

Habitat: 구현된 AI 연구를 위한 플랫폼

Manolis Savva^{1,4*}, Abhishek Kadian^{1*}, Oleksandr Maksymets^{1*}, Yili Zhao¹,
에릭 위즈만스^{1,2,3}, 바바나 자인¹, 줄리안 스트라우브², 리우 지아¹, 블라들렌 콜튼⁵,
지텐드라 말리크^{1,6}, 데비 파리크^{1,3}, 드루브 바트라^{1,3}

¹Facebook AI Research, ²Facebook Reality Labs, ³Georgia Institute of Technology, ⁴Simon Fraser University, ⁵<https://aihabitat.org>, ⁶UC 버클리

추상적인

구현된 인공지능(AI) 연구 플랫폼인 해비타트(Habitat)를 소개합니다. Habitat은 매우 효율적인 사실적인 3D 시뮬레이션에서 구현된 에이전트(가상 로봇)를 교육할 수 있습니다. 특히 Habitat은 다음으로 구성됩니다. (i) Habitat-Sim: 구성 가능한 에이전트, 센서 및 일반 3D 데이터 세트 처리 기능을 갖춘 유연한 고성능 3D 시뮬레이터입니다. Habitat-Sim은 빠릅니다. Matterport3D에서 장면을 렌더링할 때 단일 스레드로 실행되는 초당 수천 프레임(fps)을 달성하고 단일 GPU에서 10,000fps 이상의 다중 프로세스에 도달할 수 있습니다. (ii) Habitat-API: 구현된 AI 알고리즘의 종단 간 개발을 위한 모듈식 고수준 라이브러리 - 작업 정의 (예: 탐색, 지침 따르기, 질문 응답), 구현된 에이전트 구성, 교육 및 벤치마킹.

이러한 대규모 엔지니어링 기여를 통해 우리는 지금까지 실행 불가능하거나 '단지' 실행 불가능했던 실험을 요구하는 과학적 질문에 답할 수 있습니다. 구체적으로, 포인트 목표 탐색의 맥락에서: (1) 우리는 두 개의 최근 작업 [20, 16]에서 학습과 SLAM 접근 방식 간의 비교를 다시 검토하고 반대 결론에 대한 증거를 찾습니다. 이전 조사보다 훨씬 더 많은 경험, 그리고 (2) 우리는 여러 센서 {blind, RGB, RGBD, D}에 대해 최초의 교차 데이터 세트 일반화 실험 {train, test} × {Matterport3D, Gibson}을 수행하고 깊이(D) 센서는 데이터 세트 전체에서 일반화됩니다. 우리는 우리의 오픈 소스 플랫폼과 이러한 발견이 구현된 AI에 대한 연구를 발전시키기를 바랍니다.

1. 소개

체현 가설은 지능이 에이전트와 환경의 상호 작용에서 그리고 감각 운동 활동의 결과로 나타난다는 생각입니다.

스미스와 개서 [26]

가정용 로봇에게 다가가 '이봐 - 내 노트북이 내 책상 위에 있는지 확인해줄 수 있어? 그렇다면 가져 오십시오.' 이러한 로봇이 성공하려면 시각적 인식(장면과 물체 인식), 언어 이해(질문과 지침을 행동으로 변환), 복잡한 환경 내 탐색 (물건을 이동하고 찾는 데) 등 다양한 기술이 필요합니다. 변화하는 환경에서).

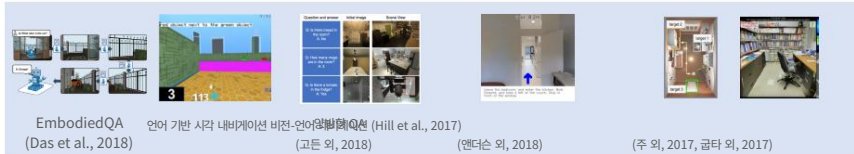
심층 표현의 최근 발전 덕분에 비전 및 언어 커뮤니티에서 상당한 진전이 있었지만 [14, 11], 이러한 진전의 대부분은 구현된 AI가 아닌 '인터넷 AI'에 있었습니다. 전자의 초점은 일반적으로 인터넷에서 선별된 데이터 세트의 이미지, 비디오 및 텍스트의 패턴 인식입니다 [10, 18, 4]. 후자의 초점은 환경에서 구현된 에이전트(예: 로봇)의 작업을 가능하게 하는 것입니다. 이것은 능동적인 인식, 장기 계획, 상호 작용을 통한 학습, 환경에 기초한 대화 유지를 전면에 내세웁니다.

간단한 제안은 실제 세계에서 에이전트를 직접 교육하여 모든 풍부함을 노출시키는 것입니다. 이는 가치가 있으며 앞으로 AI 개발에 중요한 역할을 할 것입니다. 그러나 실제 세계에서 훈련하는 로봇은 느리고 (실제 세계는 실시간보다 빠르지 않으며 병렬화할 수 없음) 위험하고 (잘 훈련되지 않은 에이전트는 자신도 모르게 자신, 환경 또는 다른 사람을 해칠 수 있음) 집약적(로봇과 로봇이 실행하는 환경은 리소스와 시간을 요구함), 제어하기 어려움(정의상 드물고 재현하기 어려운 코너 케이스 시나리오를 테스트하기 어렵습니다), 쉽게 재현할 수 없습니다 (실험 및 기간 전반에 걸쳐 조건을 복제하는 것은 어렵습니다).

*균등 기여를 나타냅니다.

우리는 보완적인 연구 프로그램을 지원하는 것을 목표로 합니다.

작업



시뮬레이터



데이터셋



해비타트 플랫폼



그림 1: 구현된 에이전트를 교육하기 위한 '소프트웨어 스택'에는 (1) 시맨틱 주석이 있는 3D 자산을 제공하는 데이터 세트, (2) 이러한 자산을 렌더링하고 구현된 에이전트를 시뮬레이션할 수 있는 시뮬레이터 및 (3) 평가 가능한 작업을 정의하는 작업이 포함됩니다. 과학적 진보를 벤치마킹할 수 있게 해주는 문제. 이전 작업(파란색 상자로 강조 표시됨)은 다양한 데이터 세트, 시뮬레이션 소프트웨어 및 작업 정의에 기여했습니다. 우리는 일반 데이터 세트 지원, 고성능 시뮬레이터 (Habitat-Sim) 및 광범위한 작업 세트의 정의 및 평가를 허용하는 유연한 API (Habitat-API)를 포함하여 Habitat 플랫폼과 통합된 구현된 에이전트 스택을 제안합니다.

풍부하고 사실적인 시뮬레이터에서 구체화된 에이전트(예: 가상 로봇)를 교육한 다음 학습된 기술을 현실로 전송합니다.

시뮬레이션은 과학 및 공학(항공우주에서 동물학에 이르기까지)에서 길고 풍부한 역사를 가지고 있습니다. 구현된 AI의 맥락에서 시뮬레이터는 앞서 언급한 문제를 극복하는 데 도움이 됩니다. 시뮬레이터는 실시간보다 훨씬 빠르게 실행할 수 있고 클러스터를 통해 병렬화할 수 있습니다. 시뮬레이션 교육은 안전하고 저렴하며 커뮤니티 전체의 공동 노력에서 진행 상황을 공정하게 비교하고 벤치마킹할 수 있습니다. 유망한 접근 방식이 개발되고 시뮬레이션에서 테스트되면 실제 세계에서 작동하는 물리적 플랫폼으로 이전될 수 있습니다 [6, 15].

데이터 세트는 컴퓨터 비전, NLP 및 기타 AI 영역에서 발전의 핵심 동인이었습니다 [10, 18, 4, 1]. 커뮤니티가 구현된 AI로 전환함에 따라 시뮬레이터가 이전에 데이터 세트가 수행한 역할을 맡게 될 것이라고 믿습니다.

이러한 전환을 지원하기 위해 우리는 구현된 에이전트를 교육하기 위한 전체 '소프트웨어 스택'을 표준화하는 것을 목표로 합니다(그림 1): 세계 스캔 및 사실적인 3D 자산 생성, 고효율 및 병렬화 가능한 차세대 시뮬레이터 개발, 구현된 AI 작업 지정 이를 통해 과학적 진보를 벤치마킹하고 구현된 에이전트를 교육하고 배포하기 위한 모듈식 고급 라이브러리를 출시할 수 있습니다. 구체적으로 해비타트는 다음과 같이 구성됩니다.

1. Habitat-Sim: 구성 가능한 에이전트, 여러 센서 및 일반 3D 데이터 세트 처리 기능이 있는 유연한 고성능 3D 시뮬레이터 (Matterport3D, Gibson 및 Replica 데이터 세트에 대한 기본 제공 지원 포함).

2. Habitat-API: 구현된 AI 알고리즘의 엔드 투 엔드 개발을 위한 모듈식 고수준 라이브러리 - 구현된 AI 작업 정의(예: 탐색, 지침 따르기,

질문 응답), 구현된 에이전트 구성 및 교육(모방 또는 강화 학습 또는 클래식 SLAM을 통해), 표준 메트릭을 사용한 벤치마킹 [2].

Habitat 아키텍처 및 구현은 모듈성과 고성능을 결합합니다.

Matterport3D 데이터 세트에서 장면을 렌더링할 때 Habitat-Sim은 단일 스레드를 실행하는 초당 수천 프레임(fps)을 달성하고 단일 GPU에서 10,000fps 이상의 다중 프로세스에 도달할 수 있으며 이는 가장 가까운 시뮬레이터보다 훨씬 빠른 속도입니다. Habitat-API를 사용하면 다양한 클래스의 방법과 다양한 3D 장면 데이터 세트에 구체화된 에이전트를 교육하고 벤치마킹할 수 있습니다.

이러한 대규모 엔지니어링 기여를 통해 우리는 지금까지 실행 불가능하거나 '단기' 실행 불가능했던 실험을 요구하는 과학적 질문에 답할 수 있습니다. 구체적으로, 포인트 클라우드 내비게이션 [2]의 맥락에서 우리는 두 가지 과학적 기여를 합니다.

1. 우리는 두 개의 최근 연구 [20, 16]에서 학습과 SLAM 접근법 사이의 비교를 재검토하고 반대 결론에 대한 증거를 찾습니다. 즉, 학습이 이전 조사보다 훨씬 더 많은 경험으로 확장될 경우 SLAM을 수행한다는 것입니다.

2. 우리는 여러 센서 {Blind1, RGB, RGBD, D} × {GPS+Compass}에 대해 첫 번째 교차 데이터 세트 일반화 실험 {train, test} × {Matterport3D, Gibson}을 수행하고 깊이 (D) 센서는 데이터 세트 전체에서 잘 일반화됩니다.

우리는 우리의 오픈 소스 플랫폼과 이러한 발견이 구현된 AI의 미래 연구를 발전시키고 안내할 것입니다.

1Blind는 시각적 감각 입력이 없는 에이전트를 말합니다.

2. 관련 업무

현실은 당신이 위로 올라가는 것입니다.

라이자 마넬리

대규모 3D 장면 데이터 세트 [5, 27, 8]의 가용성 과 활성 비전 작업에 대한 커뮤니티의 관심으로 인해 최근 작업이 급증하여 실내 환경을 위한 다양한 시뮬레이션 플랫폼이 개발되었습니다 [17, 7, 13, 24, 29, 3, 30, 31, 23]. 이러한 플랫폼은 사용하는 3D 장면 데이터, 처리하는 구현된 에이전트 작업 및 구현하는 평가 프로토콜에 따라 다릅니다.

