기술

# 로봇이 부딪히는 경로, 왜 난수 없이 피하게 만들었나

**방동걸** (주)웨이스 대표이사 2026년 8월 26일

공장 라인을 컴퓨터 안에서 미리 돌려 보는 도구를 만들고 있습니다. 로봇 팔이 교시된 경로로 움직일 때 옆 설비와 부딪히는지 재는 기능을 먼저 만들었는데, 그 기능이 생기자마자 다음 요청이 따라왔습니다. 부딪힌다고 알려만 주지 말고, 피해 가는 경로도 만들어 달라는 요청입니다. 가상공장 소프트웨어를 VEXPLOR 안에서 만드는 웨이스가 그 요청을 받고 설계한 방식을 적었습니다.

이번 주에 그 기능을 만들었습니다. 간섭이 잡힌 구간에 경유점을 끼워 넣어 팔이 비켜 가게 하는 자동 제안 기능입니다. 만들면서 원칙을 4가지 정했고, 그 과정에서 예상하지 못한 함정도 하나 밟았습니다. 로봇 경로에만 해당하는 이야기가 아니라 검사 도구에 자동 수정을 붙일 때 누구나 겪는 문제라서, 무엇을 어떻게 정했는지 순서대로 적습니다.

01

## 판정하는 코드와 제안하는 코드는 같은 계산기를 써야 합니다

충돌 검사는 이미 있었습니다. 팔의 각 링크를 캡슐 모양으로 감싸고, 경로를 따라 움직이는 동안 이웃 설비의 외곽 상자와 얼마나 떨어져 있는지를 표본을 찍어 가며 재는 방식입니다. 회피 탐색을 새로 짜기 시작하면 이 검사를 다시 구현하고 싶은 유혹이 옵니다. 탐색은 후보 자세를 수백 개씩 시험해야 하니, 탐색 전용으로 더 가벼운 검사를 만들면 빨라질 것 같기 때문입니다.

그런데 검사가 두 벌이 되는 순간 이상한 일이 생깁니다. 판정은 부딪힌다고 하는데 제안된 경로는 검사를 통과하고, 반대로 판정은 통과라는데 제안 쪽 검사가 막아서기도 합니다. 두 코드가 조금씩 다른 근사를 쓰면 반드시 어딘가에서 서로 다른 말을 하게 됩니다. 사용자 입장에서는 같은 화면 안에서 도구가 자기모순을 일으키는 셈이고, 이 지점에서 도구 전체의 신뢰가 무너집니다.

그래서 첫 원칙을 이렇게 정했습니다. **판정이 쓰는 충돌 계산을 "자세 하나를 받아 여유 거리를 돌려주는 함수"로 추출하고, 판정과 탐색이 그 하나를 함께 쓴다.** 탐색이 조금 느려지는 대가는 있지만, 판정과 제안이 갈라질 가능성 자체가 사라집니다. 덤도 있습니다. 제안 경로에 붙는 여유 거리가 판정과 같은 잣대의 실측값이 됩니다.

반환값의 모양도 이 단계에서 정했습니다. 이 함수는 답을 3가지로 구별해서 돌려줍니다. 잴 수 없는 경우, 재봤지만 못 푸는 경우, 그리고 푼 경우입니다. 관절이 없는 설비이거나 이웃이 아예 없어서 간섭이라는 개념 자체가 성립하지 않으면 "잴 수 없음"이고, 이것은 실패가 아닙니다. 간섭이 없어서 바꿀 것이 없으면 원래 교시점을 그대로 돌려주되 삽입한 경유점이 0개라고 표시해서, 호출한 쪽이 "제안할 것이 없다"와 "제안이 있다"를 구별할 수 있게 했습니다. 이런 구별이 사소해 보여도, 화면에서 "간섭 없음"과 "검사 불가"가 같은 표시로 뭉개지면 사용자는 검사가 돌았는지조차 알 수 없게 됩니다.

