

기술

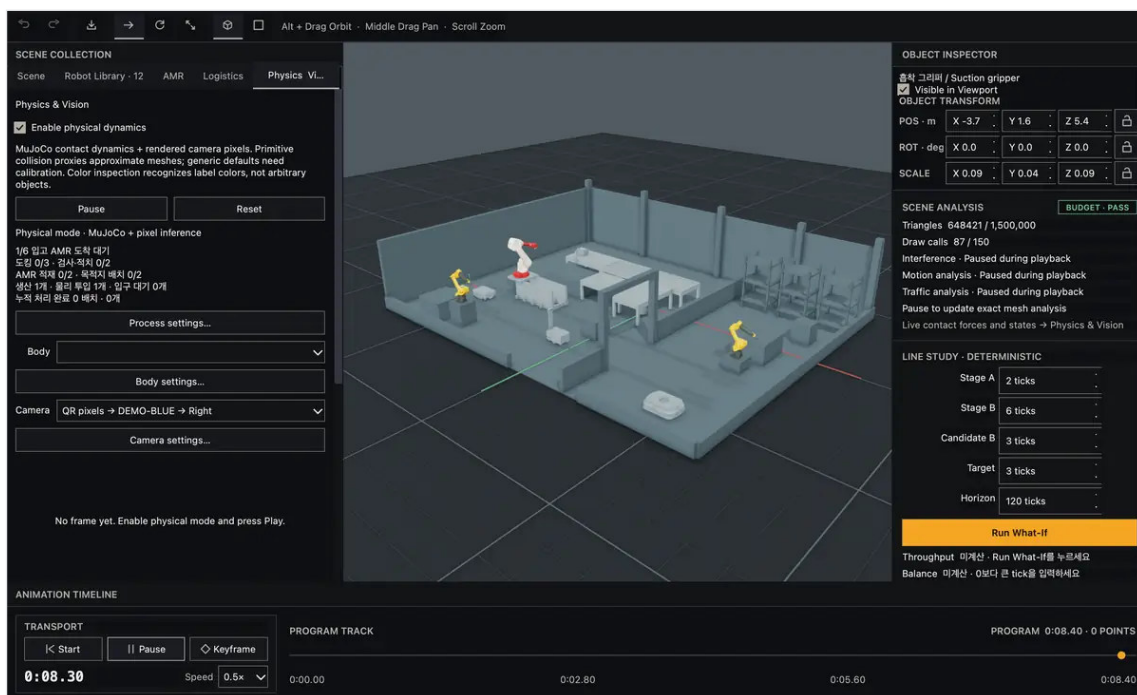
저희 공장 시뮬레이터가 2ms마다 무엇을 계산하는지 적었습니다

방동걸 (주)웨이스 대표이사

2026년 9월 7일

저희가 만드는 공장 시뮬레이터를 462초 돌렸습니다. 생산 지시가 26번 나갔고, 10개가 라인에 들어갔고, 4개는 로봇이 집어 AMR에 싣고 건너편 로봇이 내려놓는 데까지 갔습니다. 접촉을 9,243번 재는 동안 AMR이 벽이나 다른 AMR에 닿은 횟수는 0이었습니다. VEXPLOR로 공장 디지털트윈을 만들어 온 웨이스가 그 462초 동안 시뮬레이터가 무엇을 계산했는지 풀어 적었습니다.

이 숫자가 애니메이션에서 나온 값이 아니라는 것이 이 글의 전부입니다. 화면 안에서 도는 것은 미리 짜 둔 동선이 아니라 일곱 가지 계산이고, 아래에 그 일곱 가지를 하나씩 적었습니다.