이러한 활동의 급증은 짜릿하면서도 놀라운 일입니다. 한편으로는 다양한 연구 커뮤니티(컴퓨터 비전, 자연어 처리, 로봇 공학, 머신 러닝)에서 구현된 AI에 대한 관심의 신호임이 분명합니다. 다른 한편으로, 서로 다른 여러 시뮬레이션 환경의 존재는 단편화, 노력의 복제, 재생산 및 커뮤니티 전체 진행의 어려움을 유발할 수 있습니다.

또한 기존 시뮬레이터는 다음과 같은 몇 가지 단점을 나타냅니다. - 작업 (예: 탐색), 시뮬레이션 플랫폼 (예: GibsonEnv) 및 3D 데이터 세트(예: Gibson)의 긴밀한 결합. 여러 작업 또는 데이터 세트를 사용한 실험은 비실용적입니다.

- 하드 코딩된 에이전트 구성(예: 크기, 작업 공간). 에이전트 매개변수 및 센서 유형의 제거는 지원되지 않으므로 결과를 비교하기 어렵습니다.

- 차선의 렌더링 및 시뮬레이션 성능. 대부분의 기존 실내 시뮬레이터는 상대적으로 낮은 프레임 속도(10~100fps)로 작동하여 에이전트 훈련에 병목 현상이 발생하고 대규모 학습이 불가능합니다. 이러한 실험에서 메시지를 제거하면 신뢰할 수 없게 됩니다. 비교를 신뢰하도록 학습이 수렴되었습니까?

- 환경 상태의 제한된 제어. 현재 개체 측면에서 3D 장면의 구조는 프로그래밍 방식으로 수정할 수 없습니다(예: 에이전트의 견고성 테스트).

가장 중요한 것은 기존 플랫폼 위에 구축된 작업은 플랫폼과 독립적으로 재생산하기 어렵기 때문에 대상 작업과 데이터 세트가 동일한 경우에도 다른 플랫폼 기반 작업과 비교하여 평가하기 어렵다는 것입니다. 이러한 현상 유지는 바람직하지 않으며 Habitat 노력에 동기를 부여합니다. 우리는 이전 프레임워크의 성공에서 배우고 한계를 해결하면서 바람직한 특성을 결합하는 통합 플랫폼을 개발하는 것을 목표로 합니다. 공통의 통합 플랫폼은 코드 재사용과 일관된 실험 방법론을 가능하게 하여 연구를 크게 가속화할 수 있습니다. 또한 공통 플랫폼을 통해 서로 다른 패러다임(학습된 vs. 고전적) 및 데이터 세트 간의 에이전트 일반화를 기반으로 에이전트를 테스트하는 실험을 쉽게 수행할 수 있습니다.

내비게이션에 대해 학습된 접근 방식과 고전적인 접근 방식을 대조하여 수행하는 실험은 Mishkin et al.의 최근 작업과 유사합니다. [20]. 그러나 [20]에서 사용된 MINOS [24]에 대한 Habitat 스택의 성능

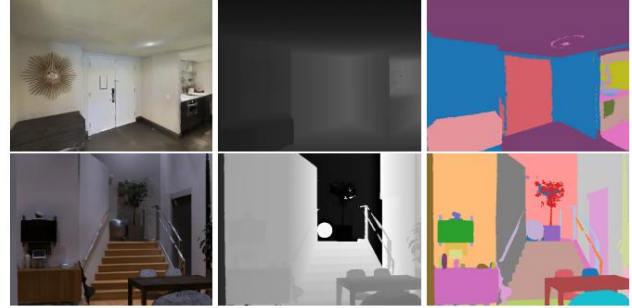


그림 2: 두 개의 서로 다른 환경 데이터 세트에서 세 개의 센서 (컬러 카메라, 깊이 센서, 시맨틱 인스턴스 마스크)에 대해 렌더링된 센서 관찰의 예. Matterport3D [8] 환경은 맨 위 행에 있고 Replica [28] 환경은 맨 아래 행에 있습니다.

- 초당 수천 대 백 프레임 - 상당히 많은 양의 경험(7,500만 단계 대 5 백만 단계)으로 훈련된 에이전트를 평가할 수 있습니다. 우리가 관찰한 경향은 학습된 상담원이 많은 양의 교육 경험을 제공받을 때 기존 접근 방식과 일치하고 이를 능가할 수 있음을 보여줍니다. Kojima와 Deng [16]의 다른 최근 연구에서도 학습된 에이전트와 수동 엔지니어링 내비게이션 에이전트를 비교했지만 에이전트의 성능을 특성화하고 내비게이션 에피소드에 대한 경도 측정을 설정하기 위해 추가 메트릭을 정의하는 데 중점을 둡니다. 우리의 지식에 있어 우리의 실험은 현실적인 실내 환경에서 수개월의 경험을 제공받은 내비게이션 에이전트를 훈련하고 이를 고전적인 방법과 대조하는 최초의 실험입니다.

3. 해비타트 플랫폼

내가 창조할 수 없는 것은 이해할 수 없다.

리차드 파인만

Habitat의 개발은 구현된 에이전트에 대한 연구를 위한 공통 작업 프레임워크 [12]의 형성을 가능하게 하여 이 분야의 체계적인 연구 진행을 지원하기 위한 장기적인 노력입니다.

설계 요구 사항. 이전 섹션에서 논의된 문제는 우리가 충족하고자 하는 일련의 요구 사항으로 이어집니다.

- 고성능 렌더링 엔진: 동시에 작동하는 여러 에이전트에 대해 시각적 정보의 여러 채널(예: RGB, 깊이, 시맨틱 인스턴스 분할, 표면 법선, 광학 흐름)을 생성할 수 있는 리소스 효율적인 렌더링 엔진입니다.

- 장면 데이터 세트 수집 API: 플랫폼이 3D 장면 데이터 세트와 무관하게 만들고 사용자가 자신의 데이터 세트를 사용할 수 있도록 합니다.

- 에이전트 API: 사용자가 잘 정의된 형상, 물리학 및 작동 특성을 가진 매개변수화된 구현 에이전트를 지정할 수 있습니다.

- 센서 스위트 API: 각 에이전트에 부착된 매개변수화된 센서(예: RGB, 깊이, 접촉, GPS, 나침반 센서)의 임의 개수 지정을 허용합니다.
- 시나리오 및 작업 API: 작업 및 해당 평가 프로토콜의 이식 가능한 정의를 허용합니다.
- 구현: Python API를 사용하는 C++ 백엔드 및 공통 학습 프레임워크와의 상호 운용은 진입 임계값을 최소화합니다.
- 컨테이너화: 클러스터의 분산 교육 및 사용자 제공 코드의 원격 서버 평가를 가능하게 합니다.
- 에이전트로서의 인간(Humans-as-agents): 시뮬레이션에서 인간이 에이전트 역할을 하여 인간 행동을 수집하고 인간-에이전트 또는 인간-인간 상호 작용을 조사할 수 있습니다.
- 환경 상태 조작: 존재하는 개체 및 관련 레이아웃 측면에서 환경 구성을 프로그래밍 방식으로 제어합니다.

디자인 개요. 위의 설계 요구 사항은 그림 1의 '소프트웨어 스택'에서 여러 계층에 걸쳐 있습니다. 모놀리식 설계는 모든 수준에서 요구 사항을 처리하는 데 적합하지 않습니다. 따라서 우리는 이 다층 추상화를 반영하도록 해비타트 플랫폼을 구성합니다.

최하위 레벨에는 유연한 고성능 3D 시뮬레이터인 Habitat-Sim이 있습니다. 이 시뮬레이터는 3D 장면을 표준화된 장면 그래프 표현으로 로드하고, 에이전트를 여러 센서로 구성하고, 에이전트 모션을 시뮬레이션하고, 에이전트의 센서 제품군에서 감각 데이터를 반환합니다. Habitat의 센서 추상화를 통해 LIDAR 및 IMU와 같은 추가 센서를 플러그인으로 쉽게 구현할 수 있습니다.

장면 그래프를 사용하는 일반 3D 데이터 세트 API. Habitat-Sim은 계층적 장면 그래프를 사용하여 합성이나 실제 재구성을 기반으로 하든 지원되는 모든 3D 환경 데이터 세트를 나타냅니다. 균일한 장면 그래프 표현을 사용하면 특정 데이터 세트의 세부 정보를 추상화하고 일관된 방식으로 처리할 수 있습니다. 장면 그래프를 사용하면 절차적 장면 생성, 편집 또는 프로그래밍 방식 조작을 통해 3D 환경을 구성할 수 있습니다.

렌더링 엔진. Habitat-Sim 백엔드 모듈은 C++로 구현되며 Magnum 그래픽 미들웨어 라이브러리²를 활용하여 다양한 하드웨어 구성에서 플랫폼 간 배포를 지원합니다. 시뮬레이터 백엔드는 컬러 카메라 센서, 깊이 센서 및 시멘틱 마스크 센서에 대한 출력을 결합하는 다중 부착 'uber-shader'를 사용하여 시각적 센서 프레임 렌더링을 구현하는 효율적인 렌더링 파이프라인을 사용합니다. 모든 출력이 단일 렌더 패스에서 생성되도록 허용함으로써 센서 매개변수가 공유되고 동일한 렌더 패스가 모든 출력에 사용될 수 있을 때 추가 오버헤드를 방지합니다.

그림 2는 지원되는 세 가지 데이터 세트에서 렌더링된 시각 센서의 예를 보여줍니다. 동일한 에이전트 및 센서 구성이 각각의 장면에서 인스턴스화되었습니다.

²<https://magnum.graphics/>

센서 / 해상도	1 공정		5개 프로세스		
	128	256 512	128	256	512
RGB	4,093	1,987 848	10,592	3,574	2,629
RGB + 깊이	423	5,223 1,774	1,348		

표 1: Intel Xeon E5-2690 v4 CPU 및 Nvidia Titan Xp GPU에서 예시 Matterport3D 장면(id 17DRP5sb8fy)에 대한 초당 프레임 수의 Habitat-Sim 성능, 다양한 프레임 해상도 및 다양한 수의 동시 시뮬레이터로 측정 GPU를 공유하는 프로세스 추가 벤치마킹 결과는 부록을 참조하십시오.

단순히 다른 입력 장면을 지정하여 세 개의 데이터 세트. 성능. Habitat-Sim은 시뮬레이터 스레드당 초당 수천 프레임을 달성하며 현실적인 실내 환경(일반적으로 초당 수십 또는 수백 프레임에서 작동)을 위한 이전 시뮬레이터보다 훨씬 빠릅니다. 요약 및 보충 자료는 표 1을 참조하십시오. 상세 사항은, 이에 비해 AI2-THOR [17] 및 CHALET [31]은 수십 fps로 실행되고 MINOS [24] 및 Gib son [30]은 약 100fps로 실행되며 House3D [29]는 약 300fps로 실행됩니다. Habitat-Sim은 2-3 배 더 빠릅니다. 초당 10,000 프레임으로 작동함으로써 병목 현상을 시뮬레이션에서 네트워크 교육을 위한 최적화로 전환합니다. TensorFlow 벤치마크를 기반으로 많은 인기 있는 네트워크 아키텍처가 10-100x의 프레임 속도로 실행됩니다.

단일 GPU3에서 낮습니다. 실제로 우리는 디스크에서 이미지를 로드하는 것보다 Habitat-Sim을 사용하여 이미지를 생성하는 것이 종종 더 빠르다는 것을 관찰했습니다.

