- 2025 여름 세미나 -

Object Change Detection

Pusan National University
Dept. of Electronics Engineering
Computer Vision & Signal Processing Lab.

Presented By
이우진

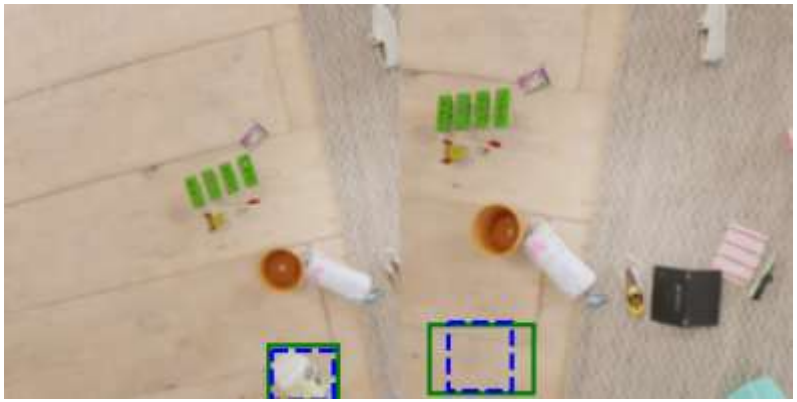
Outline

- Background
- The Change You Want to See (Now in 3D) [ICCV workshop 2023]
- Improving Zero-Shot Object-Level Change Detection by Incorporating Visual Correspondence [WACV 2025]

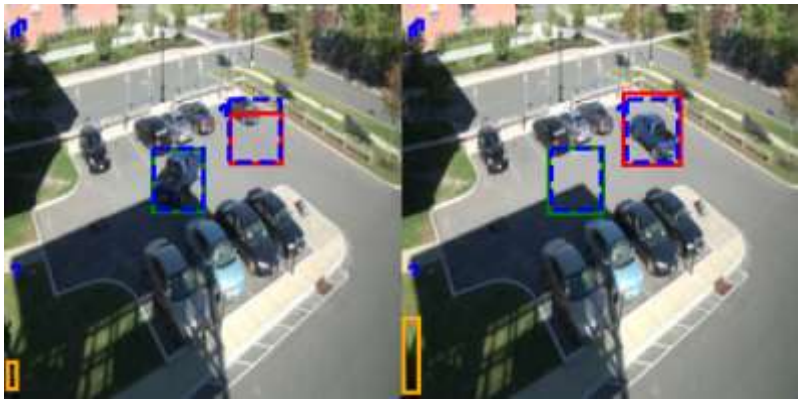
Background

Background

- Object Change Detection
  - 분야 소개
    - 두 이미지에 대해 객체가 변화된 부분을 탐지하는 분야
    - 객체의 생성, 사라짐, 위치 변화 등 다양한 유형의 변화 감지를 목표로 함
      - 생성, 사라짐: 객체가 한 이미지에는 있고 다른 이미지에는 없는 경우
      - 위치 변화: 한 이미지와 다른 이미지에 대해 객체의 위치가 바뀐 경우



생성, 사라짐 예시



위치 변화 예시

Background

- Object Change Detection
  - 연구 방향
    - 초기에는 2D 이미지 상에서만 객체의 변화를 탐지하는 연구가 시작됨
      - 초기 모델은 2D 회전된 이미지 위주로 학습해 다각도 이미지로 학습이 이루어지지 않음
      - 다각도 이미지에서 성능 저하 발생
    - 이후 3D 정보를 활용한 연구를 통해 다각도 시점 변화에 강인한 변화 탐지가 가능



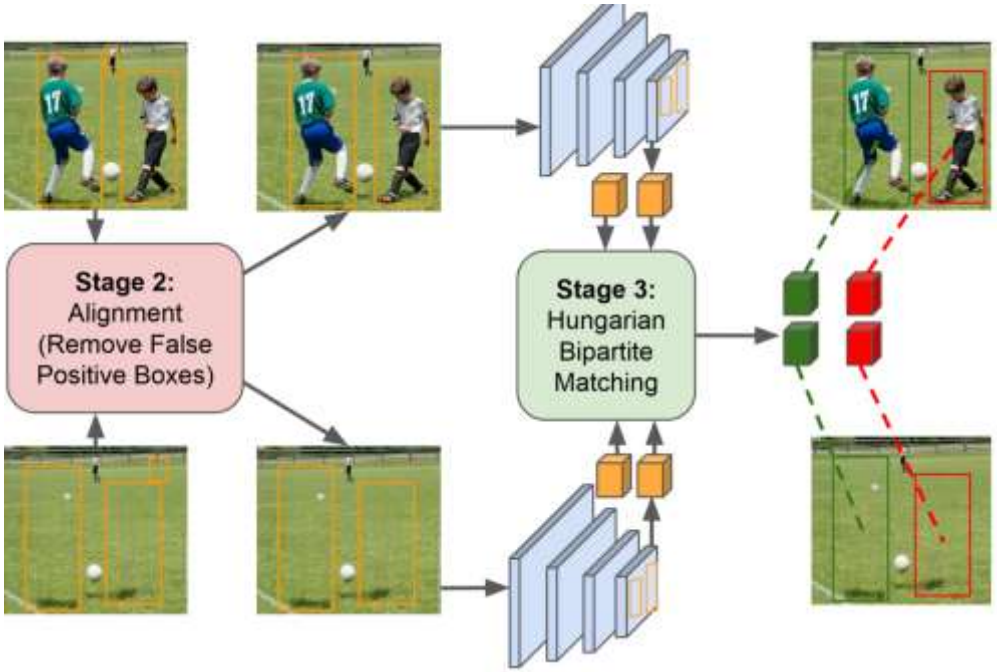
< 초기 2D모델 학습 데이터셋 예시 >



< 3D모델 inference 예시 >

Background

- Object Change Detection
  - 최신 연구 동향
    - 두 이미지 간 변화가 발생한 영역을 객체 단위로 대응시키는 방식을 도입하여 변화 탐지 성능을 효과적으로 향상시킴