공장 데모가 실제로 도는 화면. 로봇이 화물을 집어 AMR에 싣고, AMR은 자기 라이다로 길을 보며 나갑니다

1. 강체 접촉 — 2ms 고정 간격

물리는 MuJoCo 3.12.0으로 풉니다. 물체는 강체(모양이 변하지 않는 물체)로 두고 그 사이의 접촉과 마찰을 계산합니다. 핵심은 간격이 **2ms로 고정**이라는 점입니다. 화면이 몇 프레임 나오든 물리 시간은 같은 크기로만 전진합니다. 렌더링이 늦어지면 밀린 시간을 누적기에 담았다가 다음에 여러 걸음으로 나눠 밟고, 한 번에 최대 50걸음(0.1초)까지만 밟습니다. 그래서 같은 프로젝트를 두 번 돌리면 같은 자리에서 같은 일이 일어납니다.

물체마다 질량, 마찰, 반발, 무게중심을 따로 줍니다. 화물이 AMR 적재면에 얹히면 그 무게가 실제로 바퀴에 실리고, 마찰이 모자라면 미끄러집니다. 충돌 형상은 원본 메시지를 박스와 원통으로 단순화한 형상을 사용합니다. 삼각형 수십만 개를 그대로 충돌 계산에 넣지 않는 대신, 실제보다 약간 크게 잡아 보수적으로 판정합니다.

2. 차동 구동 — 모터 사양에서 속도가 나옵니다

AMR 속도를 "2 m/s"라고 적어 두고 그 속도로 밀지 않습니다. 모터 값에서 계산합니다.

최대 휠 속도 = (RPM / 감속비) x 2(파이)r / 60
구동력 = 2 x 모터 토크 x 감속비 x 효율 / 반경

차동 구동 모터 2개를 가정하고, 여기에 접지 마찰과 구름 저항, 차체 무게와 화물 무게를 얹어 가속·제동 한계를 냅니다. 그래서 짐을 많이 실으면 같은 지시에도 느리게 서고, 구동력이 모자라면 **아예 출발하지 않고 그 이유를 적어 보여 줍니다**. 화면에서 미끄러지듯 움직이는 대신 멈춰 서는 쪽을 골랐습니다.

바퀴 각속도만 지시하고 차체 자세를 직접 밀어 넣지 않습니다. 정지 지시는 "바퀴를 0으로"이지 "여기 붙어 있어라"가 아니어서, 접촉을 받으면 실제로 조금 밀립니다. 그게 맞다고 봅니다.

3. 관측과 경로 — 지도는 센서가 만듭니다

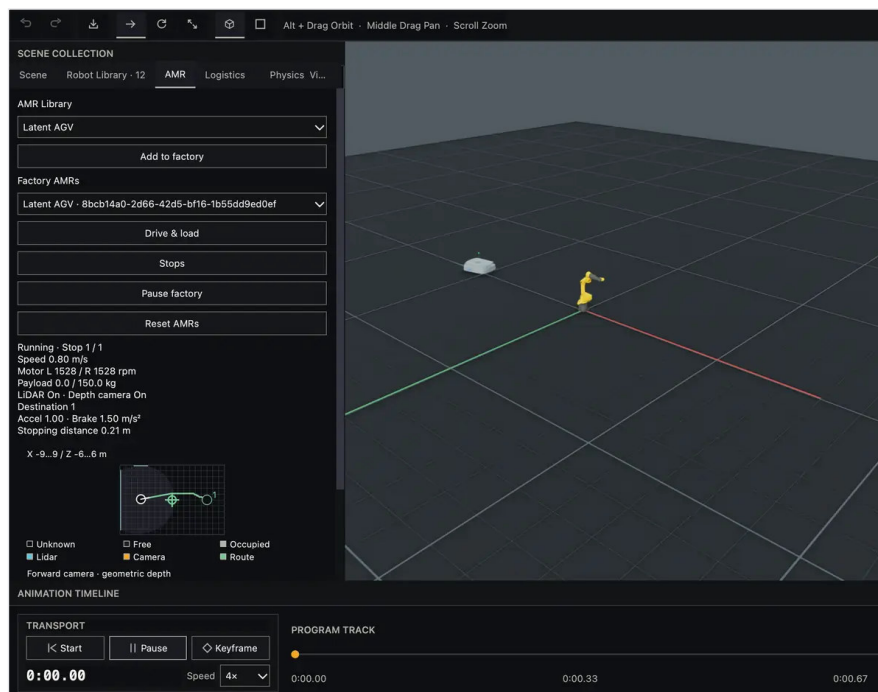
AMR은 공장 배치도를 미리 받지 않습니다. 자기 센서로 그때그때 지도를 만듭니다.