효율적인 GPU 처리량. 현재 Habitat-Sim이 렌더링한 프레임은 공유 메모리를 통해 Python 텐서로 노출됩니다. 향후 개발에서는 CUDA-GL 상호 운용 및 렌더 버퍼 및 텍스처를 텐서로 직접 공유함으로써 GPU에서 CPU로의 메모리 복사 오버헤드를 완전히 피함으로써 훨씬 더 높은 렌더링 효율성에 초점을 맞출 것입니다. 우리의 예비 내부 테스트는 이것이 2배의 속도 향상으로 이어질 수 있음을 시사합니다.

시뮬레이션 백엔드 위의 Habitat-API 계층은 구현된 AI의 종단 간 개발을 위한 모듈식 고수준 라이브러리입니다. 구현된 작업 설정에는 에이전트가 사용할 수 있는 관찰을 지정하고, 시뮬레이터에서 제공하는 환경 정보를 사용하고, 정보를 작업별 에피소드 데이터 세트와 연결하는 작업이 포함됩니다. 이 클래스는 시뮬레이터의 Observations 클래스와 행동 공간을 특정 태스크로 확장합니다. 에피소드 종료 기준 및 성공 측정은 작업에서 제외됩니다. 예를 들어 목표 기반 탐색에서 작업은 목표 및 평가 메트릭을 제공합니다 [2]. 이러한 종류의 기능을 지원하기 위해 태스크는 다음에 대한 읽기 전용 액세스 권한을 가집니다.

³<https://www.tensorflow.org/guide/performance/benchmark>

시뮬레이터 및 에피소드 데이터 세트.

- 에피소드: 에이전트 의 초기 위치 및 방향 , 장면 ID, 목표 위치 및 선택적으로 목표까지의 최단 경로를 포함하는 에피소드 사양에 대한 클래스입니다 .
에피소드는 작업 인스턴스에 대한 설명입니다.
- 환경: Habitat의 기본 환경 개념으로, 시뮬레이터로 구체화된 작업을 수행하는 데 필요한 모든 정보를 추상화합니다.

Habitat 플랫폼의 아키텍처, 성능 측정 및 API 사용 예에 대한 자세한 내용은 부록에서 제공됩니다.

4. 대규모 PointGoal 탐색

Habitat 플랫폼 설계의 효용성을 입증하기 위해 서로 다른 환경 의 데이터셋 간에 목표지향적 시각적 내비게이션 에이전트 의 일반화를 테스트하고 학습 기반 에이전트와 클래식 에이전트 의 성능을 사용 가능한 훈련량 으로 비교하는 실험을 수행합니다. 경험치가 증가합니다.

작업 정의. 우리는 PointGoal 작업(Anderson et al. [2] 이 정의한 대로)을 실험 테스트베드로 사용합니다. 이 작업은 표면적으로 정의하기 간단합니다. 에이전트는 환경의 임의의 시작 위치 및 방향 에서 초기화되고 에이전트의 위치에 상대적으로 제공된 대상 좌표로 이동하도록 요청됩니다 . 실측 지도를 사용할 수 없으며 에이전트는 탐색을 위해 감각 입력만 사용해야 합니다.

그러나 실험 과정에서 우리는 이 작업이 (a) 실험 결과에 상당한 차이를 만들 수 있고 (b) 논문 간에 지정되지 않았거나 일관성이 없어 비교하기 어려운 미묘한 선택을 위한 공간을 남겨둔다는 것을 깨달았습니다. 우리는 이러한 낮은 수준의 선택에 대해 가능한 한 설명하려고 노력합니다 . Habitat 플랫폼이 이러한 불일치를 해결하는 데 도움이 되기를 바랍니다 .

에이전트 구현 및 행동 공간. 에이전트는 직경 0.2m , 높이 1.5m 의 원통형 원시 형태로 물리적으로 구현됩니다 . 행동 공간은 네 가지 행동으로 구성됩니다: turn_left, turn_right, move_forward, stop . 이러한 동작은 선회 동작에 대해 10 도 회전하고 move_forward 동작에 대해 0.25m 의 선형 변위를 초래하는 이상적인 작동에 매핑됩니다 . 중지 작업을 통해 에이전트는 목표에 도달했음을 알릴 수 있습니다 . Habitat은 시끄러운 작동을 지원하지만 이 백서의 실험은 우리의 분석이 다른 요인에 초점을 맞추기 때문에 소음이 없는 환경에서 수행됩니다 .

충돌 역학. 일부 이전 작업 [3] 은 에이전트가 한 위치에서 다른 위치 (1-2m 간격)로 효과적으로 '텔레포트' 하는 거친 불규칙 탐색 그래프를 사용합니다 . 기타 [9] 는 에이전트가 비어 있는 셀에서 이동하고 충돌이나 부분 단계가 없는 세분화된 일반 그리드 (0.01m 해상도) 를 사용합니다. Habitat 과 우리의 실험에서 우리는 보다 현실적인 충돌 모델을 사용합니다.

연속적인 상태 공간과 모션은 의도한 방향을 따라 부분적으로(또는 전혀) 진행하지 않는 충돌을 일으킬 수 있습니다 . 간단히 말해서 에이전트가 벽이나 장애물을 따라 '미끄러지는' 것이 가능합니다. 결정적으로 에이전트는 move_forward (0.25m) 를 선택하고 시작한 위치에서 0.25m 전방이 아닌 위치에 도달할 수 있습니다. 따라서 작동 소음이 없는 경우에도 주행 거리 측정이 사소하지 않습니다.

목표 사양: 정적 또는 동적? PointGoal 작업 [2] 에서 눈에 띄는 과소 사양 중 하나는 목표 좌표가 정적인지(즉, 에피소드 시작 시 한 번 제공됨) 또는 동적인지(즉, 매 단계마다 제공됨) 입니다 .

전자가 더 현실적입니다. 오라클이 정확한 동적 목표 좌표를 제공하는 실제 작업을 상상하기 어렵습니다 . 그러나 작동 소음 및 충돌이 없는 경우 에이전트가 취하는 모든 단계는 알려진 방향 전환 또는 변환으로 이어지며 이는 초기 목표 위치와 결합되어 기능적으로 동적 목표 지정과 동일합니다.

이것이 최근 작업 [16, 20, 13] 이 동적 목표 사양을 사용한 이유라고 가정합니다. 우리는 다음과 같은 개념 설명을 따르고 규정합니다. 작업으로 정적 PointGoal 탐색을 채택합니다. 센서 세트의 경우 상담원에게 이상적인 GPS+나침반 센서를 장착합니다. 이를 통해 현실적인 작업(정적 PointGoal 탐색)을 지향하고 작업 정의에서 시뮬레이터 설계(작동 소음, 충돌 역학)를 분리하고 사용된 센서(RGB, 깊이, GPS, 나침반, 접촉 센서) 별로 기술을 비교할 수 있습니다.).

감각 입력. 에이전트는 에이전트 베이스 중앙에서 1.5m 높이에 배치되고 '전방'을 향하도록 단일 컬러 비전 센서를 부여받습니다 . 이 센서는 256 픽셀 의 해상도와 90 도 시야의 RGB 프레임を提供합니다 . 또한 컬러 비전 센서와 동일한 위치 및 방향에서 이상적인 깊이 센서를 사용할 수 있습니다. 깊이 센서의 시야와 해상도는 컬러 비전 센서와 일치합니다.

컬러 센서를 활용하는 에이전트는 RGB로, 깊이 센서를 활용하는 에이전트는 Depth로, 둘 다 활용하는 에이전트는 RGBD로 지정합니다. 센서를 사용하지 않는 에이전트는 블라인드로 표시됩니다. 모든 에이전트는 이상적인 GPS와 나침반을 갖추고 있습니다. 즉, 위치 좌표에 액세스할 수 있으며 암시적으로 목표 위치를 기준으로 방향을 지정할 수 있습니다.

에피소드 사양. 환경 바닥의 탐색 가능한 모든 위치에서 무작위 로 균일하게 샘플링된 시작 위치 및 방향에서 에이전트를 초기화합니다 . 목표 위치는 같은 층에 있고 에이전트의 시작 위치에서 탐색 가능한 경로가 있도록 선택됩니다. 에피소드 동안 에이전트는 최대 500개의 작업을 수행할 수 있습니다. 이 임계값은 최적인 에이전트가 모든 목표를 달성하는 데 필요한 단계 수를 상당히 초과합니다 (보충 참조). 각 작업 후 에이전트는

활성 센서에서 일련의 관찰을 수신합니다.

평가. 내비게이션 에피소드는 에이전트 위치에서 목표 위치까지의 최단 경로를 따라 측지선 거리로 측정할 때 에이전트가 대상 좌표 의 0.2m 이내에서 정지 동작을 실행하는 경우에만 성공적인 것으로 간주됩니다 . 에이전트가 위의 조건을 충족하지 않고 500개의 작업을 수행 하면 에피소드가 종료되고 실패한 것으로 간주됩니다 . 성능은 ' SPL (Success weighted by Path Length)' 메트릭 [2] 을 사용하여 측정됩니다 . 최단 경로의 측지선 거리가 l 이고 에이전트가 거리 p 를 통과하는 에피소드의 경우 SPL은 $S \cdot l / \max(p, l)$ 로 정의되며 , 여기서 S는 성공의 이진 지표입니다.

에피소드 데이터 세트 준비. Matterport3D [8] 및 Gibson [30] 장면 에 대한 PointGoal 탐색 에피소드 데이터 세트를 생성합니다 . Matterport3D의 경우 공개적으로 사용 가능한 훈련/발행/테스트 분할을 따랐습니다 . 최근 작업 [9, 20, 16] 에서와 같이 train, val 및 테스트 장면 간에 겹치는 부분이 없습니다. Gibson 장면 의 경우 Gibson 저자 [30]로부터 질감이 있는 3D 표면 메시를 얻었고 각 장면의 재구성 품질(작은/큰 구멍, 떠다니는/불규칙한 표면, 열악한 질감)에 대해 수동으로 주석을 달고 106개 장면의 하위 집합을 선별했습니다 . 572의); 자세한 내용은 부록을 참조하십시오. 에피소드는 장면의 고유 ID, 에이전트의 시작 위치 및 방향, 목표 위치로 정의됩니다. 시작 위치에서 목표 위치까지의 최단 경로 (GDSP)를 따른 측지선 거리와 같은 추가 메타데이터 도 포함됩니다. 에피소드를 생성하는 동안 GDSP를 1m 에서 30m 사이로 제한합니다 . 시작 위치와 목표 위치 사이에 장애물 없는 직선이 있으면 에피소드는 사소한 것입니다. 에피소드의 탐색 복잡성에 대한 좋은 척도는 시작 위치와 목표 위치 사이의 GDSP 대 유클리드 거리의 비율입니다 (GDSP 는 유클리드 거리보다 크거나 같을 수 있음). 비율이 1에 가까우면 장애물이 적고 에피소드가 쉽습니다. 비율이 1보다 훨씬 크면 전략적 탐색이 필요하므로 에피소드가 어렵습니다 . 미리 계산된 에피소드의 탐색 복잡도를 합리적으로 높게 유지하기 위해 위의 비율이 [1, 1.1] 범위에 속하는 에피소드에 대해 거부 샘플링을 수행합니다 . 그 후, 거의 직선에 가까운 에피소드([1, 1.1])의 비율이 있는 에피소드)의 수가 Gibson 데이터 세트 생성의 경우 37% 에서 10% 로 크게 감소했습니다. 이 단계는 이전 연구에서 수행되지 않았습니다 . 이 필터링이 없으면 모든 메트릭이 부풀려진 것으로 나타납니다. Gibson 장면은 Matterport3D 장면에 비해 물리적 크기가 더 작습니다. 이는 결과 PointGoal 데이터 세트에 반영됩니다. Gibson 장면의 에피소드 평균 GDSP 는 Matterport3D 장면 보다 작습니다 .