< 최신 모델의 변화객체 대응 모식도 >

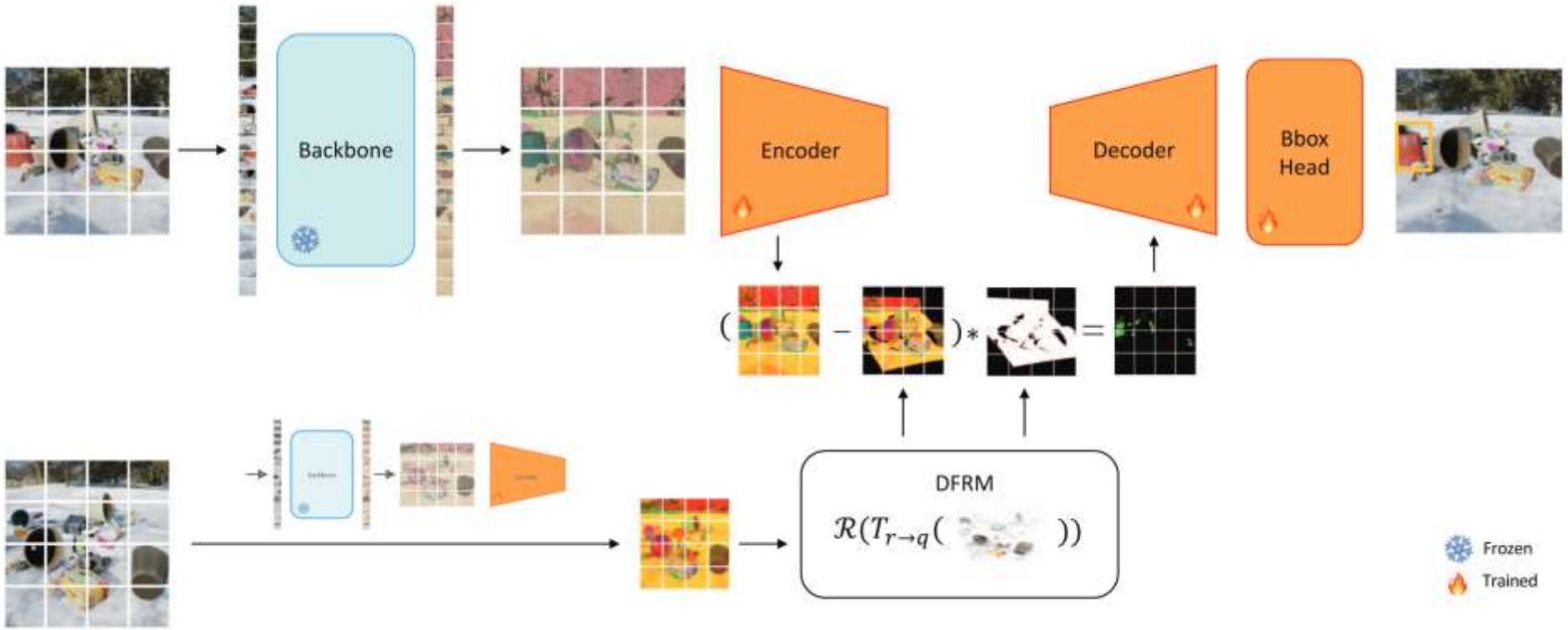
Background

- Object Change Detection
  - 논문 선정 이유
    - The Change You Want to See (Now in 3D) [ICCV workshop 2023]
      - 3D 모델의 다각도 이미지에 대한 변화 탐지 원리 소개
    - Improving Zero-Shot Object-Level Change Detection by Incorporating Visual Correspondence [WACV 2025]
      - 최신 연구 동향 파악 및 탐지성능 개선 원리 소개

The Change You Want to See (Now in 3D) [ICCV workshop 2023]

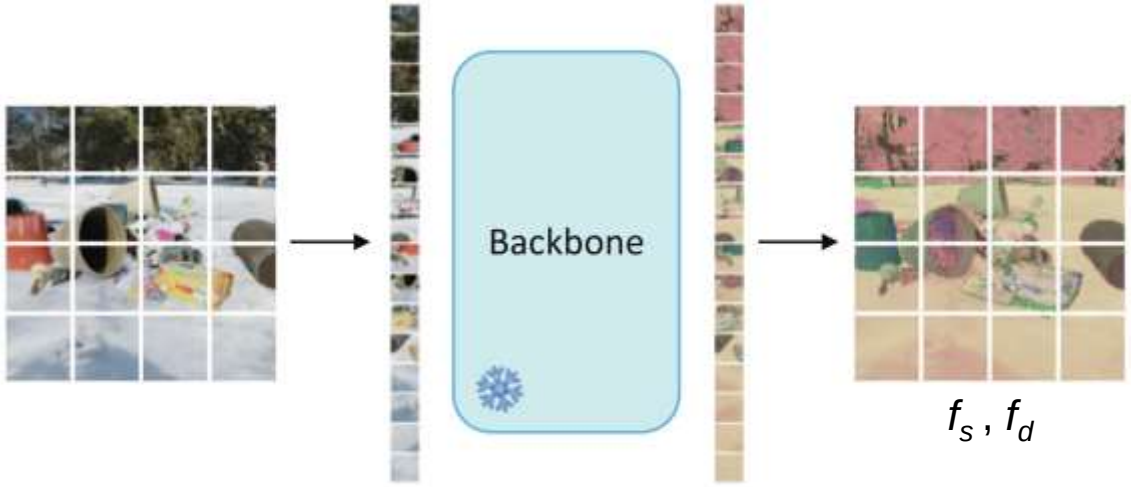
Method

- Overview
  - 비교할 두 이미지 I_1, I_2의 특징에 대해 다른 한쪽의 특징을 다른 한쪽의 특징맵에 DFRM으로 정렬시켜 변화량을 비교



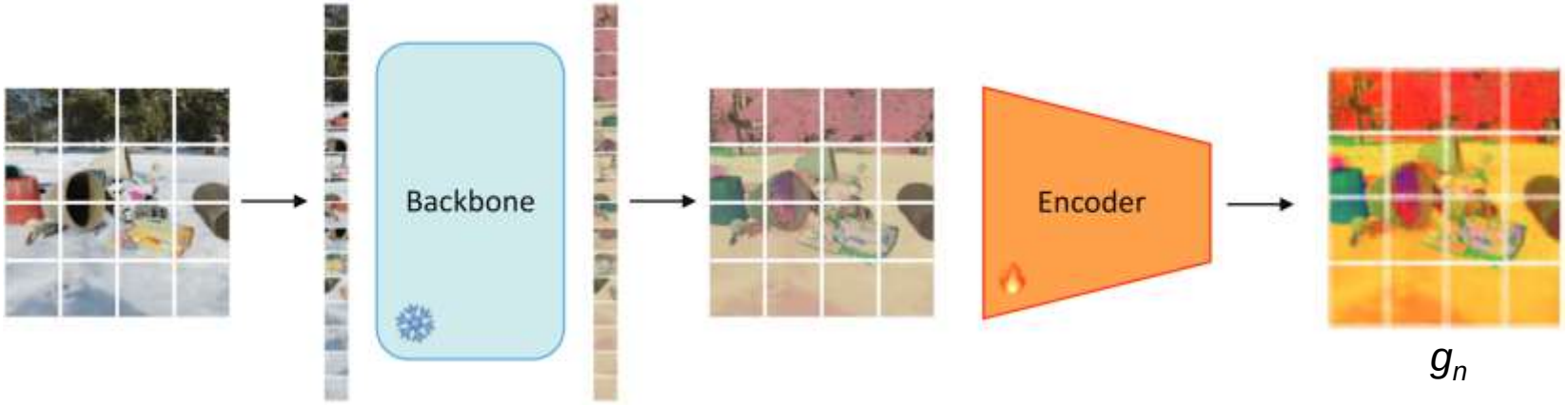
Method

- Backbone
  - 비교할 두 이미지를 각각 Frozen visual backbone을 통해 특징 f_s, f_d를 출력
    - f_s, f_d는 각각 shallow, deep feature
    - DINO 모델 가중치로 초기화된 ViT-B/8를 백본으로 사용