무엇	사양	무엇을 만드는가
라이다	360도, 사거리 6m	맞은 지점은 점유, 지나간 자리는 관측된 빈 곳으로 표시

무엇	사양	무엇을 만드는가
깊이 카메라	90도 시야	앞쪽 거리만. 켜고 끄면 볼 수 있는 범위가 실제로 달라집니다
점유 격자	20cm	아직 못 본 곳은 빈 곳과 구분해 따로 둡니다

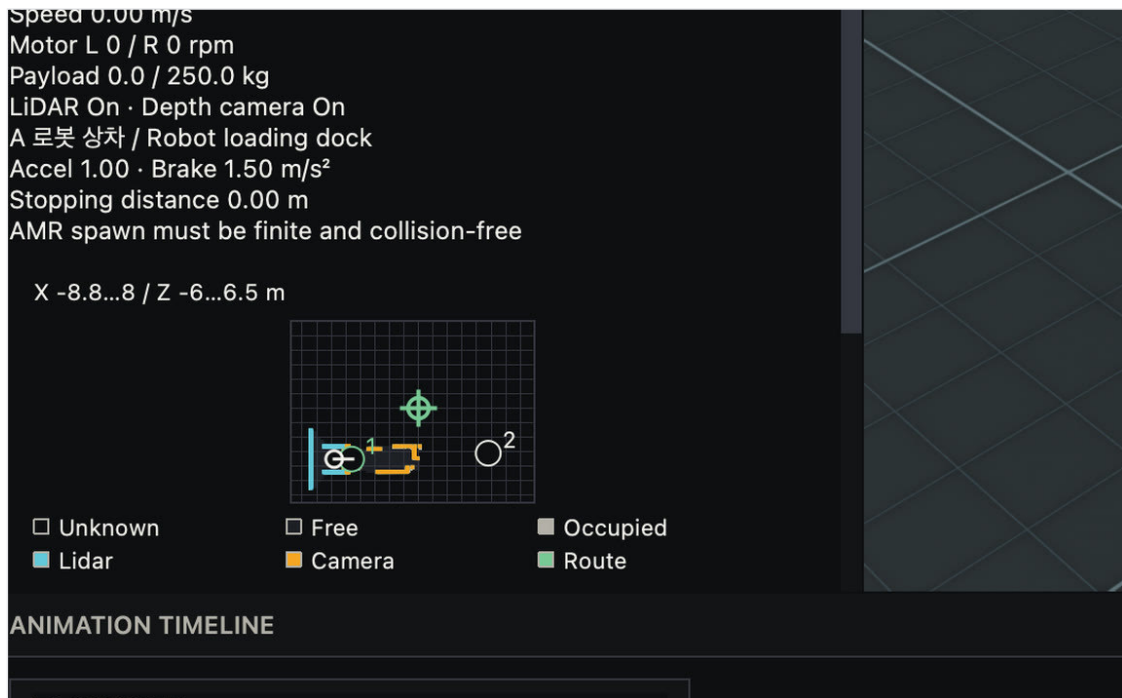
경로는 이 격자 위에서 A* 로 찾습니다. 남은 거리를 직선거리로 어림해 우선순위 큐를 돌리고, 두 지점을 잇는 선분이 **차체 외접원 반경만큼 부풀린 장애물에 닿는지**를 매번 검사합니다. 통로가 차체보다 좁으면 경로가 아예 안 나오고, 그러면 AMR은 억지로 가지 않고 섭니다.