02

## 난수를 빼니 잃은 것보다 얻은 것이 많았습니다

경로 계획의 교과서 답은 표본식 탐색입니다. RRT 계열처럼 공간에 무작위 표본을 뿌려 가며 길을 넓혀 가는 방식이고, 좁고 복잡한 공간에서 길을 잘 찾습니다. 대신 난수가 들어가는 순간 같은 입력에서 실행할 때마다 다른 답이 나올 수 있습니다.

저희 문제는 교과서 문제보다 훨씬 좁았습니다. 아무것도 없는 공간에서 길을 처음 찾는 것이 아니라, 이미 사람이 잡아 둔 교시 경로가 있고 그중 간섭이 걸린 구간만 살짝 비켜 가면 됩니다. 그래서 무작위 표본 대신 정해진 격자를 골랐습니다. 간섭 구간의 중간 자세를 기준으로 몸통 회전과 어깨, 팔꿈치 관절을 15도에서 60도까지 정해진 조합으로 움직인 후보를 만들고, 원래 자세에서 바꾼 양이 작은 순서로 줄을 세워 차례로 시험합니다. 경유점 하나로 안 풀리면 구간의 3분의 1, 3분의 2 지점에 두 개를 넣어 팔을 든 채 지나가는 모양을 시도하고, 그래도 안 되면 못 푸는 것으로 확정합니다.

두 방식의 차이를 정리하면 이렇습니다.

| 구분 | 표본식 탐색 | 격자 탐색(이번 선택) |
|---|---|---|
| **같은 입력의 답** | 달라질 수 있음 | 항상 같음 |
| **찾는 범위** | 먼 우회까지 | 기존 경로 부근만 |
| **자동 테스트** | 재현이 어려움 | 회귀 테스트 가능 |
| **제안의 근거** | 설명이 어려움 | 개입이 작은 순서 |

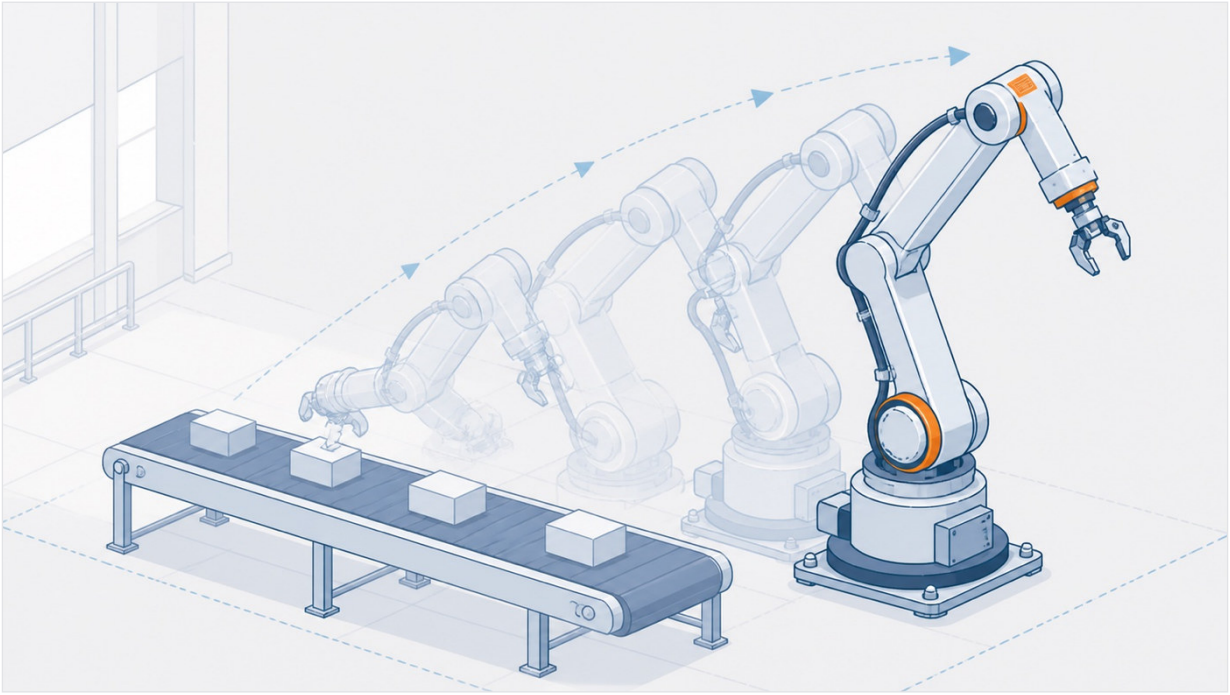
난수를 빼고 얻은 것은 3가지입니다.