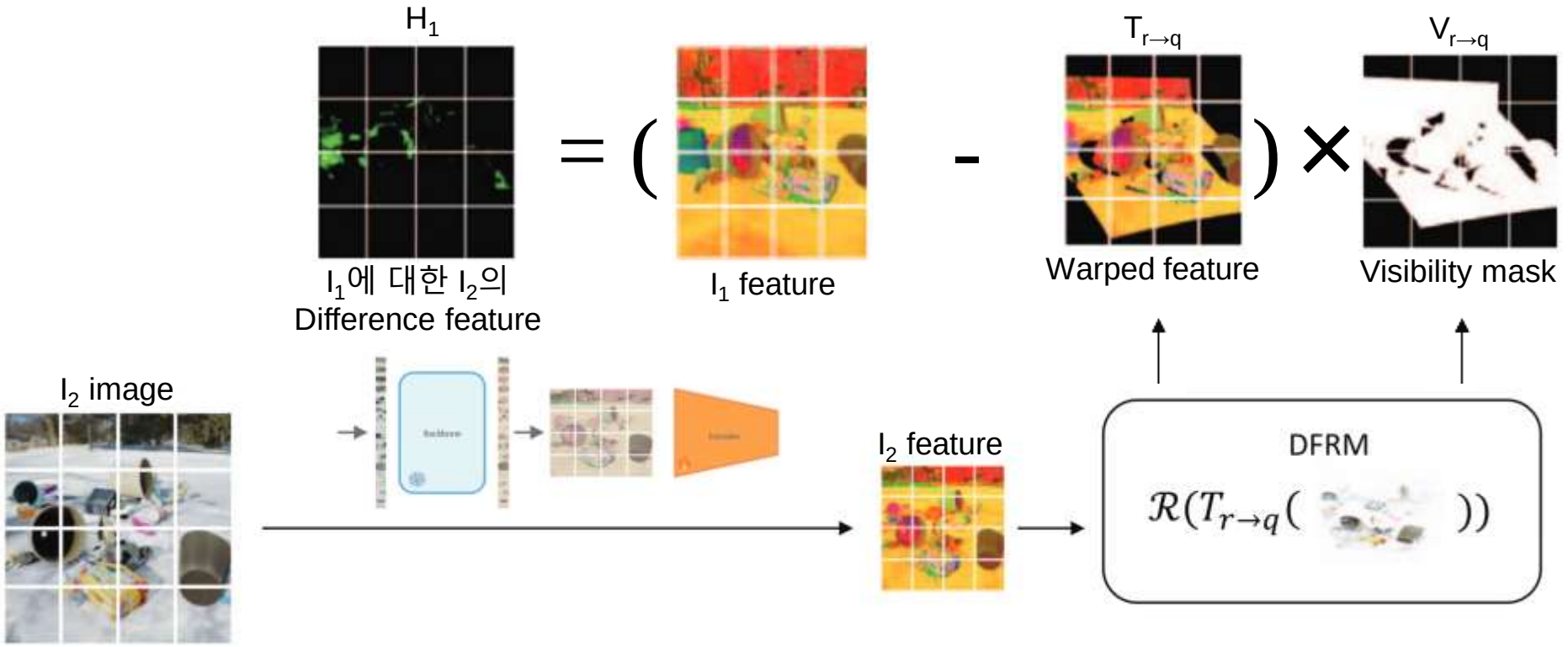
Method

- Encoder
  - f_d를 unet encoder에 넣어서 각 레이어 n마다 특징 g_n을 출력
    - 결과적으로 멀티 해상도 특징맵 $G = \{ f_d, g_1, g_2, g_3, g_4 \}$를 이미지 I_1, I_2에 대해 각각 출력



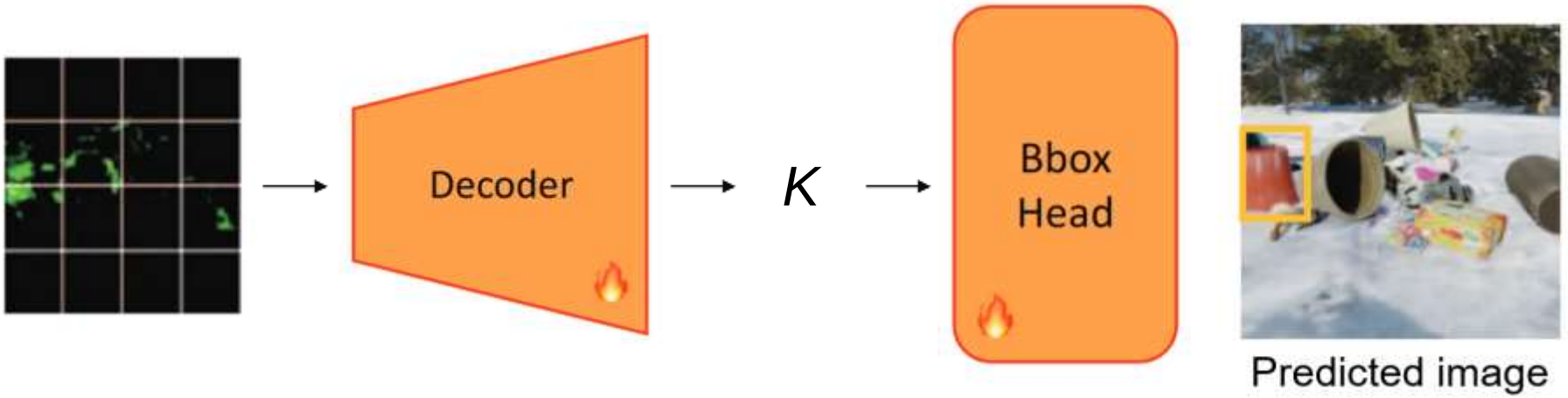
Method

- DFRM
  - DFRM을 사용해 각 공간 해상도에서 feature map warp를 얻음 ($T_{r\to q}$)
  - 그 후 가려지거나 새롭게 나타난 영역을 마스킹 ($V_{r\to q}$)
  - 최종적으로 feature 간의 차이를 얻음 ($H_1 = V_{2\to 1}(G_1 - T_{2\to 1}(G_2))$)



Method

- Decoder and Bbox head
  - 이후 feature map 집합 H_1, H_2를 unet 디코더로 디코딩하여 K_1, K_2를 생성
  - 마지막으로 f_s와 K를 concat하여 CenterNet 헤드에 입력하여 변화된 영역에 대해 바운딩 박스를 생성
    - f_s, K를 concat하는 동기: 얕은 feature f_s는 위치 정보를 잘 포착하고 깊은 feature K는 의미 정보를 잘 포착함



Method

- DFRM
  - 1단계: Estimate a 3D linear transformation
    - 대응점 추출기 C를 사용해 이미지 I_r, I_q에서 각각 n개의 대응점 P^r, P^q 추출 후 RANSAC으로 outlier 제거
      - C는 SuperGlue를 사용
    - 깊이 추정기 D를 사용하여 깊이 맵 추정
      - D는 ZoeDepth를 사용
    - 변환추정을 위해 대응점 $(x_j , y_j) \in P^r, P^q$ 를 역투영하여 3D point cloud로 표현
      - 깊이 정보 d_j를 z값으로 사용, (x , y)를 깊이정보에 맞춰 scaling

$$\hat{P}_j = \begin{bmatrix} d_j x_j & d_j y_j & d_j & 1 \end{bmatrix}^T$$

    - 다음의 closed-form 해를 이용해 r의 대응점을 q로 투영하는 변환 행렬 T 추정

$$\hat{P}^{qT} = T_{r\to q} \cdot \hat{P}^{rT} \longrightarrow T_{r\to q} = \left(\hat{P}^{r+} \hat{P}^{q}\right)^T$$

Method

- DFRM
  - 2단계: Lift the features to 3D and warp
    - 추정한 T를 이용해 r(또는 q)의 3d point를 q(또는 r)의 3d point로 정렬

$$\begin{bmatrix} x_j' & y_j' & z_j' & k \end{bmatrix}^T = T_{r \to q} \begin{bmatrix} d_j x_j & d_j y_j & d_j & 1 \end{bmatrix}^T$$

변환 후 3d point　　변환 전 3d point

Method

- DFRM
  - 3단계: Differentiable feature rendering
    - 생성된 3D point cloud에 대해 renderer R을 이용해 2D로 랜더링함
    - RGB 색상을 렌더링하는 대신 각 점에 대해 c차원 feature 벡터를 렌더링 함
    - 이를 위해 differentiable point cloud renderer를 사용
      - 장점1: small neighbourhood problem 문제 해결
        - ✓Feature 포인트가 소수의 픽셀로만 투사되는 문제를 해결
          - 소수의 픽셀만 투사되면 손실함수가 소수의 영역에 대해서만 계산됨
        - ✓포인트를 조절 가능한 크기의 디스크로 splatting함
          - 디스크 형태로 splatting하여 다수의 영역에 픽셀이 생기게 되고 이로 인해 손실함수가 다수의 영역에서 계산되어 학습이 안정화됨
      - 장점2: hard z-buffer problem 해결
        - ✓하나의 픽셀에 여러 3D 포인트가 투사될 때 가장 가까운(z가 작은) 한 포인트만 살아남고 나머지는 완전히 무시되는 현상
        - ✓논문에서 쓴 renderer는 k개의 가장 가까운 포인트의 효과를 누적함으로써 해결