기준선. 다음 기준선을 비교합니다. - Random은 균일한 분포로 turn_left, turn_right 및 move_forward 중에서 무작위로 작업을 선택합니다 . 에이전트는 목표에서 0.2m 이내일 때 정지 동작을 호출합니다(

정적 목표와 동적 GPS 좌표의 차이).

- 전진만 항상 move_forward 동작을 호출 하고 골대 0.2m 이내일 때 정지 동작을 호출합니다.
- 목표 팔로워가 목표 방향으로 이동합니다. 목표를 향하지 않는 경우(측에서 15도 이상), 왼쪽 방향 회전 또는 오른쪽 방향 전환 을 수행하여 정렬합니다. 그렇지 않으면 move_forward를 호출합니다. 에이전트는 목표에서 0.2m 이내일 때 중지 동작을 호출합니다.
- RL(PPO) 은 강화 학습, 특히 근접 정책 최적화 [25] 로 훈련된 에이전트입니다 . 시각 입력 없음 (Blind), RGB 입력, 깊이 입력 및 깊이가 있는 RGB (RGBD) 와 같은 다양한 시각 센서가 장착된 RL 에이전트를 실험합니다 . 모델은 시각적 입력을 위한 임베딩을 생성하는 CNN으로 구성되며 상대 목표 벡터와 함께 행위자 (GRU)와 비평가(선형 계층)에서 사용됩니다. CNN의 아키텍처는 {Conv 8×8 , ReLU, Conv 4×4 , ReLU, Conv 3×3 , ReLU, Linear, ReLU}입니다 (자세한 내용은 부록 참조). rt 는 시간 단계 t 에서의 보상 , dt는 시간 단계 t에서 목표까지의 측지선 거리, s 는 성공 보상 , λ는 시간 패널티(효율성을 높이기 위해)를 나타냅니다. 모든 모델은 다음 보상 기능으로 훈련되었습니다.

$$RT = \frac{s + dt}{dt + \lambda} \quad \begin{matrix} 1 & \text{dt} + \lambda \text{ 목표에 도달한 경우} \\ & \text{dt} + \lambda \text{ 그렇지 않은 경우} \end{matrix}$$

실험에서 s는 10 으로 설정되고 λ는 -0.01 로 설정됩니다 .

보상은 훈련 환경에서만 제공됩니다 . 에이전트가 보이지 않는 테스트 환경으로 일반화해야 하므로 작업이 까다롭습니다.

- SLAM [20] 은 RGB 및 깊이 센서를 사용하여 고전적인 로봇 탐색 파이프라인 (현저화, 매핑 및 계획을 위한 구성 요소 포함)을 구현하는 에이전트입니다 . 우리는 Mishkin 등의 클래식 에이전트를 사용합니다. [20] ORB-SLAM2 [21] 현저화 파이프라인을 활용 하며 원본 작업에서 보고된 것과 동일한 매개변수를 사용합니다.

교육 절차. 학습 기반 에이전트를 교육할 때 먼저 교육 세트의 장면을 동시에 실행 중인 시뮬레이터 작업자 스레드 8개 (Gibson), 6개 (Matterport3D)에 균등하게 나눕니다. 각 스레드는 훈련 세트 파티션의 각 장면에 대해 500개의 훈련 에피소드 블록을 설정 하고 이러한 블록의 순서를 섞습니다. 훈련은 이 어레이의 셔플된 복사본을 통해 계속됩니다. 일반성을 유지하고 GPS 입력을 가정하지 않는 향후 작업과 비교할 수 있도록 정지 동작을 하드코딩하지 않습니다 . 여기에 보고된 실험 의 경우 모든 작업자 스레드에서 7,500 만 에이전트 단계가 누적될 때까지 학습합니다 . 이것은 이전 조사에서 사용된 경험보다 15배 더 큼니다 [20, 16]. 에이전트를 7,500 만 단계까지 훈련했습니다(3개 데이터 세트 합계): 블라인드 의 경우 320 GPU 시간, RGB 의 경우 566 GPU 시간, 깊이 의 경우 475 GPU 시간, RGBD 의 경우 906 GPU 시간(전체 2267 GPU 시간).

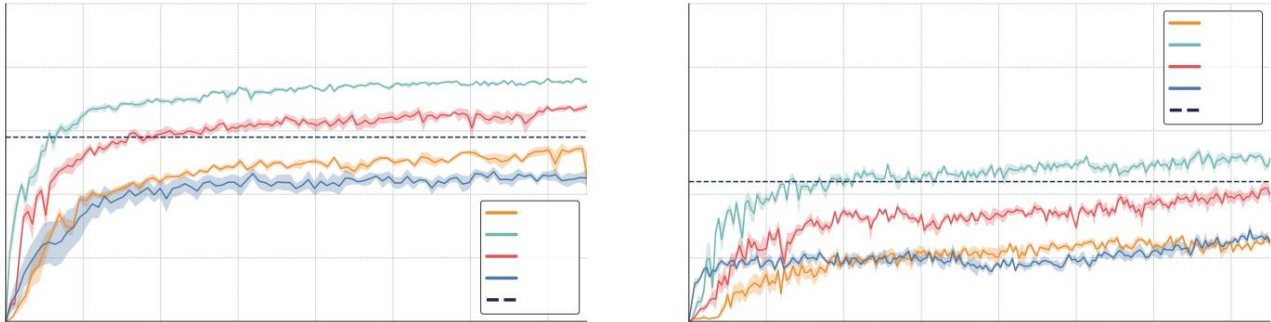


그림 3: 교육 과정 동안 설정된 값에 대한 상담원의 평균 SPL. 이전 작업 [20, 16] 은 5-1000만 단계에서 성능을 분석했습니다 . 더 많은 경험을 통해 흥미로운 추세가 나타납니다. i) 블라인드 에이전트는 초기에 RGB 및 RGBD를 능가 하지만 빠르게 포화됩니다. ii) 학습 기반 깊이 에이전트는 기존 SLAM을 능가합니다. 곡선 주변의 음영 영역은 5개 시드에 대한 SPL의 표준 오차를 나타냅니다.

센서	기준선	깊은		MP3D	
		SPL 성공	SPL 성공	SPL 성공	SPL 성공
눈이 먼	무작위의	0.02	0.03	0.01	0.00
	앞으로만	0.00	0.23	0.23	0.12
	목표 추종자	0.42	0.62	0.25	0.35
RGB	RL(PPO)	0.46	0.64	0.30	0.42
깊이 RL(PPO)		0.79	0.89	0.54	0.69
RGBD	RL(PPO)	0.70	0.80	0.42	0.51
	슬램 [20]	0.47		0.62	0.39

표 2: 여러 센서 구성 에서 Gibson [30] 및 MP3D [8] 테스트 세트에서 테스트 한 PointGoal 작업 [2] 에 대한 기본 방법의 성능. RL 모델은 7,500 만 단계에 대해 학습되었습니다 . 평균 에피소드 성공률과 SPL을 보고합니다 [2].

5. 결과 및 결과

i) 교육 경험의 양이 증가함에 따라 학습 기반 에이전트가 기존 SLAM 및 손으로 코딩된 기준선과 어떻게 비교되는지, 그리고 ii) 학습된 에이전트가 3D 데이터 세트에서 얼마나 잘 일반화되는지라는 두 가지 질문에 답하려고 합니다.

암묵적으로 이해해야 하지만 명시적으로 '학습'과 'SLAM'은 광범위한 기술군(단일 방법이 아니며 상호 배타적일 필요는 없으며 개발 과정에서 '정착'되지 않습니다. 우리는 스케일링 및 일반화 문제에 대한 통찰력을 얻기 위해 이러한 패밀리의 대표적인 사례를 비교하고 둘 중 하나의 본질적인 우월성에 대해 주장하지 않습니다 .

학습 대 SLAM. 첫 번째 질문에 답하기 위해 우리는 그림 3 (상단: Gibson, 하단: Matterport3D)의 교육 과정 동안 검증(즉, 보이지 않는) 에피소드에 대한 에이전트 성능 (SPL) 을 플롯합니다. SLAM [20] 은 교육이 필요하지 않으므로 일정한 성능(Gibson에서 0.59, Gibson에서 0.42)을 갖습니다.

Matterport3D). 모든 RL(PPO) 에이전트는 훨씬 더 나쁜 SPL로 시작하지만 특히 RL(PPO) 깊이는 극적으로 개선되고 약 10M 프레임(Gibson) 또는 30M 프레임(Matterport3D)의 경험에서 클래식 기준선과 일치하며 계속해서 이후 개선. [20] 에서와 같이 5M 프레임에서 실험을 종료하면 SLAM [20] 이 우세하다는 결론도 내릴 수 있습니다 . 흥미롭게도 RGB 에이전트는 Blind 에이전트를 크게 능가하지 않습니다 . 둘 다 GPS 센서가 장착되어 있기 때문에 우리는 가설을 세웁니다 . 실제로 질적 결과(그림 4 및 보충 비디오)는 블라인드 에이전트가 벽을 '꺼안고' '벽을 따라가는' 휴리스틱을 구현함을 시사합니다. 대조적으로, RGB 센서는 장면 전체의 다양성으로 인해 (동일한 데이터 세트 내에서도) 훈련 환경에 과적합되기 쉬운 고차원의 복잡한 신호를 제공합니다 . 또한 그림 3 에서 모든 방법이 Matterport3D보다 Gibson에서 더 잘 수행됨을 알 수 있습니다. 이것은 Gibson이 더 작은 장면과 더 짧은 에피소드를 포함한다는 이전 분석과 일치합니다 .

다음으로, 각 에이전트 및 데이터 세트에 대해 유효성 검사에서 가장 성능이 좋은 체크포인트를 선택하고 테스트 결과를 표 2에 보고합니다. 데이터 세트 전체에서 균일하게 RL(PPO) 깊이가 RL(PPO)보다 성능이 가장 우수함을 관찰합니다.

RGBD(0.09-0.16 SPL), SLAM(0.15-0.28 SPL), RGB (0.13-0.33 SPL) 순서대로 (노이즈 깊이와 관련된 추가 실험은 보충 자료 참조). i) PointGoal 탐색 작업은 여유 공간에 대한 추론만 필요하고 깊이는 관련 정보를 직접 제공하고 ii) RGB가 훨씬 더 많은 엔트로피(집마다 매우 다르게 보임)를 갖기 때문에 Depth가 RGBD보다 성능이 더 우수하다고 생각합니다 . RGB를 사용합니다 . 우리는 이러한 차이가 통계적으로 유의미하다는 것을 확인하기 위해 실험당 5개의 임의의 시드를 사용하여 실험을 실행했습니다. 차이는 모든 경우에 대해 평균 SPL의 표준 편차보다 한 자릿수 더 큼니다(예: Gibson 데이터 세트 오류는 깊이가 ±0.015, RGB: ±0.055, RGBD: ±0.028, 블라인드: ±0.005). 랜덤 및 전단 전용 에이전트는

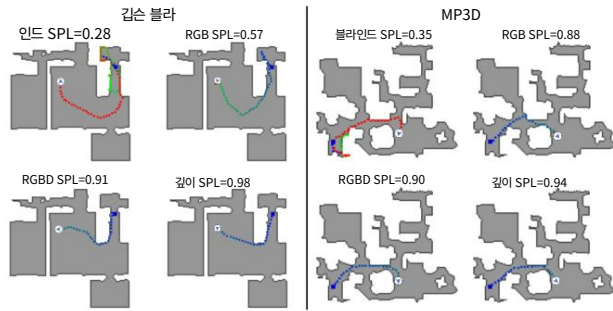


그림 4: RL(PPO) 에이전트의 다양한 감각 구성에 대한 탐색 예, Gibson 및 MP3D val 세트의 시도를 시각화합니다. 파란색 점과 빨간색 점은 시작 및 목표 위치를 나타내고 파란색 화살표는 최종 에이전트 위치를 나타냅니다.