- **같은 입력이면 같은 제안** — 어제 나온 제안이 오늘 달라지면, 사용자는 도구가 아니라 운을 상대하게 됩니다.
- **회귀 테스트에 넣을 수 있다** — 이 배치에서는 이 경유점이 나와야 한다는 기대를 테스트로 박아 둘 수 있습니다. 실행마다 답이 흔들리면 이 테스트 자체가 성립하지 않습니다.

- **제안을 말로 설명할 수 있다** — 개입이 가장 작은 후보부터 시험하므로, 채택된 답은 언제나 "이보다 덜 움직여서는 안 풀렸다"는 뜻을 함께 담고 있습니다.

격자에 관해 정한 세부 규칙이 둘 더 있습니다. 하나는 손목입니다. 후보를 만들 때 몸통과 어깨, 팔꿈치까지만 움직이고 손목 관절은 건드리지 않습니다. 경유점은 팔을 비켜 가게 하는 장치이지 작업 자세를 바꾸는 장치가 아니라서, 손목까지 흔들면 회피가 아니라 다른 작업이 되기 때문입니다. 다른 하나는 가동범위입니다. 로봇에 관절 한계가 등록돼 있으면 격자가 만든 후보를 그 범위 안으로 잘라 냅니다. 수학으로는 풀리는데 그 로봇이 물리적으로 취할 수 없는 자세를 제안하면, 그 제안은 화면에서만 성립하는 답입니다.

물론 포기한 것도 있습니다. 격자는 기존 경로 부근만 뒤지기 때문에 미로처럼 얽힌 공간을 크게 우회하는 답은 못 찾습니다. 그런데 로봇 셀이 그 지경이라면 그것은 경로가 아니라 배치를 고칠 문제라는 것이 저희 판단이었습니다.



경유점 후보를 개입이 작은 순서로 시험하는 격자 탐색의 진행 — 작은 들기부터 차례로 시도한다

03

## "못 찾았습니다" 한마디가 가장 나쁜 답이었습니다

이 기능에서 가장 공들인 부분은 경로를 찾는 쪽이 아니라 못 찾았을 때입니다. 회피 제안이 못 푸는 경우는 3가지인데, 세 경우의 처방이 전부 다릅니다.

- **교시점 자세 자체가 간섭인 경우** — 경유점은 점과 점 사이의 길만 바꾸는 장치입니다. 출발점이나

도착점이 이미 이웃 설비를 파고들고 있으면 사이를 아무리 바꿔도 소용이 없습니다. 이때는 그 교시점을 고치거나 배치를 옮겨야 합니다.

- **작업 구간의 간섭** — 용접선을 따라가는 직선 작업 구간이 간섭이면, 그 선을 구부려 피할 수는 없습니다. 경로 문제가 아니라 배치와 공정의 문제이므로, 제안하지 않고 그렇게 말합니다.
- **탐색이 실제로 실패한 경우** — 최대 60도 들기까지 시험해도 답이 없으면 공간이 정말 좁은 것입니다. 배치를 옮기거나 교시를 다시 잡아야 합니다.