Method

- DFRM
  - 3단계: Differentiable feature rendering
    - Small neighbourhood problem 문제 해결
      - 3D 포인트 p_i의 렌더링된 중심 p_i^c에 대해 픽셀 l_{xy}에서의 영향력을 나타낸 함수
        - ✓거리 r안에서만 p_i의 영향력이 유효함
        - ✓픽셀의 거리가 중심에서 멀어질수록 영향력이 약해짐
      - 3d 포인트의 렌더링을 한 점에만 투영하는 것이 아닌 디스크 형태로 중심에 대해 splatting하는 구조

$$N(p_i, l_{xy}) = \begin{cases} 0 & \text{if } d^2(p_i^c, l_{xy}) > r \\ 1 - \frac{d^2(p_i^c, l_{xy})}{M} & \text{otherwise} \end{cases}$$

< 디스크 splatting 렌더링 함수 >

Method

- DFRM
  - 3단계: Differentiable feature rendering
    - hard z-buffer problem 해결
      - 앞에서 구한 렌더링 함수를 기반으로 (m,n)에서 가장 가까운 topk의 겹치는 렌더링을 가중합하는 방식으로 해결함
      - 영향력이 클수록 투명도가 낮다는 원리를 바탕으로 가까이 있는 렌더링의 영향력은 크게, 멀리 있는 영향력은 작게 반영하여 (m,n)에서의 렌더링 합 계산

$$\rho_{i_{mn}} = \mathcal{N}(p_i, l_{mn})$$

$$\bar{F}_{mn} = \sum_{i=1}^{K} \rho_{i_{mn}}^{\gamma} F_i \prod_{j=1}^{i-1} (1 - \rho_{j_{mn}}^{\gamma})$$

< 디스크 splatting 렌더링 함수 >

Experiment

- 데이터셋
  - 기존 2D 모델 데이터셋 외 다른 데이터셋을 추가하여 다각도 이미지에 대한 성능 평가를 진행함
    - KC-3D
      - Train 데이터와 test 데이터로 구성된 다각도 synthetic 이미지 데이터셋
        - ✓Train 데이터 81859쌍 / Test 데이터 4548쌍
    - RC-3D
      - 일반화 성능을 평가하기 위해 테스트 데이터 100쌍으로 구성된 다각도 real 이미지 데이터셋



< KC-3D 데이터셋 예시 >



< RC-3D 데이터셋 예시 >

Experiment

- 성능
  - 2D 모델은 KC-3D로 학습 시 KC-3D 테스트셋에서 높은 성능을 보임
    - 하지만 RC-3D에서는 성능이 낮게 나타나는 것으로 보아 오버피팅된 것으로 판단
  - 3D 모델은 RC-3D에서도 안정적인 성능을 보여 더 나은 일반화 성능을 갖춤
    - Depth 맵 사용과 정렬 알고리즘이 다각도 이미지에 대한 변화 탐지 성능을 올린것으로 판단

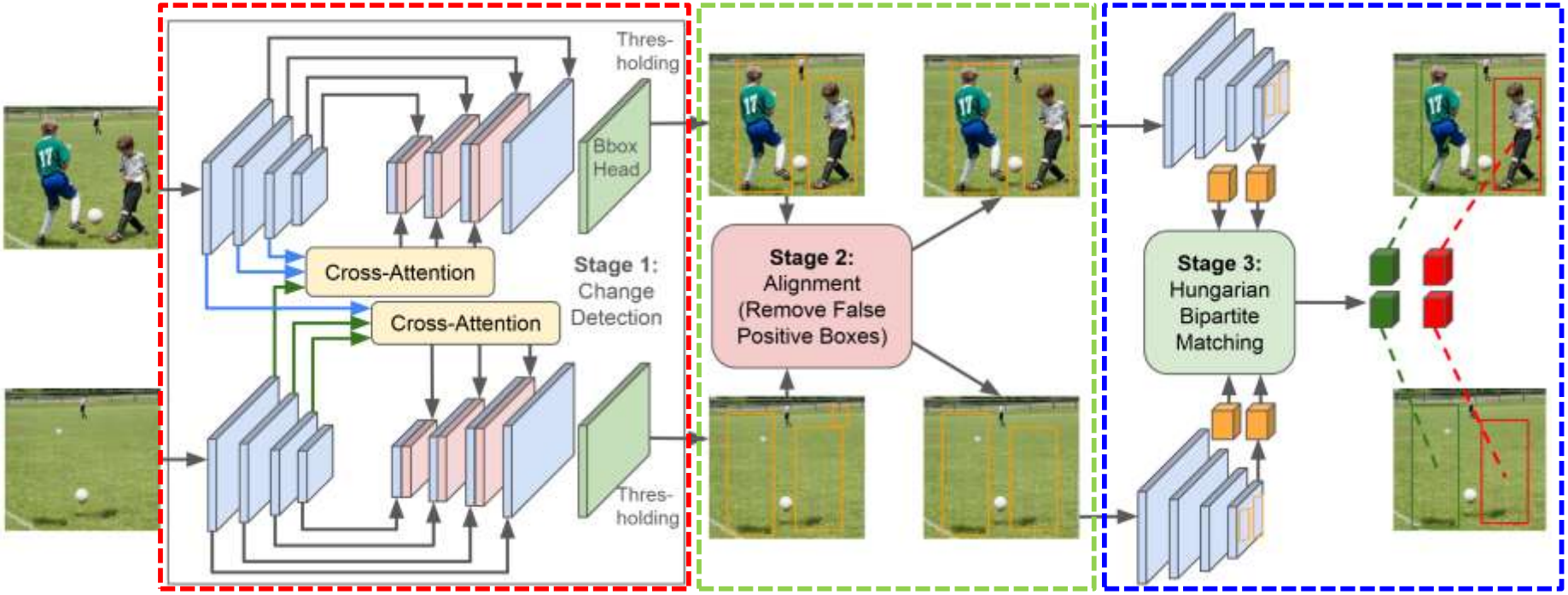
	test dataset	COCO-Inpainted small	COCO-Inpainted medium	COCO-Inpainted large	COCO-Inpainted all		VIRAT-STD	Synthtext-Change	Kubric-Change	KC-3D		RC-3D	
	depth	Const.					Const.	Const.	Const.	GT		GT	Est.
	registration	GT			GT	Est.	Id.	Id.	Est.	GT	Est.	Est.	
method	training data	(Const. = Constant, Id. = identity, Est. = Estimated, GT = Ground Truth)											
cyws	coco-inpainted	**0.46**	**0.79**	**0.85**	**0.63**		**0.65**	**0.89**	0.76	0.13		0.12	
cyws	coco-inpainted + KC-3D	0.41	0.73	0.78	0.57		0.54	0.87	0.76	**0.87**		0.14	
ours	coco-inpainted	0.34	0.69	0.76	0.52	0.51	0.46	0.85	**0.84**	0.14	0.10	0.35	0.27
ours	KC-3D	0	0.03	0.06	0.02	0.02	0.01	0.01	0.23	0.83	0.69	0.19	0.19
ours	coco-inpainted + KC-3D	0.36	0.72	0.77	0.53	0.52	0.49	0.84	**0.84**	0.82	0.68	**0.50**	0.41

Table 2. **Results:** We report the AP of cyws [26] and our model on test sets described in Sec 4.2.