주행 중에도 계속 다시 봅니다. 사람이 설비를 옮겨 놓으면 다음 관측에서 잡혀 경로가 새로 나옵니다. 센서를 모두 끄면 정지합니다.



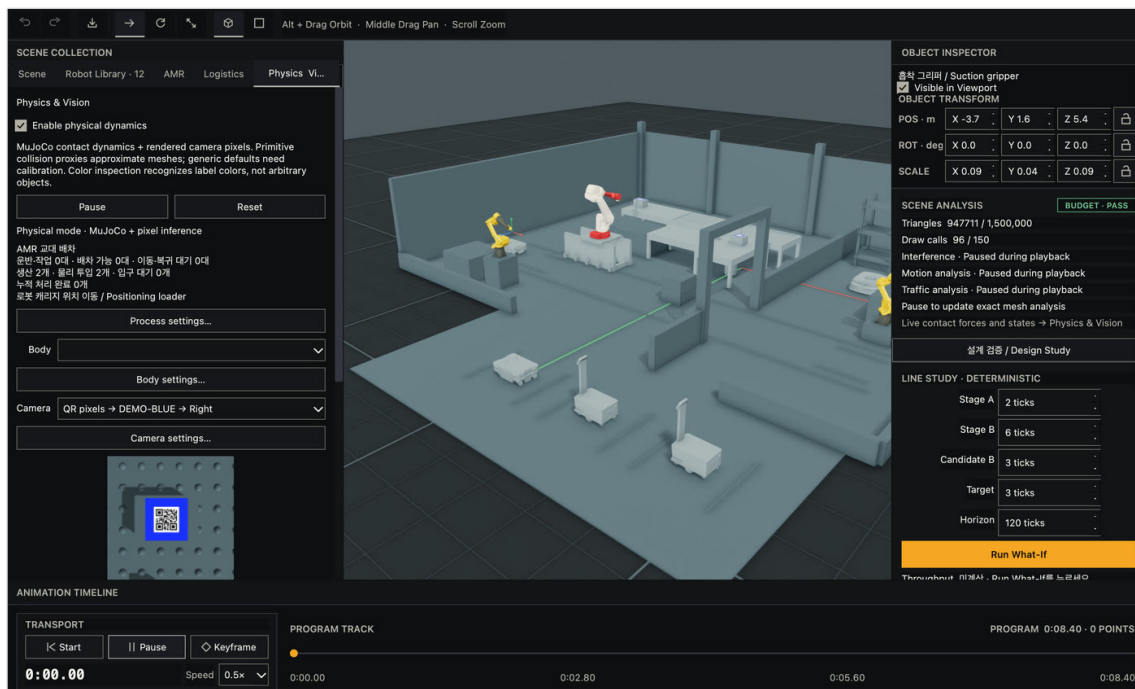
AMR 한 대에 목적지를 찍고 보낸 장면. 왼쪽 지도가 그 AMR이 관측한 만큼 채워집니다

왼쪽 아래가 그 AMR이 만든 지도입니다. 밝아지는 영역이 센서가 닿아 관측된 곳이고, 초록 선이 그때그때 다시 계산되는 경로, 원이 AMR과 목적지입니다. 아직 아무 센서도 닿지 않은 자리는 빈 곳과 구분해 따로 둡니다. 위 숫자는 같은 순간의 속도와 좌우 모터 회전수, 적재량, 가속·제동 한계, 정지거리입니다. 지도와 숫자가 같은 계산에서 나옵니다.