빠뜨리기 쉬운 구간이 하나 더 있어서 적어 둡니다. 교시 경로는 마지막 점에서 첫 점으로 돌아오는 순환이므로, 되돌아오는 구간도 같은 규칙으로 검사하고 같은 규칙으로 고칩니다. 앞으로 가는 구간만 검사하면 팔이 일을 마치고 복귀하는 길에서 부딪히는 경우를 놓치는데, 실제 셀에서 사고가 나는 자리는 오히려 이런 복귀 구간인 경우가 많습니다.

이 구별이 왜 중요한지 개발 중에 바로 실감했습니다. 데모 배치에서 여유가 마이너스 32mm인 간섭이 잡혔고, 처음에는 탐색이 풀어 줄 문제라고 생각했습니다. 그런데 거부 사유를 구별해 보니 특정 교시점의 자세 자체가 이웃 설비와 겹쳐 있었습니다. 경유점을 아무리 넣어도 풀리지 않는 문제였고, 시스템이 몇 번 교시점이 문제이니 그 점을 고치거나 배치를 옮기라고 말해 준 덕분에 로봇 배치 자체를 옮겨 여유를 플러스 622mm까지 벌렸습니다. 만약 도구가 "회피 경로를 찾지 못했습니다"라고만 했다면, 탐색 범위를 늘리는 엉뚱한 방향으로 시간을 썼을 겁니다.

자동화가 무언가를 거부할 때는, 거부의 이유를 처방이 갈리는 단위로 구별해서 말해야 합니다. 못 한다는 사실보다 왜 못 하는지가 사용자의 다음 행동을 정하기 때문입니다.