청록색 선은 에이전트의 궤적입니다. 최대 에이전트 단계 수에 도달하면 색상이 파란색에서 빨간색으로 바뀝니다.

더 많은 예제 궤적은 보충 자료를 참조하십시오.

	깊은	MP3D
블라인드 깊은		0.34
MP3D	0.28	0.25
RGB 깊은	0.46	0.40
MP3D	0.25	0.30
깊이 깊은	0.79	0.68
MP3D	0.56	0.54
RGBD 깊은	0.70	0.53
MP3D	0.44	0.42

그림 5: 데이터 세트 간의 에이전트 일반화. 각 열의 대상 데이터 세트에 대한 테스트 에피소드에서 평가된 대로 각 행의 소스 데이터 세트에서 훈련된 모델의 평균 SPL을 보고합니다.

매우 낮은 성능을 보이는 반면 수동으로 코딩한 목표 추종자 및 블라인드 기준선은 적당한 성능을 보입니다. 훈련된 상담원 행동에 대한 추가 분석은 부록을 참조하십시오.

그림 4에서는 RL(PPO) 에이전트에 대한 예제 궤적을 플롯하여 동일한 에피소드에서 그들의 행동을 질적으로 대조합니다. 집계 통계와 일관되게 우리는 Blind가 장애물과 충돌하고 벽을 따라가는 반면 Depth가 가장 효율적이라는 것을 관찰했습니다. 더 많은 예제 궤적은 부록과 비디오를 참조하십시오.

데이터 세트 전반에 걸친 일반화. 지금까지 우리의 발견은 RL(PPO) 에이전트가 SLAM을 훨씬 능가한다는 것입니다 [20]. 이것은 우리의 두 번째 질문을 촉발합니다. 이러한 결과 데이터 세트는 특정한 것입니까, 아니면 학습된 에이전트가 데이터 세트 전체에 걸쳐 일반화합니까? 우리는 그림 5에서 철저한 비교를 보고합니다. 특히 모든 에이전트 {Blind, RGB, RGBD, Depth}에 대한 {train, test} × {Matterport3D, Gibson}의 모든 조합에 대한 평균 SPL입니다. 행은 (에이전트, 기차 세트) 쌍을 나타내고 열은 테스트 세트를 나타냅니다. 흥미로운 트렌드가 많이 있습니다. 첫째, RGBD Gibson→Gibson 0.70 대 RGBD Gibson→Matterport3D 0.53 (0.17 하락)과 같이 거의 모든 에이전트가 하나의 데이터 세트에서 훈련되고 다른 데이터 세트에서 테스트될 때 성능 저하를 겪습니다. RGB 및 RGBD 에이전트는 상당한 어려움을 겪습니다.

Blind 에이전트는 예상대로 가장 적은 영향을 받는 반면 성능은 저하됩니다.

둘째, 직관에 반할 가능성이 있는 경향을 발견했습니다. 즉, Gibson에서 훈련된 에이전트는 Matterport3D에서 평가할 때에도 Matterport3D에서 훈련된 에이전트보다 지속적으로 우수한 성능을 보였습니다. 우리는 Gibson 장면이 Matterport3D보다 더 작고 에피소드가 더 짧다는(더 낮은 GDSP) 이전에 언급된 관찰 결과가 그 이유라고 생각합니다. Gibson 에이전트는 '더 쉬운' 에피소드에 대해 교육을 받고 무작위 탐색 중에 더 쉽게 긍정적인 보상을 만나 학습을 부트스트래핑합니다. 결과적으로 고정 계산 예산의 경우 Gibson 에이전트는 보편적으로 더 강력합니다(Gibson에서만 아님). 이 발견은 시각적 내비게이션 에이전트가 커리큘럼 학습에서 이점을 얻을 수 있음을 시사합니다.

이러한 통찰력은 이러한 실험을 평가 데이터 세트 이름을 변경하는 것만큼 간단하게 만든 Habitat의 엔지니어링을 통해 가능합니다.

6. 해비타트 챌린지

어떤 전투 계획도 적과의 접촉에서 살아남지 못합니다.

헬무트 칼 베른하르트 폰 몰트케

도전이 발전을 이끕니다. AI 하위 분야의 역사는 올바른 질문의 공식화, 올바른 데이터 세트의 생성, 올바른 문제를 둘러싼 커뮤니티의 결합이 과학적 진보를 이끈다는 것을 나타냅니다.

우리의 목표는 구체화된 AI를 위해 이 프로세스를 지원하는 것입니다. 해비타트 챌린지는 목표 지향적인 시각적 탐색의 노력을 벤치마킹하고 발전시키는 것을 목표로 하는 자율 탐색 챌린지입니다.

구현된 AI 작업에 대한 도전 과제를 생성하는 한 가지 어려움은 정적 예측(수동적 인식에서와 같이)에서 순차적 의사 결정(감각 운동 제어에서와 같이)으로 전환하는 것입니다. 기존의 '인터넷 AI' 과제(예:

ImageNet [10], COCO [18], VQA [4]) 정적 테스트 데이터 세트를 릴리스하고 참가자에게 이 세트에 예측을 업로드하도록 요청할 수 있습니다. 대조적으로 구현된 AI 작업에는 일반적으로 순차적인 의사 결정 및 에이전트 기반 제어가 포함되므로 테스트 데이터 세트를 미리 패키징하는 것이 불가능합니다. 본질적으로 구현된 AI 과제는 참가자가 예측이 아닌 코드를 업로드하도록 요구합니다. 그런 다음 업로드된 에이전트를 새로운(보이지 않는) 테스트 환경에서 평가할 수 있습니다.

인프라에 도전하십시오. 우리는 EvalAI 플랫폼의 프론트엔드 및 챌린지 제출 프로세스를 활용하고 백엔드 인프라를 직접 구축합니다. Habitat Challenge 참가자는 EvalAI를 통해 에이전트와 함께 Docker 컨테이너 [19]를 업로드해야 합니다. 제출된 에이전트는 라이브 AWS GPU 지원 인스턴스에서 평가됩니다. 특히, 참가자는 원하는 대로 에이전트를 자유롭게 교육할 수 있습니다 (모든 언어, 모든 프레임워크, 모든 인프라). 이러한 에이전트를 평가하기 위해 참가자는 기본 Habitat Docker 컨테이너에서 파생되고 특정 사양을 구현하도록 요청받습니다.

그들의 모델에 대한 cific 인터페이스 – 에이전트가 각 단계에서 환경으로부터 관찰을 통해 취한 행동. 이 도커화 된 인터페이스를 통해 새로운 환경에서 참여자 코드를 실행할 수 있습니다 .

CVPR 2019에서 개최되는 Habitat Challenge에 대한 자세한 내용은 <https://aihabitat.org/challenge/> 웹사이트. 이 챌린지의 향후 반박에서 우리는 시뮬레이션과 현실 사이의 격차를 줄이고 작업의 난이도를 높이기 위해 설계된 세 가지 주요 차이점을 소개할 것입니다.

– 2019년 챌린지에서 목표를 지정하는 상대 좌표는 에이전트 이동 중에 지속적으로 업데이트되었습니다. 즉, 본질적으로 완벽한 현지화 및 방향 추정으로 에이전트를 시뮬레이션합니다 (예: 이상화된 GPS+나침반이 있는 에이전트). 그러나 실내 환경의 고정밀 위치 파악은 실제 환경에서 가정할 수 없습니다. GPS는 실내에서 정확도가 낮고, (시각적) 주행 거리계에 잡음이 많을 수 있으며, SLAM 기반 위치 파악에 실패할 수 있습니다. 따라서 우리는 에이전트에게 제공하는 것만 조사할 것입니다. 시각 위치에서 목표 위치에 대한 고정 상대 좌표 .

– 마찬가지로 2019 Habitat Challenge는 에이전트 동작(예: 앞으로, 왼쪽으로 10° 회전 ,...)을 최소한으로 모델링했습니다. 그러나 실제 설정에서는 에이전트 의도 (예: 1m 앞으로 이동)와 결과가 완벽하게 일치하는 경우가 거의 없습니다. 작동 오류, 다양한 표면 재료 및 무수한 기타 오류 소스로 인해 긴 궤적에서 상당한 드리프트가 발생합니다. 이를 모델링하기 위해 실제 로봇 플랫폼을 벤치마킹하여 얻은 노이즈 모델을 소개합니다 [22].

시각적 감지는 이 "데드레커닝" 드리프트를 방지하는 훌륭한 수단이며, 이러한 변화를 통해 참가자는 이 노이즈 에 강력하고 수정할 수 있는 방법론을 연구할 수 있습니다.

– 마지막으로 RGB 및 깊이 센서에 대한 현실적인 센서 노이즈 모델을 소개하여 에이전트가 시뮬레이션 과 현실 에서 가질 수 있는 지각 경험 간의 격차를 좁힐 것입니다 .

구현된 내비게이션 에이전트에 대한 최신 방법을 평가하기 위한 벤치마크를 설정하는 커뮤니티를 지원하기를 기대합니다 .

7. 향후 작업

Habitat 플랫폼의 설계 및 구현에 대해 설명했습니다 . 우리의 목표는 기존 커뮤니티의 노력을 통합 하고 구현된 AI에 대한 연구를 가속화하는 것입니다. 이것은 광범위한 연구 커뮤니티의 완전한 참여를 통해서만 성공할 수 있는 장기적인 노력입니다.

일반 데이터 세트 지원 및 Habitat 스택의 고성능에 의해 가능해진 실험은 i) 학습 기반 에이전트가 충분히 오랫동안 훈련되었을 때 고전적인 시각적 탐색 방법의 성능과 일치하거나 초과할 수 있고 ii) 깊이가 있는 학습된 에이전트를 나타냅니다.

센서는 RGB만 장착된 에이전트와 비교하여 서로 다른 3D 환경 데이터 세트 간에 잘 일반화됩니다 .

기능 로드맵. 우리의 단기 개발 로드맵은 물리 시뮬레이션을 통합하고 3D 환경에서 모바일 에이전트와 개체 간의 물리 기반 상호 작용을 활성화하는 데 중점을 둘 것입니다. Habitat-Sim의 장면 그래프 표현은 물리 엔진과의 통합에 매우 적합하여 장면 그래프 내에서 개별 개체 및 에이전트의 상태를 직접 제어할 수 있습니다 . 미래 작업의 또 다른 계획된 방법은 3D 재구성 과 가상 개체 데이터 세트의 조합을 활용하여 3D 환경 의 절차적 생성을 포함합니다 . 넓은 실내 공간의 고품질 재구성과 별도로 재구성되거나 모델링된 객체를 결합함으로써 우리는 계층적 장면 그래프 표현을 최대한 활용하여 시뮬레이션된 3D 환경에서 제어된 변형을 도입 할 수 있습니다.