통합 공장 데모에서 같은 패널. 물리 모드에서는 구동 상태와 목적지가 이 자리에 뜹니다

4. 여러 대가 같이 다닐 때

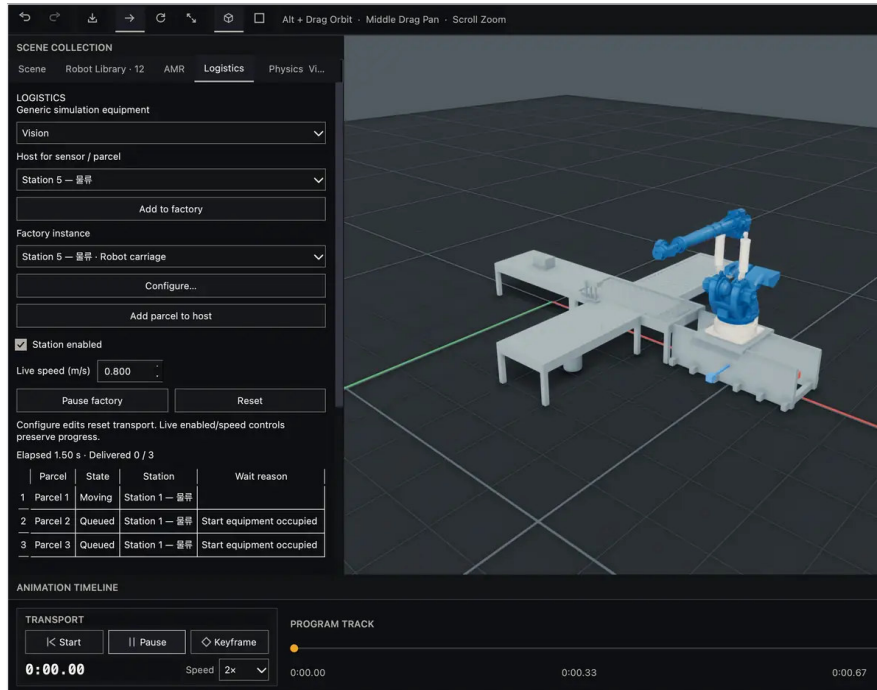


같은 통로를 세 대가 함께 쓰는 순간. 왼쪽 패널에 배치 상태가 같이 뜹니다

AMR이 여러 대면 서로가 서로의 장애물입니다. 관측된 동료는 순간 위치의 원이 아니라 **0.6초 동안 쓸고 갈 영역**으로 잡아 제동합니다. 좁은 구간은 먼저 잡은 차가 지나갈 때까지 뒤차가 기다리도록 통행 순서를

됩니다. 다만 모든 교차 상황의 교착이 저절로 풀린다고 말하지는 않습니다. 지금 기본으로 배포하는 공장은 3대 편성이고, 그 편성으로 462초를 끝까지 검증했습니다.

