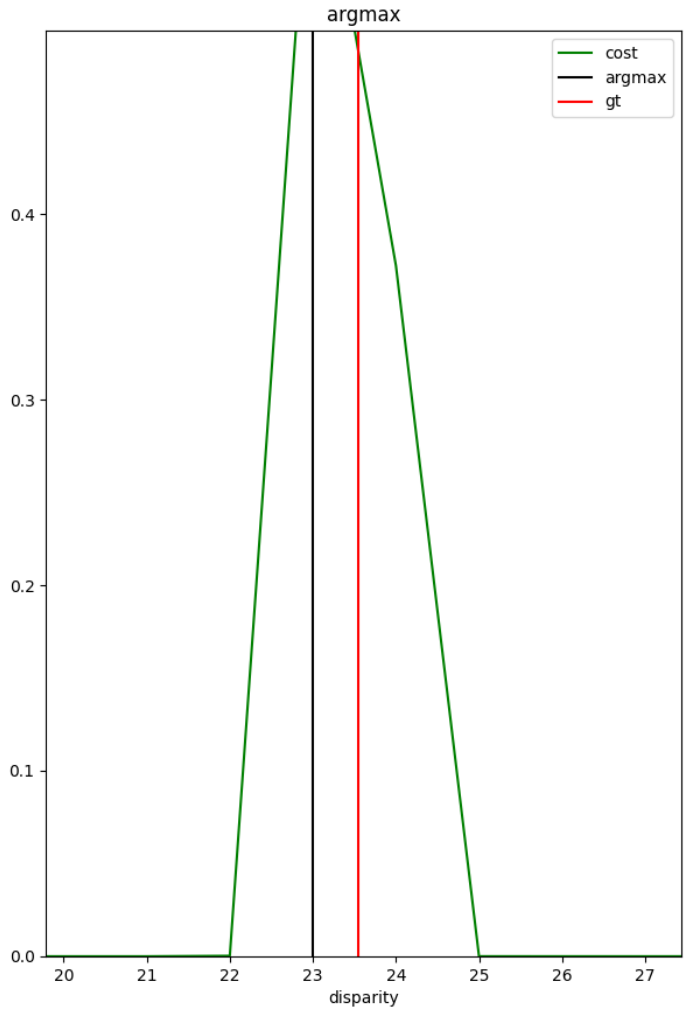
- 처음엔 일반 컨볼루션을 사용하여 오프셋을 계산한 다음, 오프셋 만큼 필터를 변형시킨 후 변형된 필터를 사용하여 컨볼루션 연산을 하는 네트워크

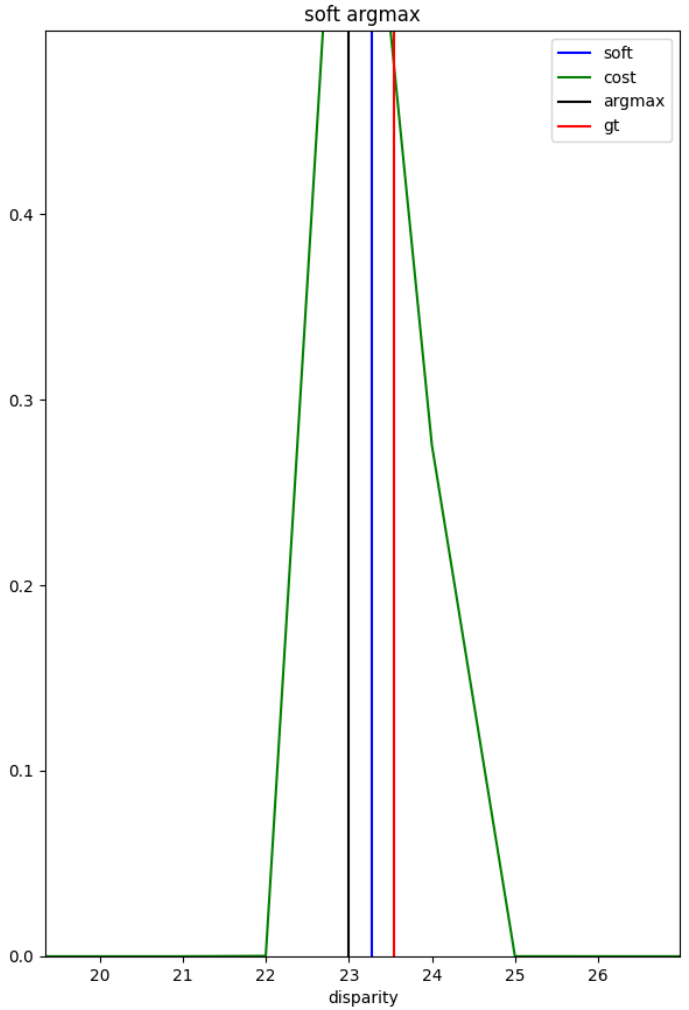
- Cost computation 단계에 적용할 수 없는 이유
  - Deformable convolution을 실제로 동일한 위치를 나타내는 right 이미지의 픽셀과 left 이미지의 픽셀에 각각 적용했을 때 픽셀 offset이 거의 같게 계산되어야 하기 때문으로 생각됨.