마지막으로 경쟁 또는 협업 시나리오에서 잠재적으로 서로 상호 작용하는 많은 수의 에이전트를 포함하는 분산 시뮬레이션 설정에 중점을 둘 계획입니다 .

감사의 말 유익한 제안을 해주신 심사위원들께 감사드립니다. Habitat 프로젝트는 많은 개인의 지원과 기여 없이는 불가능했을 것입니다 . Mandeep Baines, Angel Xuan Chang, Alexander Clegg, Devendra Singh Chaplot, Xin lei Chen, Wojciech Galuba, Georgia Gkioxari, Daniel Gordon, Leonidas Guibas, Saurabh Gupta, Jerry (Zhi-Yang) He, Rishabh Jain, Or에게 감사드립니다. 도움을 주신 Litany, Joel Marcey, Dmytro Mishkin, Marcus Rohrbach, Amanpreet Singh, Yuandong Tian, Yuxin Wu, Fei Xia , Deshraj Yadav, Amir Zamir, Jiazhi Zhang .

참조된 데이터 세트에 대한 라이선스.

김슨: https://storage.googleapis.com/gibson_material/Agreement%20GDS%2006-04-18.pdf Matterport3D: http://kaldir.vc.in.tum.de/matterport/MP_TOS.pdf .

참조

- [1] Phil Ammirato, Patrick Poirson, 박은병, Jana Kosecká, Alexander C Berg. 활성 시력을 개발하고 벤치마킹하기 위한 데이터 세트입니다. 2017년 ICRA에서.
- [2] Peter Anderson, Angel X. Chang, Devendra Singh Chaplot, Alexey Dosovitskiy, Saurabh Gupta, Vladlen Koltun, Jana Kosecka, Jitendra Malik, Roozbeh Mottaghi, Manolis Savva 및 Amir Roshan Zamir. 내비게이션 에이전트 평가. arXiv:1807.06757, 2018.
- [3] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, Anton van den Hengel. 시각 및 언어 내비게이션: 실제 환경에서 시각적 기반 내비게이션 지침을 해석합니다. CVPR, 2018.
- [4] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, Devi Parikh. VQA: 시각적 질문 답변. 2015년 ICCV에서.
- [5] Iro Armeni, Ozan Sener, Amir R. Zamir, Helen Jiang, Ioannis Brilakis, Martin Fischer, Silvio Savarese. 대규모 실내 공간의 3D 스캔팅 파싱. CVPR, 2016.
- [6] Alex Bewley, Jessica Rigley, Yuxuan Liu, Jeffrey Hawke, Richard Shen, Vinh-Dieu Lam, Alex Kendall. 실제 라벨 없이 시뮬레이션에서 운전하는 법을 배웁니다. 2019년 ICRA에서.
- [7] Simon Brodeur, Ethan Perez, Ankesh Anand, Florian Golemo, Luca Celotti, Florian Strub, Jean Rouat, Hugo Larochelle, Aaron C. Courville. HoME: 가정용 복합기 환경. arXiv:1711.11017, 2017.
- [8] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, Yinda Zhang. Matterport3D: 실내 환경에서 RGB-D 데이터로부터 학습합니다. 2017년 3DV(International Conference on 3D Vision)에서.
- [9] Abhishek Das, Samyak Datta, Georgia Gkioxari, Stefan Lee, Devi Parikh 및 Dhruv Batra. 구현된 질문 답변. CVPR, 2018.
- [10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li 및 Fei-Fei Li. ImageNet: 대규모 계층적 이미지 데이터베이스입니다. CVPR, 2009.
- [11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova. BERT: 언어 이해를 위한 심층 양방향 트랜스포머 사전 훈련. arXiv:1810.04805, 2018.
- [12] 데이비드 도노호. 데이터 과학의 50년. 2015년 투키 100주년 워크숍에서.
- [13] Saurabh Gupta, James Davidson, Sergey Levine, Rahul Sukthankar, Jitendra Malik. 시각적 탐색을 위한 인지 매핑 및 계획. CVPR, 2017.
- [14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. 이미지 인식을 위한 심층 잔차 학습. CVPR, 2016.
- [15] 황보 제민, 이준호, 알렉세이 도소비츠키, 다리오 벨리코소, 바실리오스 추니스, 블라들렌 쿨튼, 마르코 후터. 다리 로봇을 위한 민첩하고 역동적인 운동 기술을 학습합니다. 사이언스 로봇틱스, 2019.
- [16] 코지마 노리유키와 덩 자아. 배울 것인가 말 것인가: 가상 환경에서 내비게이션을 위한 학습의 역할 분석. arXiv:1907.11770, 2019.
- [17] Eric Kolve, Roozbeh Mottaghi, Daniel Gordon, Yuke Zhu, Abhinav Gupta 및 Ali Farhadi. AI2-THOR: 시각적 AI를 위한 대화형 3D 환경입니다. arXiv:1712.05474, 2017.
- [18] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, C. Lawrence Zitnick. Microsoft COCO: 컨텍스트의 공통 객체. ECCV, 2014.
- [19] 더크 메르켈. Docker: 일관된 개발 및 배포를 위한 경량 Linux 컨테이너입니다. 라눅스 저널, 2014.
- [20] Dmytro Mishkin, Alexey Dosovitskiy, Vladlen Koltun. 복잡한 3D 환경에서 클래식 및 학습된 탐색을 벤치마킹합니다. arXiv:1901.10915, 2019.
- [21] 라울 무르-아탈과 후안 D. 타르도스. ORB-SLAM2: 단안, 스테레오 및 RGB-D 카메라용 오픈 소스 SLAM 시스템입니다. 로보틱스에 관한 IEEE 트랜잭션, 33(5), 2017.
- [22] Adithyavairavan Murali, Tao Chen, Kalyan Vasudev Alwala, Dhiraj Gandhi, Lerrel Pinto, Saurabh Gupta 및 Abhinav Gupta. Pyrobot: 연구 및 벤치마킹을 위한 오픈 소스 로봇 프레임워크입니다. arXiv 프리프린트 arXiv:1906.08236, 2019.
- [23] Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, Antonio Torralba. VirtualHome: 프로그램을 통한 가정 활동 시뮬레이션. CVPR, 2018.
- [24] Manolis Savva, Angel X. Chang, Alexey Dosovitskiy, Thomas Funkhouser, Vladlen Koltun. MINOS: 복잡한 환경에서 탐색을 위한 다중 실내 시뮬레이터. arXiv:1712.03931, 2017.
- [25] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, Oleg Klimov. 근접 정책 최적화 알고리즘. arXiv:1707.06347, 2017.
- [26] 린다 스미스와 마이클 개서. 체화된 인지 발달: 아가에게서 배우는 여섯 가지 교훈. 인공지능, 11(1-2), 2005.
- [27] Shuran Song, Fisher Yu, Andy Zeng, Angel X Chang, Manolis Savva, Thomas Funkhouser. 단일 깊이 이미지에서 스캔팅 장면 완성. CVPR, 2017.
- [28] Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijkman, Simon Green, Jakob J. Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, Anton Clarkson, Mingfei Yan, Brian Budge, Yajie Yan, Xiaqing Pan, June Yon, Yuyang Zou, Kimberly Leon, Nigel Carter, Jesus Briales, Tyler Gillingham, Elias Mueggler, Luis Pesqueira, Manolis Savva, Dhruv Batra, Hauke M. Strasdat, Renzo De Nardi, Michael Goesele, Steven Lovegrove, Richard Newcombe. Replica 데이터 세트: 실내 공간의 디지털 복제본. arXiv:1906.05797, 2019.
- [29] Yi Wu, Yuxin Wu, Georgia Gkioxari, Yuandong Tian. 사실적이고 풍부한 3D 환경으로 일반화 가능한 에이전트를 구축합니다. arXiv:1801.02209, 2018.
- [30] Fei Xia, Amir R. Zamir, Zhiyang He, Alexander Sax, Jitendra Malik 및 Silvio Savarese. Gibson env: 구체화된 에이전트에 대한 실제 인식. CVPR, 2018.
- [31] Claudia Yan, Dipendra Misra, Andrew Bennett, Aaron Walsman, Yonatan Bisk, Yoav Artzi. 살라: 코넬 하우스 에이전트 학습 환경. arXiv:1801.07357, 2018.

가. 해비타트 플랫폼 세부사항

주요 문서에 설명된 대로 Habitat은 다음 구성 요소로 구성됩니다. • Habitat-Sim: 구성 가능한 에이전트, 여러 센서 및 일반 3D 데이터 세트 처리가 포함된 유연한 고성능 3D 시뮬레이터(Matterport3D [8 내장 자원 포함]), Gibson [30] 및 기타 데이터 세트). Habitat-Sim은 빠릅니다. Matterport3D 데이터 세트에서 사실적인 스캔 장면을 렌더링할 때 Habitat-Sim은 단일 스레드로 실행되는 초당 수천 프레임(fps)을 달성하고 단일 GPU에서 10,000fps 이상의 다중 프로세스에 도달할 수 있습니다. • Habitat-API: 구현된 AI의 엔드 투 엔드 개발을 위한 모듈식 고급 라이브러리 – 구현된 AI 작업 정의(예: 탐색 [2], [3]에 따른 지침, 질문 응답 [9]), 구현된 에이전트 구성 (물리적 형태, 센서, 기능), 이러한 에이전트 교육(모방 또는 강화 학습 또는 클래식 SLAM을 통해), 표준 메트릭을 사용하여 정의된 작업에 대한 성능 벤치마킹 [2].

Habitat-API는 현재 Habitat-Sim을 코어 시뮬레이터로 사용하지만 여러 시뮬레이터에 대한 호환성을 유지하기 위해 시뮬레이터 백엔드용 모듈식 추상화로 설계되었습니다.

주요 추상화. Habitat 플랫폼은 3차원 실내 환경에서 수행할 수 있는 구현된 에이전트 및 작업의 도메인을 모델링하는 여러 주요 추상화에 의존합니다. 여기에서는 주요 추상화에 대한 간략한 요약을 제공합니다. • 에이전트: 일련의 센서가 있는 물리적으로 구현된 에이전트입니다. 환경을 관찰할 수 있고 에이전트 또는 환경 상태를 변경하는 작업을 수행할 수 있습니다. • 센서: 지정된 빈도로 환경에서 관찰 데이터를 반환할 수 있는 특정 에이전트와 연결됩니다.

- SceneGraph: 프로그래밍 방식으로 조작할 수 있는 영역 및 개체로 환경을 구성하는 3D 환경의 계층적 표현입니다.

- 시뮬레이터: 시뮬레이터 백엔드의 인스턴스입니다. 구성된 일련의 에이전트 및 SceneGraph에 대한 작업이 주어지면 에이전트 및 SceneGraph의 상태를 업데이트하고 에이전트가 보유한 모든 활성 센서에 대한 관찰을 제공할 수 있습니다.

이러한 추상화는 플랫폼의 서로 다른 계층을 연결합니다. 또한 구현된 AI 작업의 일반적이고 이식 가능한 사양을 가능하게 합니다.

Habitat-Sim. Habitat-Sim 백엔드 모듈의 아키텍처는 그림 6에 설명되어 있습니다. 이 모듈의 설계는 몇 가지 주요 속성을 보장합니다.

- 3D 환경 리소스(삼각형 메시 지오메트리, 텍스처, 셰이더)의 메모리 효율적인 관리로 공유 리소스를 캐시하고 재사용할 수 있습니다.