Improving Zero-Shot Object-Level Change Detection by Incorporating Visual Correspondence [WACV 2025]

Method

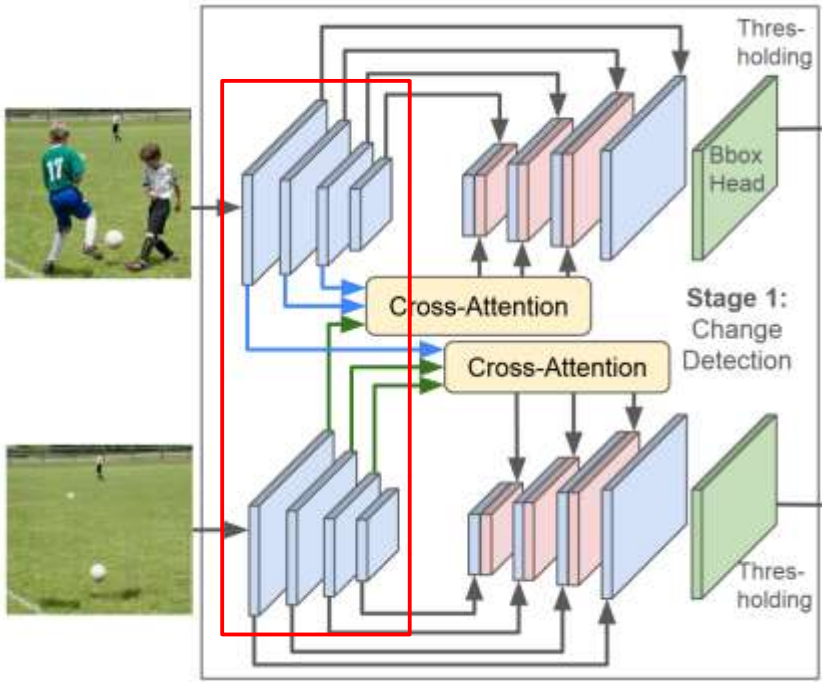
- Overview
  - 기존 2D 모델에 두 이미지 간 대응하는 바운딩 박스를 매칭하는 알고리즘을 추가함으로써 변화 영역의 위치 정합성과 탐지 정확도를 개선
    - 2D 모델에서 검출된 바운딩 박스 중 false positive를 제거한 후, 두 이미지에서 동일한 객체에 해당하는 박스를 매칭 알고리즘을 통해 대응시킴



CYWS Model | False positive 제거 | Box matching

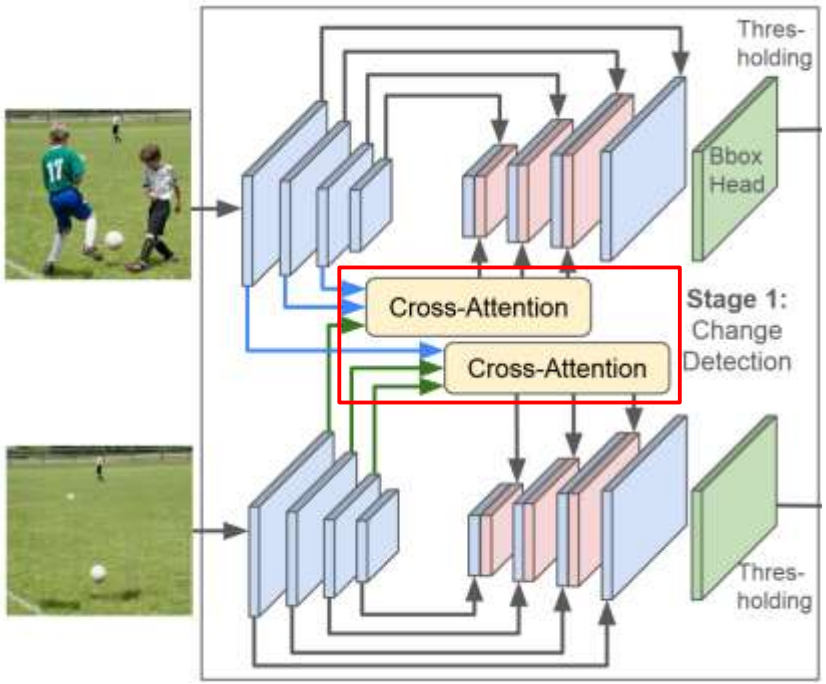
Method

- Stage 1: change detection
  - U-Net 인코더
    - Resnet-50을 인코더 백본으로 사용
    - 입력 이미지는 3 × 3 × 256, 마지막 레이어 출력은 8 × 8 × 2048 형태
    - 각 레이어에서 나온 feature를 skip connection으로 디코더에 전달



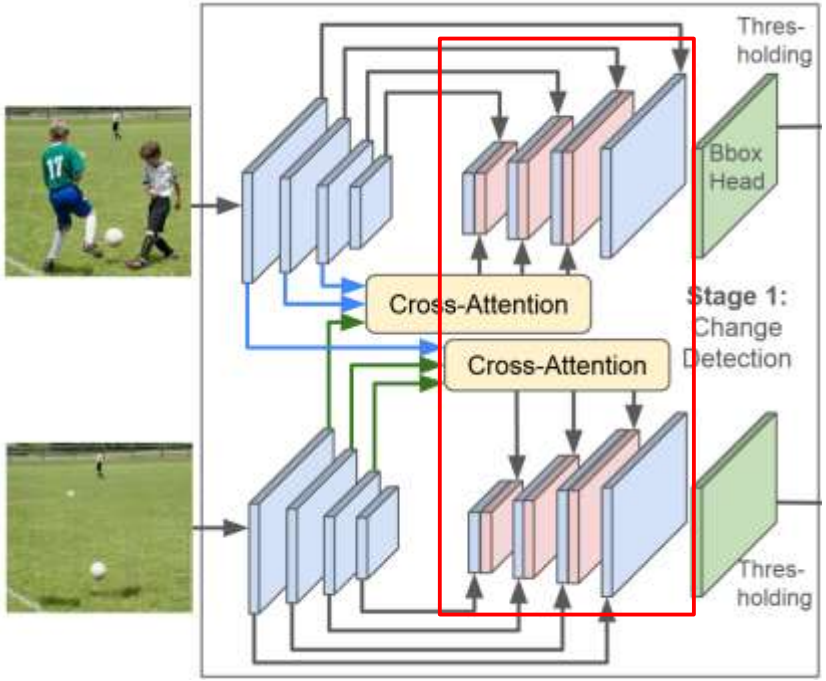
Method

- Stage 1: change detection
  - Cross attention
    - 두 이미지 간 정보 교환 가능하게 함
      - 두 이미지 사이 변화를 정확하게 계산
    - $8 \times 8 \times 4096, 16 \times 16 \times 2048, 32 \times 32 \times 1024$ 크기의 3개의 특징 맵 생성