Soft Argmax의 대안 제시

Soft argmax의 이상적인 동작 조건







Soft argmax의 이상적인 동작 조건

However, compared to the argmin operation, its output is influenced by all values. This leaves it susceptible to multi-modal distributions, as the output will not take the most likely. Rather, it will estimate a weighted average of all modes. To overcome this limitation, we rely on the network's regularization to produce a disparity probability distribution which is predominantly unimodal. The network can also pre-scale the matching costs to control the peakiness (sometimes called temperature) of the normalized post-softmax probabilities (Figure 2). We explicitly omit batch normalization from the final convolution layer in the unary tower to allow the network to learn this from the data.

GC-net

- soft argmin은 봉우리가 여러 개인 확률분포에 취약하며, 네트워크가 파라미터를 조정해서 봉우리를 되도록 하나로 만들어야 함.
- 그것을 위해서 마지막에 값을 극단적으로 조정하는 것이 가능하도록 마지막의 배치 정규화 계층을 생략하였음.

Soft argmax의 이상적인 동작 조건

- 확률 분포에서 단 하나의 봉우리(argmax)가 true disparity의 -0.5~0.5 범위 내에서 형성.
- 봉우리를 제외한 나머지 분포는 0
- **봉우리의 정상에서 true disparity 쪽으로는 (반대쪽보다는) 원만하게 변하고 반대쪽으로는 급격히 변해야 함.**

Soft argmax의 대안

- Soft argmin은 모든 봉우리의 가중 평균을 취하는 방식이기 때문에 multi-modal 분포에 취약함.

- 가장 높은 봉우리 언저리의 값만 regression에 사용하면 multi-modal 분포에서도 문제없이 작동할 수 있을 것으로 생각됨.

Soft argmax의 대안

- 봉우리의 argmax인 지점으로 픽셀 단위로 disparity를 구할 수 있음.
- 봉우리 주변에서의 높이 변화로 residual disparity를 예측할 수 있음. 기울어진 정도를 -0.5~0.5사이의 값으로 맵핑. (함수 R)

- True disparity = R(argmax의 주변 값) + argmax
  - 여기서 $-0.5 \leq R \leq 0.5$ (residual 이므로)

- 다음 식으로 residual disparity를 구하는 신경망 학습 가능.
  - Residual = True disparity – Argmax

Soft argmax의 대안

- 구현
  - 기울기 계산
    - 각각이 특정 기울기 값을 나타내는 벡터들과 내적한 다음 유사도를 구함 => 컨볼루션 신경망을 이용하여 학습하도록 함.
    - 내적하는 벡터의 길이가 길수록 봉우리에서 더 멀리 떨어진 값들을 residual regression에 반영.
  - Argmax 값이 True disparity의 -0.5~0.5 범위 내에 있지 않을 때는 오차를 전파하지 않음.

Soft argmax의 대안

- 구현 예시
  - Input: Softmax를 적용하기 직전 logit 혹은 softmax 적용 후의 값
    - Shape: (batch_num, h, w, disparity) -> (batch_num, h, w, disparity, 1) -> (batch_num * h * w, disparity, 1)
  - Disparity 축으로 1차원 컨볼루션 & 배치 정규화
    - (batch_num * h * w, disparity, 1) -> (batch_num * h * w, disparity, feature)
  - reshape 후, Fully connected layer
    - (batch_num * h * w * disparity, feature) -> (batch_num * h * w * disparity, 1)
  - reshape 후, Sigmoid – 0.5 (-0.5~0.5 범위로 맞추기 위함)
  - Output: (batch_num, h, w, disparity)

Soft argmax의 대안

- 구현 예시
  - Batch_num=0, height=h1, width=w1에서의 최종disparity는 다음 식으로 구함.
    - argmax = np.argmax(logit[0, h1, w1, :])
    - res = residual[0, h1, w1, argmax]
      - residual은 이전 슬라이드의 output 텐서
    - disparity = argmax + res