- SceneGraphs를 사용하여 3D 환경을 유연하고 구조적으로 표현하여 개체 상태를 프로그래밍 방식으로 조작하고 다양한 환경의 개체를 결합할 수 있습니다.

- 다중 센서의 오버헤드를 줄이기 위한 다중 첨부 렌더 패스가 포함된 고효율 렌더링 엔진. • SceneGraph에 연결하여 3D 환경에 연결할 수 있는 에이전트 및 해당 센서의 임의 개수.

시뮬레이션 백엔드의 성능은 실제 재구성 데이터 세트에서 작동하는 이전 작업의 성능을 크게 증가합니다. 표 3은 Matterport3D 데이터 세트의 테스트 장면에 대한 성능 통계를 보고합니다. 단일 스레드 성능은 초당 수천 프레임(fps)에 도달하는 반면 여러 시뮬레이션 백엔드가 있는 다중 프로세스 작업은 단일 GPU에서 10,000fps 이상에 도달할 수 있습니다. 또한 OpenGL-CUDA 상호 운용성을 채택함으로써 이미지 해상도가 증가함에 따라 성능에 측정 가능한 영향을 미치지 않고 PyTorch와 같은 ML 프레임워크와 렌더링된 이미지 프레임을 직접 공유할 수 있습니다 (그림 7 참조).

해비타트-API. Habitat 플랫폼의 두 번째 계층 (Habitat-API)은 구현된 에이전트, 수행할 수 있는 작업 및 해당 작업에 대한 평가 메트릭을 정의하기 위한 일반적이고 유연한 API를 만드는 데 중점을 둡니다. 이러한 API를 설계할 때 주요 고려 사항은 정의된 추상화를 쉽게 확장할 수 있도록 하는 것입니다. 이는 구현된 에이전트 작업, 특정 에이전트 구성 및 3D 환경 설정의 많은 매개변수가 흥미로운 방식으로 변경될 수 있기 때문에 특히 중요합니다. 향후 연구에서는 새로운 작업, 새로운 에이전트 구성 및 새로운 3D 환경을 제안할 가능성이 높습니다.

API를 사용하면 우리가 구현한 Habitat-Sim 모듈 외에 대체 시뮬레이터 백엔드를 사용할 수 있습니다. 이 모듈성은 이전 작업에서 레거시 프레임워크를 사용하여 수행한 실험에서 전환하는 데 도움이 되도록 기존 시뮬레이터 백엔드를 통합할 수 있는 이점이 있습니다. Habitat-API의 아키텍처는 그림 8에 나와 있으며 핵심 API 기능과 핵심에 대한 확장으로 구현된 기능을 나타냅니다.

API 수준 위에서 시각적 탐색과 같은 구체적으로 구현된 작업을 정의합니다. 여기에는 특정 데이터 세트 구성 정의, 에피소드 구조 (예: 수행한 단계 수, 종료 조건), 교육 커리큘럼(에피소드 진행, 난이도 램프) 및 평가 절차 (예: 테스트 에피소드 세트 및 작업 메트릭) 지정이 포함됩니다.

미리 구성된 작업(PointNav)을 로드하고 임의의 에이전트로 환경을 단계별로 실행하는 예가 아래 코드에 나와 있습니다.

5참고: Matterport3D의 시맨틱 센서는 기하학적 복잡성이 훨씬 더 많은 추가 3D 메시를 사용해야 하므로 성능이 저하됩니다. 우리는 이 문제가 향후 버전에서 해결되어 RGB + 깊이에 필적하는 속도로 이어질 것으로 기대합니다.

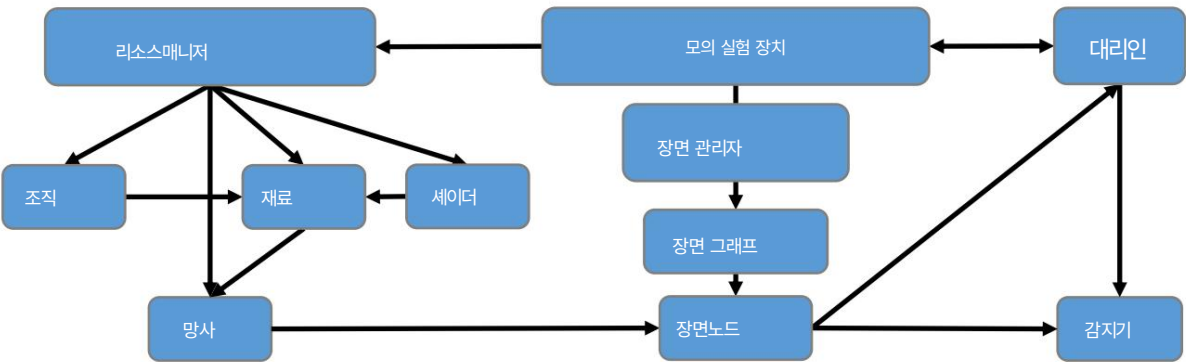


그림 6: Habitat-Sim 메인 클래스의 아키텍처. 시뮬레이터는 3D 환경과 관련된 모든 리소스의 관리를 다양한 온디스크 형식의 3D 환경 데이터 로드 및 캐싱을 담당하는 ResourceManager에 위임합니다. 이러한 리소스는 특정 장면에서 고유한 개체 또는 영역을 나타내는 개별 SceneNode 수준의 SceneGraph 내에서 사용됩니다. 에이전트와 해당 센서는 특정 SceneGraph의 SceneNode에 연결되어 인스턴스화됩니다.

센서 / 프로세스 수	GPU→CPU→GPU			GPU→CPU			GPU→GPU		
	1	삼	5	1	삼	5	1	삼	5
RGB	2,346	6,049	7,784	3,919	8,810	11,598	4,538	8,573	7,279
RGB + 깊이	4,307	5,522	2,151	3,557	3,486	464	453		
RGB + 깊이 + 의미론5	378	463	470	396	465	466	455		

표 3: Xeon E5-2690 v4 CPU 및 Nvidia Titan Xp GPU의 Matterport3D 장면(id 17DRP5sb8fy)에 대한 초당 프레임 수의 Habitat-Sim 성능, 128x128의 프레임 해상도에서 측정, 다양한 프레임 메모리 전송 전략 및 다양한 수의 동시 시뮬레이터 프로세스가 GPU를 공유합니다. 'GPU-CPU-GPU'는 최적화에 사용하기 위해 OpenGL 컨텍스트에서 CPU 호스트 메모리로 그리고 다시 GPU 장치 메모리로 렌더링된 프레임들을 전달하는 것을 나타내며, 'GPU-CPU'는 OpenGL 컨텍스트에서 CPU 호스트 메모리의 복사만 보고하는 반면 'GPU- GPU'는 OpenGL-CUDA 연동을 통한 직접 공유를 나타냅니다.

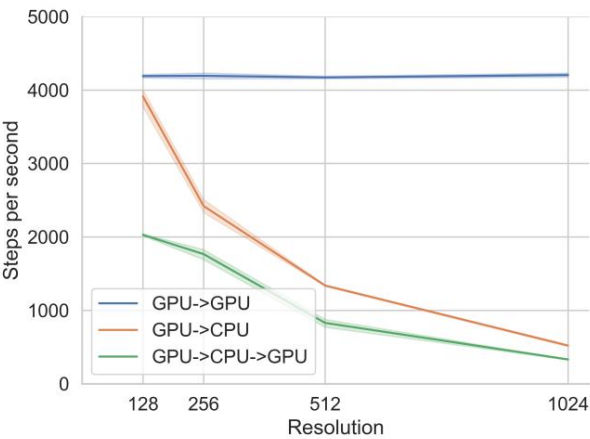


그림 7: 이미지 해상도를 높이기 위해 다양한 센서 프레임 메모리 전송 전략 하에서 Habitat-Sim의 성능. 'GPU->GPU'는 이미지 해상도에 영향을 받지 않는 반면 다른 전략은 빠르게 저하됩니다.

B. 추가 데이터 세트 통계

표 5에는 실험에 사용된 세 가지 데이터 세트 모두에 대한 학습, 검증 및 테스트 분할 크기가 요약되어 있습니다. 또한 우리

시작점과 목표 위치 사이의 최단 경로 (GDSP)를 따라 평균 측정된 거리를 보고합니다. 주요 문서에서 언급한 바와 같이 Gibson 에피소드는 Matterport3D 에피소드보다 훨씬 짧습니다. 그림 9는 측정된 거리 (GDSP), 시작 위치와 목표 위치 사이의 유클리드 거리, 둘의 비율 (에피소드의 대략적인 복잡성 측정)에 대한 에피소드 분포를 시각화합니다.

Gibson 에피소드에는 거리가 짧은 에피소드가 더 많기 때문에 Matterport3D 데이터 세트보다 데이터 세트가 전반적으로 더 쉽습니다.

```
수입 서식지

# 구현된 AI 작업 로드(PointNav) # 미리 지정된 가상 로봇 config =
habit.get_config(config_file="pointnav.yaml")

env = 서식지.Env(config)

관찰 = env.reset()

# env.episode_over가 아닌 동안 임의의 작업으로 환경을 단계별로 실행합니다 : observes = \
env.step(env.action_space.sample())
```

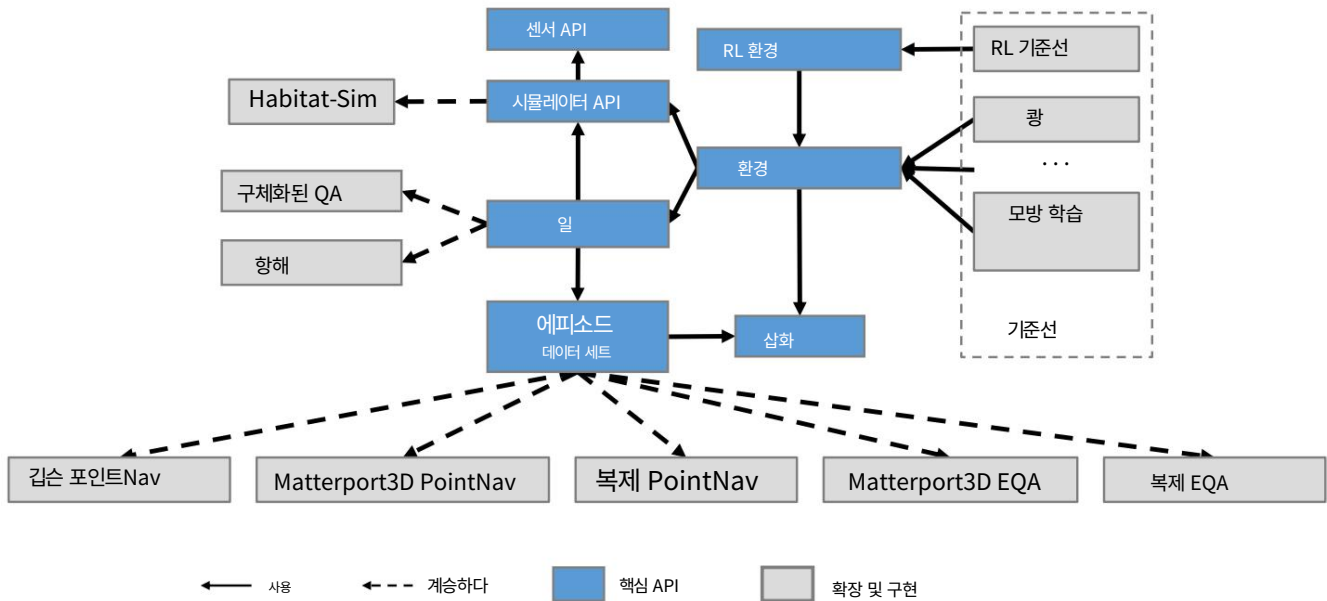


그림 8: Habitat-API의 아키텍처. 핵심 기능은 시뮬레이터 백엔드와 상호 작용하고 센서를 통해 관찰을 수신하기 위한 API와 같은 기본 빌딩 블록을 정의합니다. 구체적인 시뮬레이션 백엔드, 3D 데이터 세트 및 구현된 에이전트 기준선은 핵심 API에 대한 확장으로 구현됩니다.