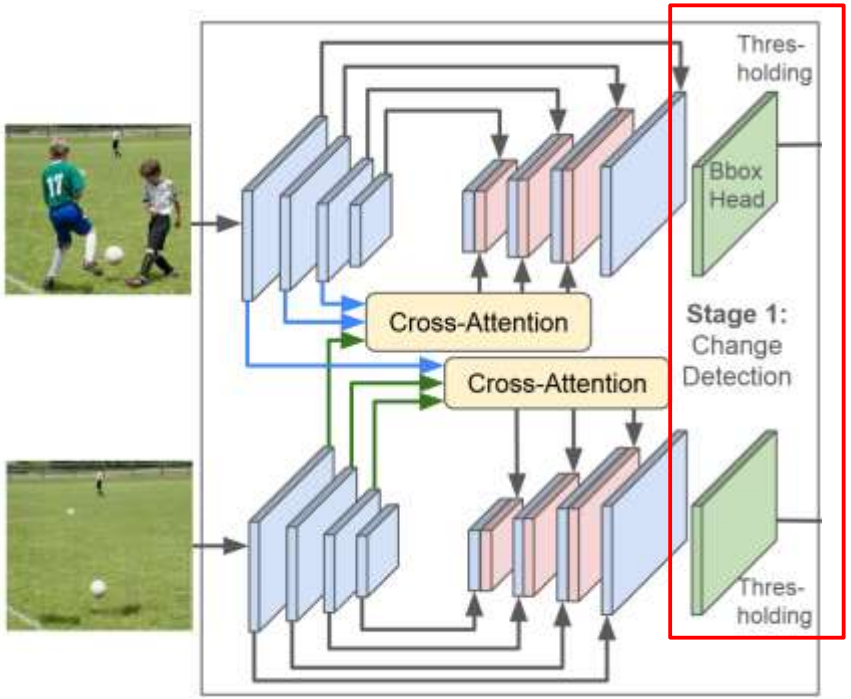
Method

- Stage 1: change detection
  - U-Net 디코더
    - 인코더의 스킵 커넥션과 크로스 어텐션 모듈로부터 생성된 세 개의 특징 맵을 입력으로 사용하여 업샘플링을 통해 64 × 256 × 256 크기의 특징 맵을 생성
      - 업샘플링 과정에서 인코더의 스킵 커넥션과 sCSE 블록이 통합됨
    - 디코더의 출력은 Bbox 헤드로 전달됨



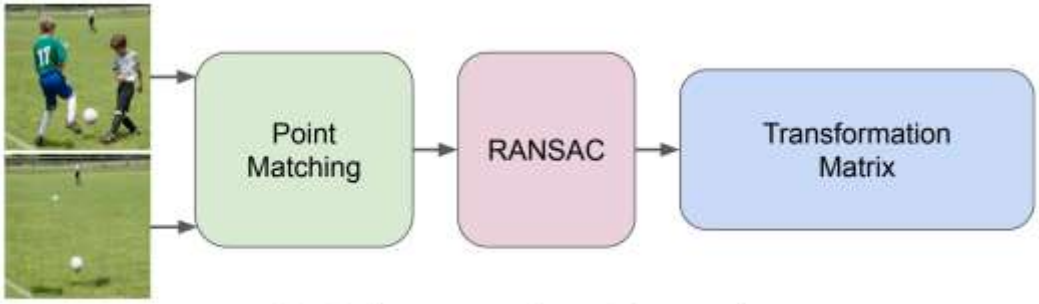
Method

- Stage 1: change detection
  - Bbox 헤드
    - CenterNet을 활용하여 두 이미지 간 변화 영역에 대한 바운딩 박스를 예측
    - CenterNet은 3개의 출력 맵을 생성함
      - 센터 맵, 오프셋 맵, 높이 - 너비 맵



Method

- Stage 2: alignment
  - 검출된 바운딩 박스 중 정합성이 떨어지는 박스를 추가적으로 제거
    - Stage 1에서 detection threshold로 걸러낸 박스에 대해 추가적으로 정합성이 떨어지는 박스를 제거
      - Point matching(SuperGlue)으로 두 이미지 간 포인트 대응점을 찾은 후 이상치 제거
        - ✓이상치를 제거(RANSAC 활용)한 후 남은 박스들에 대해 affine transform으로 정합하여 매칭