5. 물류 설비 — 읽은 값으로 갈림길을 고릅니다



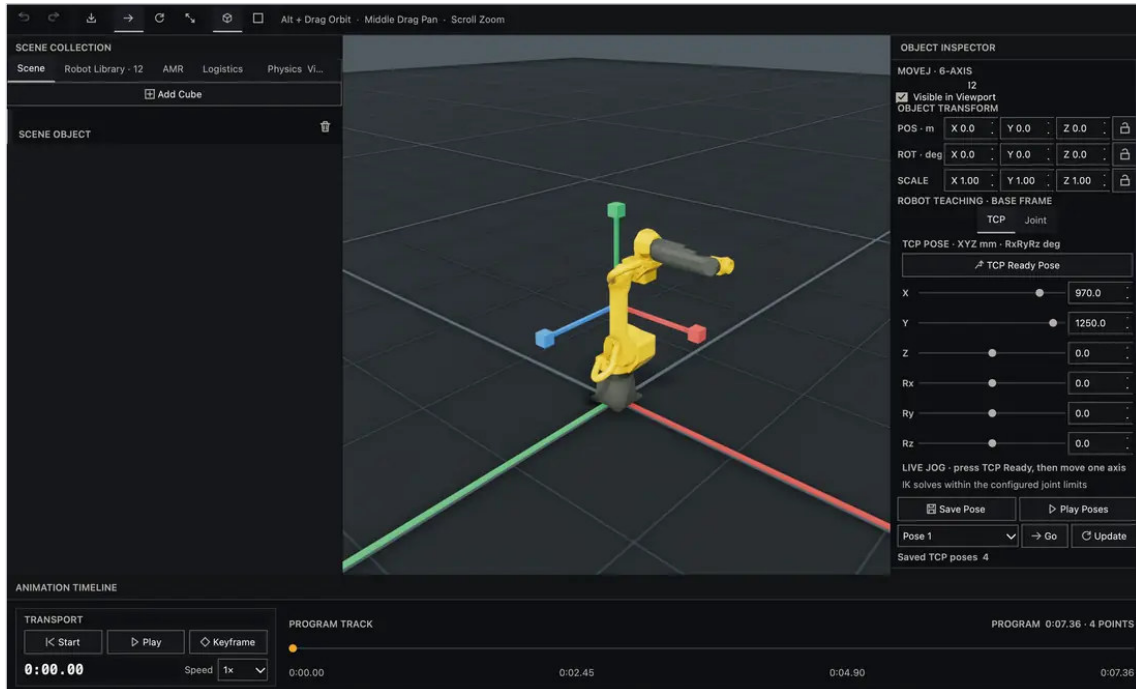
컨베이어와 회전대, 캐리지를 이어 붙이고 분류 규칙을 준 장면. 화물 3개가 각각 다른 출구로 나갑니다

벨트와 롤러, 볼 이송장치, 회전대, 캐리지를 이어 붙이고 분기 지점에 감지 센서와 바코드 리더, 비전 센서를 겁니다. 규칙은 위에서부터 정확히 비교해 첫 일치의 출구를 고르고, 아무것도 맞지 않으면 앞으로 보냅니다.

위 화면의 화물 3개는 각각 바코드가 LEFT, 비전 분류가 RIGHT, 나머지 하나는 어느 규칙에도 걸리지 않습니다. 15.75초 뒤 표에 남은 결과가 **Station 2 / Station 4 / Station 5** — 왼쪽, 오른쪽, 그리고 앞입니다. 규칙대로 갈렸습니다.

다음 설비가 멈춰 있거나 이미 화물을 물고 있으면 상류가 기다립니다. 표의 Wait reason 칸에 그 이유가 그대로 뜹니다. 허용 무게와 치수, 연결 지점의 높이·방향도 함께 검사해서, 안 맞으면 넘기지 않고 세웁니다.

6. 로봇 기구학 — 좌표를 주면 관절이 나옵니다



좌표로 자세를 만들고 저장한 뒤 타임라인에서 재생하는 화면

관절 여섯 개짜리 로봇에 자세를 가르치는 방법은 둘입니다. 관절 각도를 직접 돌리거나, 손끝 좌표(TCP)를 주는 것입니다. 뒤쪽이 역기구학입니다.

X·Y·Z와 Rx·Ry·Rz 여섯 값을 함께 푸는 수치 해법으로 풀립니다. 한 좌표만 밀면 나머지 다섯은 2mm·1도 안에서 그대로 있도록 매번 다시 풀고, 그 결과로 J1~J6가 전부 갱신됩니다. 관절 한계를 넘는 해는 채택하지 않고, 닿지 않는 자리를 지시하면 마지막 유효 자세를 유지합니다.

가르친 자세들 사이는 아래 곡선으로 잇습니다.