데이터 세트	장면(#)	에피소드(#)	평균 GDP(m)
Matterport3D 58 / 11 / 18 4.8M / 495 / 1008			11.5 / 11.1 / 13.2
깁슨	72 / 16 / 10 4.9M / 1000 / 1000		6.9 / 6.5 / 7.0

표 4: Matterport3D 및 Gibson 데이터 세트에 대해 미리 계산한 PointGoal 탐색 데이터 세트의 통계: 총 장면 수, 총 에피소드 수 및 시작 위치와 목표 위치 사이의 평균 최단 거리. 각 셀은 훈련/발행/테스트 분할 통계를 보고합니다.

데이터 세트	최소 중앙값 평균 최대값
Matterport3D	18 90.0 97.1 281
깁슨	15 60.0 63.3 207

표 5: PointGoal 내비게이션 검증 세트에서 최단 거리 기술자의 음수를 따르기 위해 조치를 탐색스럽게 맞추는 오라클의 경로 길이(조치 내) 통계. 이것은 거의 완벽에 가까운 에이전트에 대한 예상 수평선 길이를 제공 하고 최대 단계 제한 500에 대한 결정을 맥락화합니다.

다. 추가 실험 결과

논문에 제시된 실험 결과에 대해 관찰한 경향이 훨씬 더 많은 경험을 유지하는지 확인하기 위해 실험을 800M 단계로 확장했습니다. 우리는 (1) 시각적 입력의 순서가 Depth > RGBD > RGB > Blind로 유지됨을 발견했습니다. (2) RGB는 Blind보다 지속적으로 우수하고 (Gibson/Matterport3D에서 0.06/0.03 SPL), (3) RGBD는 Matterport3D에서 SLAM보다 성능이 우수합니다(0.16 SPL).

다.1. 충돌 분석

내비게이션 중 학습된 에이전트의 동작을 더 특성화하기 위해 그림 10에 평균 충돌 수를 표시합니다. Blind가 다른 에이전트보다 훨씬 많은 수의 충돌을 발생시켜 '벽을 따라가는' 동작에 대한 증거를 제공합니다. 깊이가 장착된 에이전트는 충돌 횟수가 가장 적고 RGB 에이전트는 그 사이에 있습니다.

다.2. 시끄러운 깊이

노이즈 깊이 측정이 에이전트 성능에 미치는 영향을 조사하기 위해 간단한 노이즈 모델을 사용하여 생성된 노이즈 깊이에 대해 깊이 에이전트 (재훈련 없이)를 재평가했습니다. 반대 깊이(깊은 깊이 = 더 많은 노이즈). 우리는 Gibson-val에서 depth-RL 및 SLAM에 대해 0.13 및 0.02 SPL의 강화를 관찰합니다 (depth-RL은 여전히 SLAM을 능가함). [20]의 SLAM은 잡음에 매우 강한 ORB-SLAM2를 사용하는 반면 깊이 RL은 잡음 없이 학습되었습니다.

σ 를 0.1로 늘리면 깊이 RL은 0.12 SPL이 되는 반면 SLAM은 치명적인 오류를 겪습니다.

D. Gibson 데이터 세트 큐레이션

Gibson 3D 텍스처 메시 [30]의 전체 데이터 세트를 수동으로 선별하여 구멍이나 텍스처 품질 문제와 같은 중요한 재구성 아티팩트를 나타내지 않는 메시지를 선택했습니다. 우리가 피하려고 노력한 주요 문제는

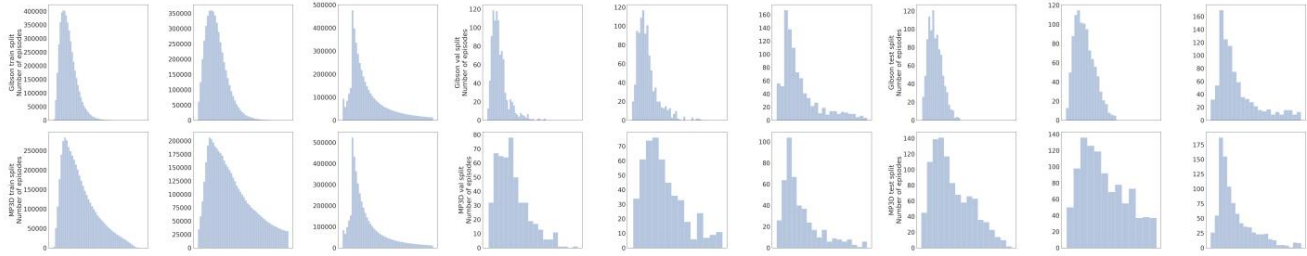


그림 9: PointGoal 탐색 에피소드의 통계. 왼쪽부터: 시작과 목표 사이의 유클리드 거리에 대한 분포, 시작과 목표 사이의 최단 경로를 따른 측지선 거리에 대한 분포, 측지선 대 유클리드 거리의 비율에 대한 분포.

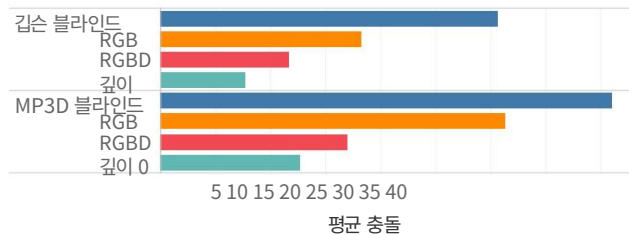


그림 10: Gibson 및 Matter port3D 데이터 세트에 대한 테스트 세트 에피소드에서 RL(PPO) 기본 에이전트의 다양한 감각 구성에 대한 성공적인 탐색 에피소드 동안의 평균 충돌 수. 블라인드 에이전트는 충돌 횟수가 가장 많은 반면 깊이 센서 (깊이 및 RGBD)를 보유한 에이전트는 충돌 횟수가 평균적으로 가장 적습니다.

바닥 표면의 구멍 또는 균열. 이는 겉보기에 연결된 탐색 가능한 영역을 횡단할 수 없는 분리된 구성 요소로 나누기 때문에 탐색 작업에 특히 문제가 됩니다. 우리는 수동으로 장면에 주석을 달고 (그림 11에 표시된 0에서 5 품질 척도 사용) 등급이 4 이상인 장면만 사용합니다.

E. 실험 결과 재현

실험 결과는 Habitat-API (commit [ec9557a](#))를 사용하여 재현할 수 있습니다. 및 Habitat-Sim (커밋 [d383c20](#)) 저장소. 실험을 실행하기 위한 코드는 폴더 아래에 있습니다.

서식지-api/habitat_baselines. 아래는 RL 실험에 사용한 셸 스크립트입니다.

```
# 참고: []의 매개변수는 실험에 따라 다릅니다.
# 참고: Gibson, MP3D에 대해 각각 8, 6 프로세스를 사용합니다.
```

```
파이썬habitat_baselines/train_ppo.py \
--센서 {RGB_SENSOR,DEPTH_SENSOR} \ --blind {0,1} --use-
gae --lr 2.5e-4 \ --clip-param 0.1 --use-linear-lr-decay \ --num-
프로세스 {8,6} --num-steps 128 \ --num-mini-batch 4 --num-updates
135000 \ --use-linear-clip-decay \
```

SLAM 실행에 대해서는 [Habitat api/habitat_baselines/slambased](#)를 참조하십시오.

F. 내비게이션 에피소드 예시

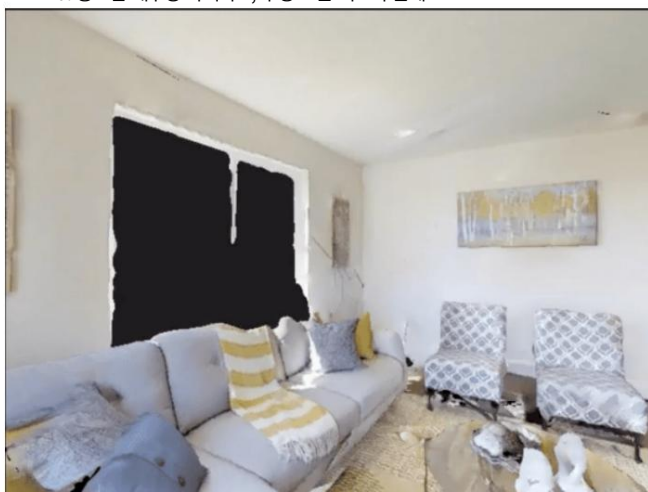
그림 12는 기본 문서에서 설명하는 RL(PPO) 에이전트의 다양한 감각 구성에 대한 추가 예제 탐색 에피소드를 시각화합니다. 블라인드 에이전트는 성능이 가장 낮고 환경과 훨씬 더 자주 충돌하며 탐색을 위해 '벽 포용' 전략을 채택합니다. RGB 에이전트는 충돌 가능성이 적지만 경우에 따라 목표 위치로 성공적으로 이동하는 데 여전히 어려움을 겪고 있습니다. 대조적으로 깊이가 장착된 에이전트는 훨씬 더 효율적이고 더 적은 충돌을 나타내며 더 성공적으로 목표를 탐색합니다 (전체적으로 더 높은 SPL 값으로 표시됨).



0: 중요한 재구성 아티팩트, 구멍 또는 텍스처 문제



1: 큰 구멍 또는 중요한 텍스처 문제 및 재구성 아티팩트



2: 구멍이 크거나 텍스처 문제가 크지만 재구성이 양호함



3: 작은 구멍, 약간의 텍스처 문제, 양호한 재구성



4: 구멍 없음, 약간의 텍스처 문제, 양호한 재구성



5: 구멍 없음, 균일한 질감, 양호한 재구성

그림 11: Gibson 데이터 세트에서 3D 텍스처 매쉬 재구성의 큐레이션에 사용되는 등급 척도. Habitat Challenge 데이터 세트에는 등급이 4 이상인 메시만 사용됩니다 .

집순



그림 12: RL(PPO) 에이전트의 다양한 감각 구성에 대한 추가 탐색 예제 에피소드, Gibson 및 MP3D val 세트의 시도를 시각화합니다. 파란색 점과 빨간색 점은 시작 및 목표 위치를 나타내고 파란색 화살표는 최종 에이전트 위치를 나타냅니다. 청록색 선은 에이전트의 궤적입니다. 허용되는 상단원 단계의 최대 수에 도달하면 색상이 파란색에서 빨간색으로 바뀝니다.

MP3D



그림 12: RL(PPO) 에이전트의 다양한 감각 구성에 대한 추가 탐색 억제 에피소드, Gibson 및 MP3D val 세트의 시도를 시각화합니다. 파란색 점과 빨간색 점은 시작 및 목표 위치를 나타내고 파란색 화살표는 최종 에이전트 위치를 나타냅니다. 청록색 선은 에이전트의 궤적입니다. 허용되는 상단원 단계의 최대 수에 도달하면 색상이 파란색에서 빨간색으로 바뀝니다.