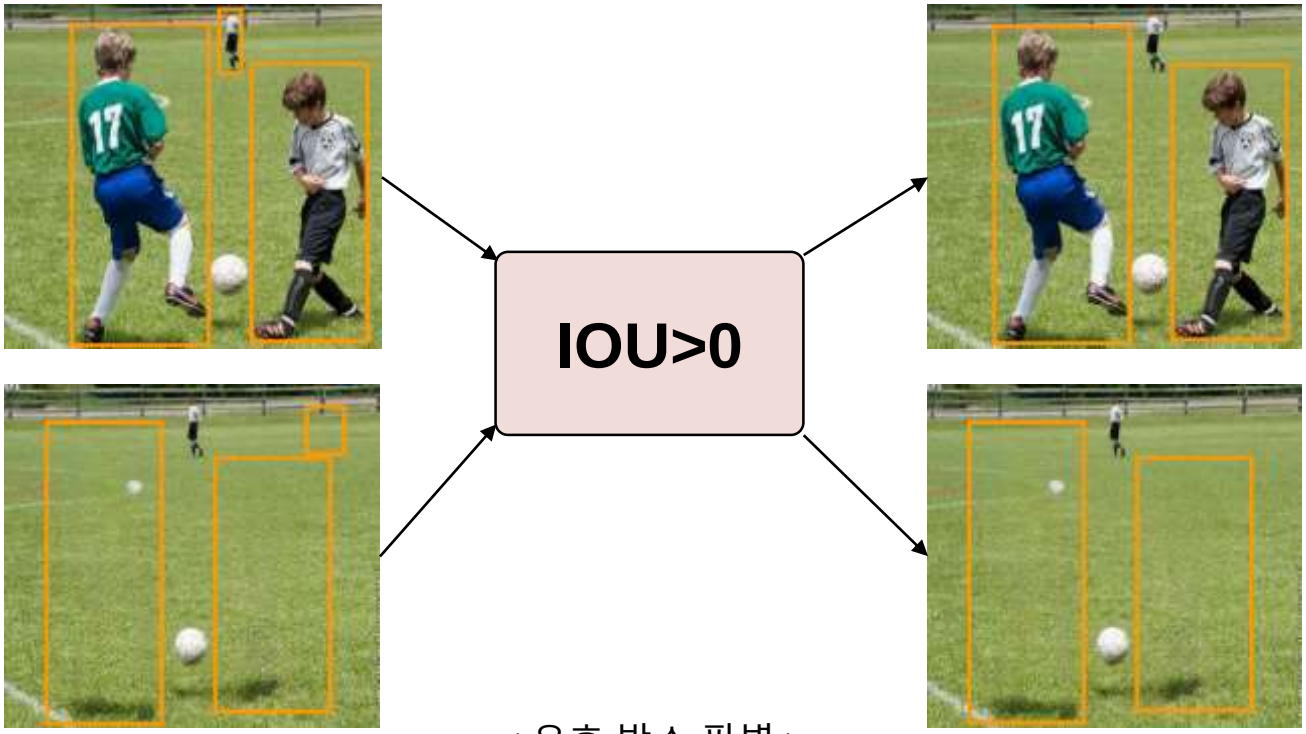
(a) Estimate transformation matrix



(b) Estimate bounding boxes locations in left image to right image

Method

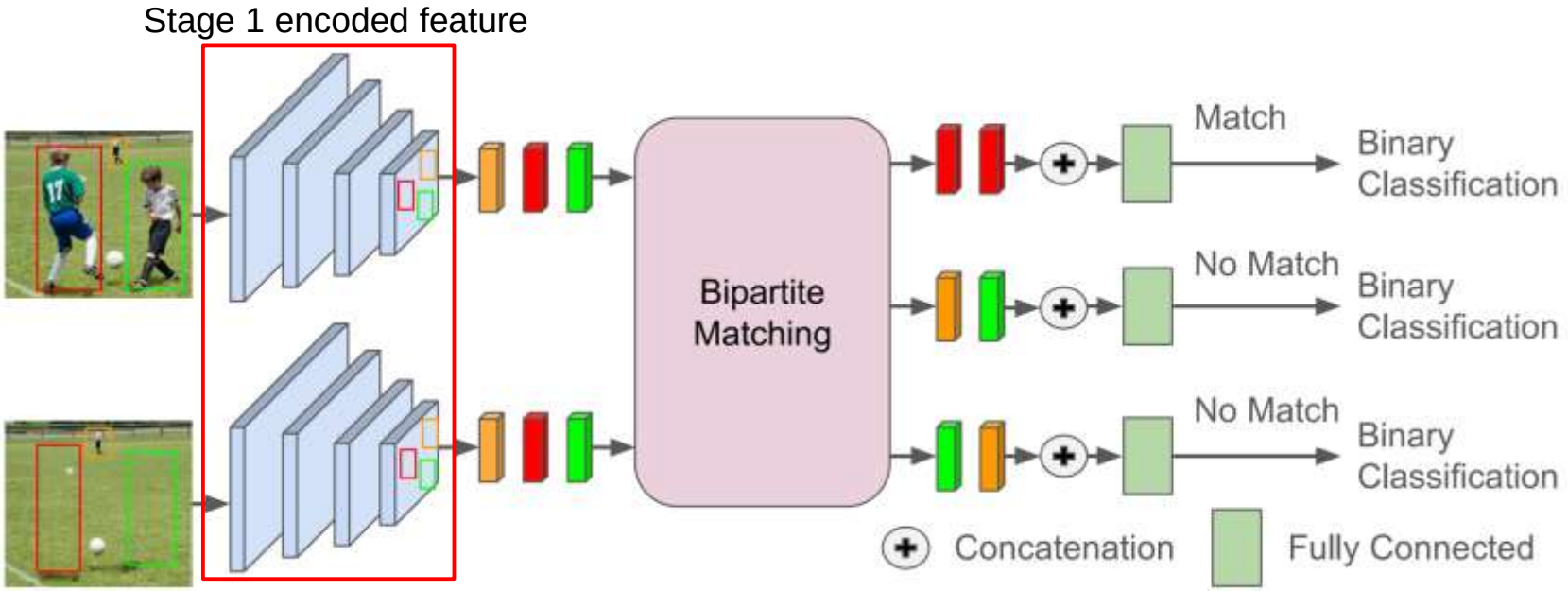
- Stage 2: alignment
  - 이후 남은 후보 박스는 정합된 위치가 다른 이미지의 박스와 겹치면(IOU>0) 유효한 것으로 판단
    - 한쪽 이미지에 변화가 예측된 박스가 있다면 다른쪽 이미지에도 이에 대응하는 박스가 있어야 한다는 전제



< 유효 박스 판별 >

Method

- Stage 3: Correspondence prediction
  - 각 이미지 쌍에 대해 예측된 바운딩 박스들 간의 대응 관계를 예측
    - Stage 2에서 정합된 각 박스에 대한 feature map을 stage 1의 인코더에서 얻음
    - 이 임베딩을 기반으로 헝가리안 이분 매칭 알고리즘과 contrastive matching loss를 함께 사용하여 최종적인 대응 관계를 예측



< feature 매칭 >

Method

- Stage 3: Correspondence prediction
  - Framework
    - Average pooling, crop 중 하나를 사용해 박스임베딩 추출
    - 추출된 박스들에 대해 두 박스 간 임베딩을 e_i, e_j라 할 때 cost matrix를 다음으로 정의

$$d_{ij} = 1 - \frac{\langle e_i \cdot e_j \rangle}{\|e_i\| \|e_j\|}$$

    - Cost matrix를 기반으로 헝가리안 알고리즘은 전체 매칭 비용이 최소가 되도록 두 이미지의 박스 간 대응 관계를 할당
    - 모델이 헝가리안 알고리즘을 통해 얻어진 매칭된 박스 쌍들을 분류할 수 있도록 학습하기 위해 contrastive matching loss를 사용

Method

- Stage 3: Correspondence prediction
  - Final loss : $L_{total} = L_{CenterNet} + \alpha L_{DETR} + \beta L_{contrastive}$

- Centernet loss 구성
  - Heatmap loss: 각 객체의 중심점이 어디인지를 학습
  - Size regression loss: 객체의 width, weight 학습
  - Offset loss: 중심점의 offset 학습

- DETR loss 구성
  - L1 loss: 예측박스 좌표와 GT 박스 좌표의 거리 차이 최소화 하는 학습
  - GIOU loss: 예측박스와 GT 박스의 GIOU 차이 최소화 하는 학습

- Contrastive matching loss
  - positive feature와 negative feature 쌍을 연결한 후 이들을 이진 분류 문제로 간주하여 BCE loss를 통해 올바른 매칭 관계를 학습하게 하는 손실 함수

Experiment

- 변화 탐지 성능
  - 모든 세 가지 후처리 단계에서 기존 2D 모델(CYWS)을 능가함
    - 이는 박스 간 대응을 학습하는데에 쓰인 contrastive matching loss에 기인함

	Model	Det-thres	Align	Hung	Thres					
	CYWS				n/a	62.73	53.98	76.13	88.97	69.70
(a)	Our				n/a	**71.77**	**57.00**	**81.07**	**90.02**	**78.84**
						+9.04	+3.02	+4.94	+1.05	+9.14
	CYWS	✓			0.25	51.25	43.40	65.46	86.38	61.68
(b)	Our	✓			0.25	**62.22**	**47.80**	**71.25**	**88.08**	**72.69**
						+10.97	+4.40	+5.79	+1.70	+11.01
	CYWS	✓	✓		0.25	47.83	40.44	64.98	84.76	44.39
(c)	Our	✓	✓		0.25	**59.23**	**44.62**	**69.23**	**86.59**	**58.83**
						+11.40	+4.18	+4.25	+1.83	+14.44
	CYWS	✓	✓	✓	0.25	44.48	39.93	54.16	83.78	44.41
(d)	Our	✓	✓	✓	0.25	**56.59**	**44.67**	**60.85**	**85.86**	**58.88**
						+12.02	+5.34	+6.69	+2.08	+14.47

< 변화 탐지 mAP 성능 지표 >

Experiment

- 매칭 성능
  - 기존 2D 모델과 최신 모델의 매칭 성능 비교
    - 실험 결과 기존 2D 모델보다 최신 모델이 매칭 성능이 더 뛰어남을 확인
      - ∴ 이는 객체 박스 쌍의 관계를 효과적으로 학습하는 contrastive matching loss의 영향

Model	Det-thres	Align	Hung	Thres					
CYWS	✓		✓	0.25	39.72	18.78	56.70	54.73	55.91
Our	✓		✓	0.25	**45.77**	**18.81**	**63.84**	**55.27**	**68.15**
					+6.05	+0.03	+7.14	+0.54	+12.24
CYWS	✓	✓		0.25	38.40	53.22	63.93	83.83	57.59
Our	✓	✓		0.25	**46.00**	**56.06**	**68.98**	**85.22**	**69.45**
					+7.60	+2.84	+5.05	+1.39	+11.86
CYWS	✓	✓	✓	0.25	41.79	53.35	63.96	84.00	57.67
Our	✓	✓	✓	0.25	**48.35**	**56.99**	**69.72**	**85.31**	**69.53**
					+6.56	+3.64	+5.76	+1.31	+11.86

< 매칭 성능에 대한 F1 score >

Experiment

- 매칭 성능
  - Stage 2 alignment의 필요성 확인
    - 실험 결과 stage 2를 적용하지 않았을때 보다 적용한 후 stage 3를 적용했을때 더 성능이 높아짐