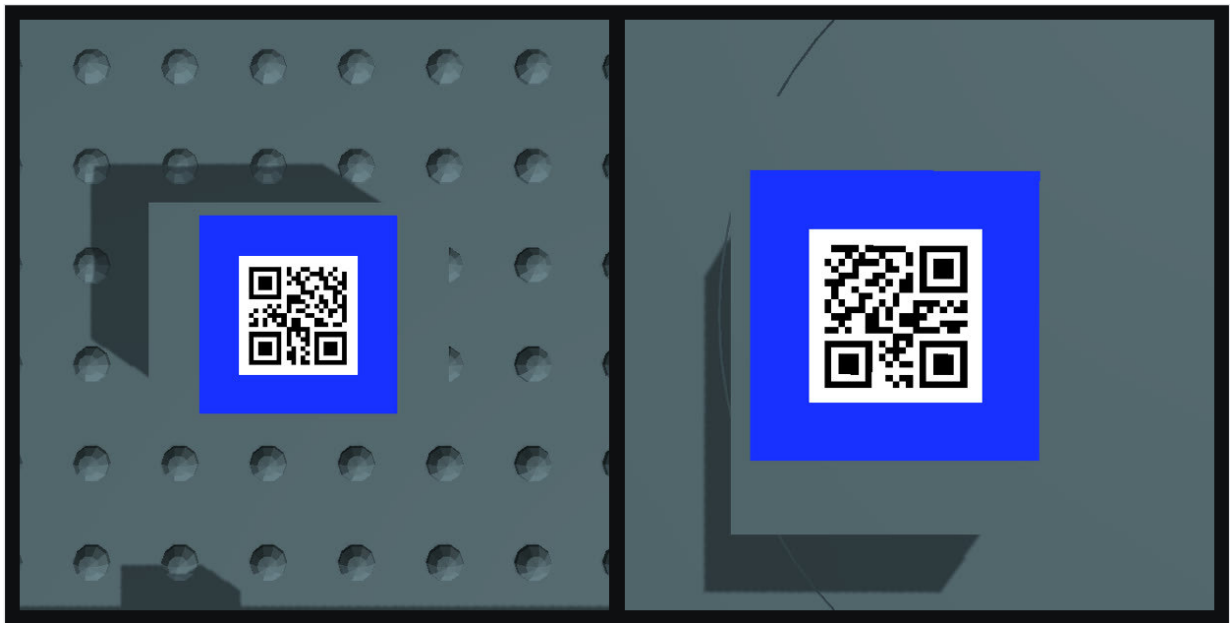
$$s(t) = 6t^5 - 15t^4 + 10t^3$$

시작과 끝에서 속도와 가속도가 0이 되는 곡선이라 관절이 튀지 않습니다. 여섯 관절이 동시에 출발해 동시에 도착하도록 시간을 맞추고, 기종별 속도 상한을 넘지 않게 전체 구간 시간을 늘립니다. 손끝을 실시간으로 미는 조그는 120mm/s, 45도/s로 묶어 두었습니다. 입력 한 번에 순간이동하지 않게 하려는 것입니다.

7. 픽셀 추론 — 렌더된 화면을 다시 읽습니다

공장 안 카메라는 "이 화물의 바코드는 A입니다" 같은 값을 데이터에서 꺼내 오지 않습니다. 그 순간 렌더된 실제 화면을 읽습니다. QR은 ZXing으로 해독하고, 색상은 ONNX 런타임에서 분류 모델을 돌립니다. 인식에 실패하면 그 화물은 통과하지 못하고 대기합니다.

그래서 카메라 높이나 화각을 잘못 잡으면 실제로 못 읽습니다. 번거롭지만, 현장에서 카메라를 어디에 달아야 하는지를 미리 확인하려면 이 방향이 맞습니다.



카메라가 실제로 읽은 프레임 두 장. 왼쪽이 바코드 리더, 오른쪽이 비전 카메라입니다

위 두 장은 시뮬레이션 안의 카메라가 그 순간 찍은 화면 그대로입니다. 합성한 그림이 아니라 렌더 결과를 640×640으로 읽어 온 것이고, 여기서 QR을 해독하고 색상을 분류합니다. 각도가 기울면 기울어진 채로 들어오고, 그래서 못 읽는 경우도 그대로 재현됩니다.