04

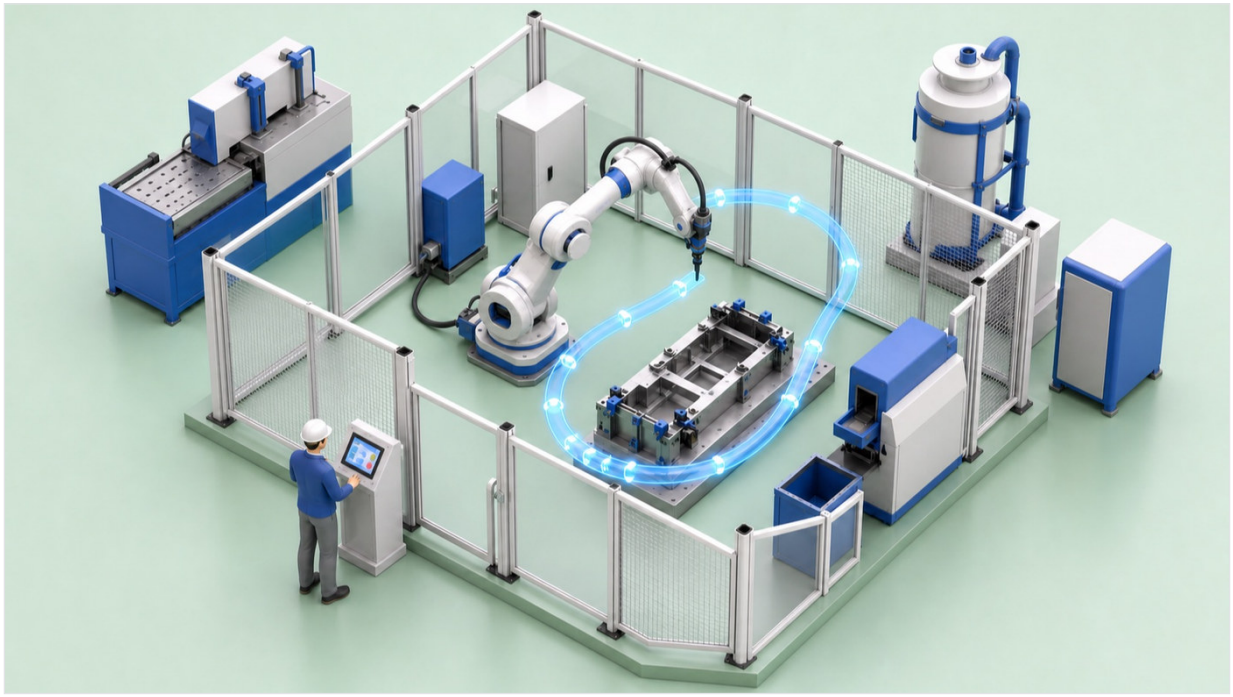
## 찾아낸 경로도 한 번 더 재고, 뒤집히면 제안하지 않습니다

격자 탐색은 빠르게 돌기 위해 구간 하나를 24개 표본으로 근사해서 검사합니다. 판정이 궤적 전체를 90개 표본으로 재는 것보다 성긴 검사인데, 후보 수백 개에 90개짜리 검사를 다 돌리면 탐색이 감당하기 어렵게 느려집니다. 대신 후보가 채택되면 마지막에 한 단계가 더 있습니다. 경유점이 들어간 제안 경로 전체를 실제 궤적으로 다시 계산해서, 직선 보간 구간까지 포함해 판정과 같은 밀도로 여유를 재는 재검입니다. 성긴 검사에서 통과했던 경로가 이 실측에서 간섭으로 뒤집히면, 그 제안은 버립니다.

찾느라 들인 계산이 아깝다고 내보내는 순간, 검사 도구가 스스로 만든 값을 참이라고 말하는 도구가 되어 버립니다. 그래서 제안에는 실측으로 잰 최소 여유가 함께 실리고, 이 값이 어떤 가정 위에서 계산됐는지를 적은 문장도 그대로 붙어 나갑니다. 탐색에 쓴 여유 목표 30mm는 규격이나 법규가 아니라 저희가 정한 설계값이라서, 그 사실도 가정 문장에 들어 있습니다.

마지막 원칙은 적용입니다. 제안된 경로는 사람이 확인하고 적용 버튼을 눌러야 교시에 반영됩니다. 계산

결과가 사람 확인 없이 원본 데이터를 바꾸기 시작하면, 검사 도구와 자동 조작 도구의 경계가 사라지고 사용자는 화면에 보이는 경로가 자기가 잡은 것인지 도구가 바꾼 것인지 알 수 없게 됩니다.



제안 경로는 실제 궤적 재검을 통과해야 하고, 반영은 사람의 적용으로만 이루어진다

05

## 정리하면, 계산은 자동으로 하되 결정은 사람에게 남겼습니다

이번에 정한 원칙 4가지를 다시 정리하면 이렇습니다.

- **판정과 제안은 같은 충돌 계산기를 쓴다** — 두 벌이면 반드시 서로 다른 말을 한다.
- **탐색에서 난수를 뺀다** — 재현과 회귀 테스트, 설명 가능성이 무작위의 탐색력보다 값어치가 컸다.
- **못 풀면 이유를 구별해서 말한다** — 교시점 문제, 공정 문제, 공간 문제는 처방이 전부 다르다.
- **실측 재검에서 뒤집히면 제안하지 않고, 반영은 사람이 한다** — 도구가 값을 만들기 시작하면 신뢰가 끝난다.

덧붙여 둘 것이 하나 있습니다. 이 모든 계산은 시뮬레이션 안의 예측이고, 실제 로봇 컨트롤러의 검증과 현장 시운전을 대체하지 않습니다. 이 한계 문장 역시 화면에 그대로 표시합니다. 도구가 스스로의 한계를 말하게 만드는 것까지가 이번 기능의 범위였습니다.

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

**원문** blog.vexplor.com/post/collision-avoidance-without-randomness
**발행** 2026년 8월 26일 · 방동걸 (주)웨이스 대표이사

**회사** 주식회사 웨이스 (WACE Inc.) · 대표이사 방동걸 · 사업자등록번호 439-87-02852
**주소** (18462) 경기 화성시 동탄구 동탄영천로 150, 에이동(A동) 1702-1708호 (영천동, 실리콘앨리동탄)
**연락처** 전화 070-4730-0629 · contact@wace.me
**홈페이지** wace.me · linkedin.com/company/wace-inc

원문 홈페이지 링크드인