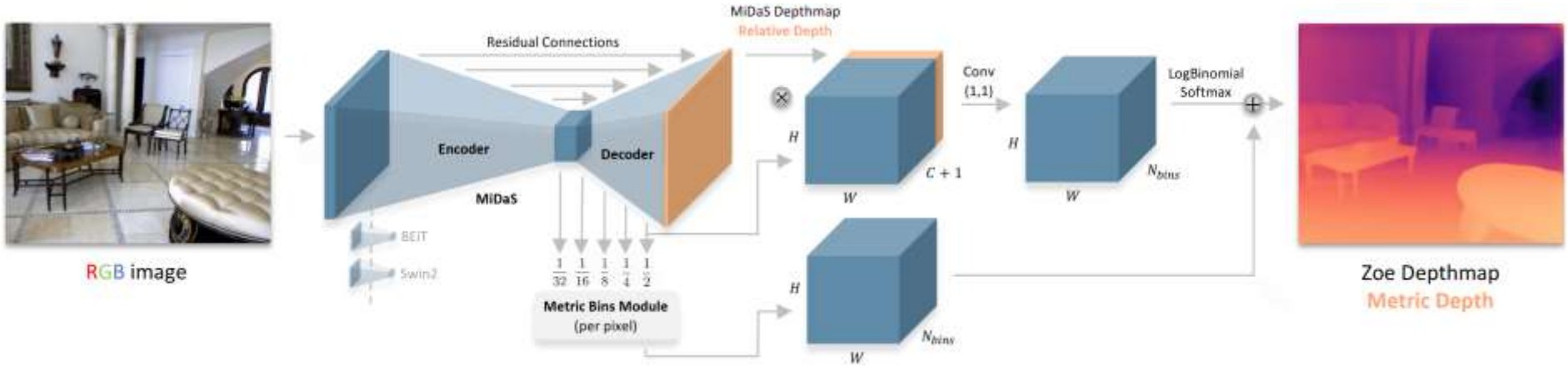
Model	Det-thres	Align	Hung	Thres					
CYWS	✓		✓	0.25	39.72	18.78	56.70	54.73	55.91
CYWS	✓	✓		0.25	38.40	53.22	63.93	83.83	57.59
CYWS	✓	✓	✓	0.25	**41.79**	**53.35** (+34.57)	**63.96**	**84.00** (+29.27)	**57.67**
Our	✓		✓	0.25	45.77	18.81	63.84	55.27	68.15
Our	✓	✓		0.25	46.00	56.06	68.98	85.22	69.45
Our	✓	✓	✓	0.25	**48.35**	**56.99** (+38.18)	**69.72**	**85.31** (+30.04)	**69.53**

< 매칭 성능에 대한 F1 score >

Supplement

Supplement

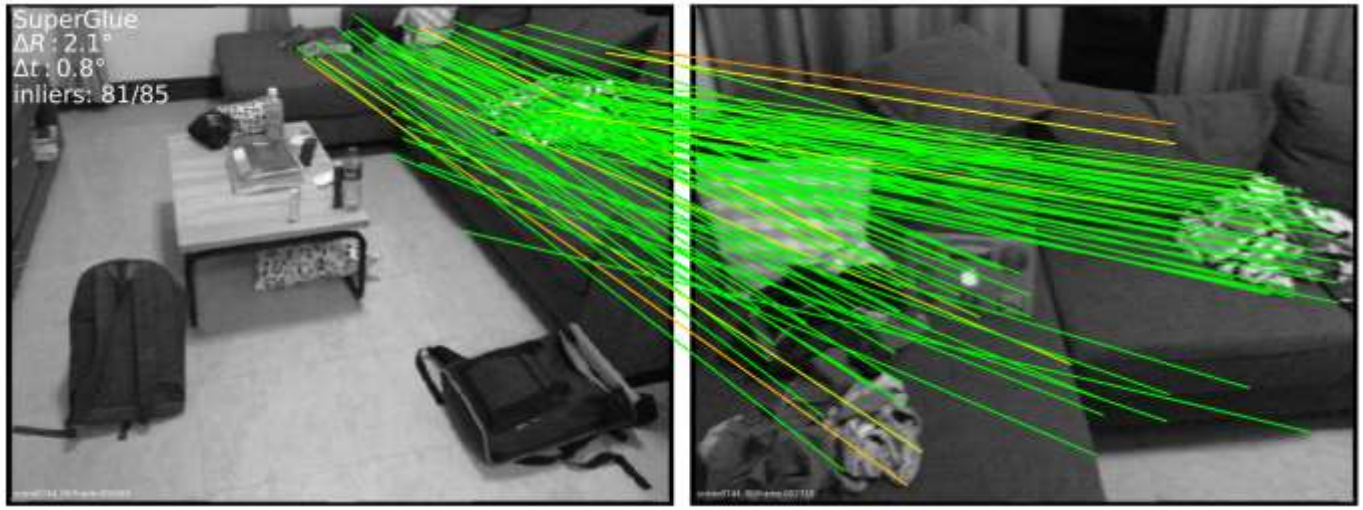
- ZoeDepth
  - Method
    - 상대적인 depth 맵과 절대적인 depth 맵을 함께 학습하여 depth 맵을 추정하는 모델
      - 상대 depth 맵 추정은 MiDas(Mixed Datasets for Monocular Depth Estimation) 기법을 사용
        - ✓스케일이 다른 여러 데이터셋으로 파인튜닝함으로써 상대적 depth맵 추정의 일반화 성능 향상
      - 절대 depth 맵 추정은 Metric Bins Module 사용
        - ✓깊이 추정을 각 bin(깊이 구간)에 대한 가중치 확률의 합으로 결정



< ZoeDepth architecture >

Supplement

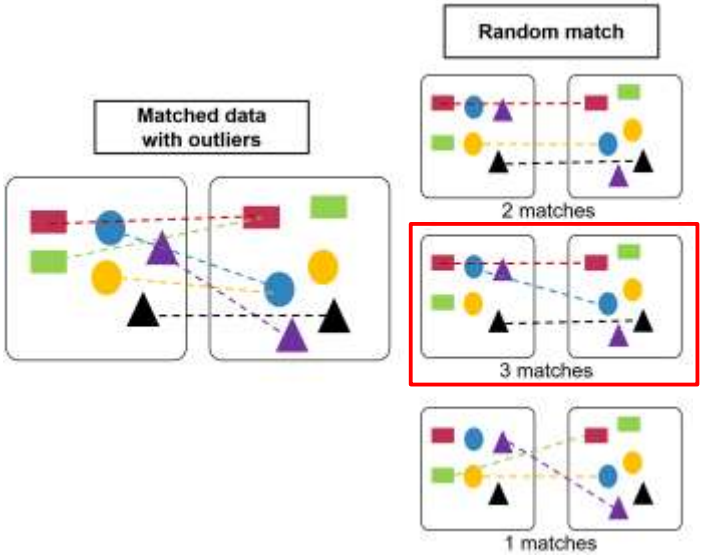
- SuperGlue
  - Viewpoint 변화가 큰 두 이미지에 대해서도 대응점 매칭 성능이 뛰어난 모델
    - 시점이 크게 달라진 두 이미지의 대응점을 찾아내는데에 활용하기 좋은 모델



< SuperGlue 사용 예시 >

Supplement

- RANSAC
  - 검출된 대응점 중 정합성이 떨어지는 대응점을 추가적으로 제거
    - 이상치 제거 시 RANSAC 사용
      - 반복적으로 소수의 매칭 점들을 무작위로 선택
      - 선택된 점들의 affine transform 계산
      - 그 변환이 많은 점들과 일치할 경우(inlier 수가 많다는 뜻) 해당 변환을 최종 정렬 변환(affine transform)으로 선택
      - 나머지 일치하지 않는 점들(outlier)은 제거



→ 최종 모델로 선택

< RANSAC 원리 모식도 >

Thank you