08

그리고 확정적 계산 하나

라인 검토는 물리와 따로 돌립니다. 단계별 작업 시간과 목표 사이클을 넣으면 처리량·병목·균형을 계산합니다. 난수를 쓰지 않아 같은 입력이면 언제나 같은 결과가 나오고, 화면에 뜨는 값과 보고서에 찍히는 값이 같은 계산에서 나옵니다.

09

계산하지 않는 것

- 위치 추정 오차를 넣지 않았습니다. 센서는 기하학적 거리만 재고 바닥은 평면으로 봅니다. SLAM 오차나 실제 장비 드라이버는 없습니다.
- 비전은 색상 라벨을 가리는 모델입니다. 임의의 물체를 찾아내지 않고, 화면 가운데 고정 영역만 봅니다.
- 충돌 형상은 단순화한 프록시입니다. 제조사가 검증한 값이 아닙니다.
- 로봇 전체 관절의 토크 동역학은 없습니다. 관절 프로그램은 재생이고, 힘으로 도는 것은 AMR 바퀴와 화물 쪽입니다.

시뮬레이션 결과로 설비를 결재하려면 이 목록이 먼저 있어야 한다고 봅니다. 무엇을 계산했는지보다 무엇을 계산하지 않았는지가 판단을 가릅니다.

10

462초가 남긴 숫자

항목	값	어떻게 썼는가
물리 시간	462초	2ms 고정 간격, 생산 지시는 18초마다 오차 50.001ms 안
생산·투입·대기	26 / 10 / 16	투입은 물리 버퍼와 입구 여유가 허락할 때만
전 구간을 건넌 화물	4개	로더 그리퍼에서 AMR 적재면으로, 다시 목적지 그리퍼로
AMR 복귀	3대 전부	순간이동이 아니라 실제로 주행해서
동시 이동 최대	3대	서로 양보하면서
경로 이탈 최대	0.79m	계획 경로와 실제 주행선의 최대 거리
AMR 접촉	0 / 9,243 샘플	장애물·다른 AMR 모두

11

왜 여기까지 넣나

3D가 값을 만드는 자리는 셋뿐이라고 봅니다. 아직 없는 것, 아직 오지 않은 시점, 그리고 공간 판정(통로 폭·회전 반경·충고·간섭)입니다. 이미 있는 것을 그대로 비추는 화면은 값이 없습니다.

셋 다 물리가 없으면 답이 안 나옵니다. 통로 폭이 되는지는 차체 외접원과 제동거리를 함께 봐야 알고, 그 제동거리는 짐 무게에서 나옵니다. 그림만으로는 "넓어 보인다"까지가 끝입니다.

지금 만드는 중이고, 파는 물건이 아닙니다. 다음에는 이 화면으로 실제 라인 하나를 세워 볼 생각입니다.

여러분 현장에서는, 도면으로는 되는데 물건을 들여놓고서야 안 되는 것이 보인 자리가 어디였습니까.

WACE(웨이스)는 공장이 스스로 판단하고 멈추지 않게 하는 제조 AI 운영체제 VEXPLOR를 만드는 회사다.



원문 blog.vexplor.com/post/what-our-factory-simulator-computes-every-2ms

발행 2026년 9월 7일 · 방동걸 (주)웨이스 대표이사



원문



홈페이지



링크드인

회사 주식회사 웨이스 (WACE Inc.) · 대표이사 방동걸 ·
사업자등록번호 439-87-02852

주소 (18462) 경기 화성시 동탄구 동탄영천로 150, 에이동(A동)
1702·1706호 (영천동, 실리콘엘리동탄)

연락처 전화 070-4730-0629 · contact@wace.me

홈페이지 wace.me · linkedin.com/company/wace-inc