POLITECHNIKA RZESZOWSKA
im. IGNACEGO ŁUKASIEWICZA

KATEDRA INFORMATYKI I AUTOMATYKI
POLITECHNIKI RZESZOWSKIEJ

Śledzenie i rozumienie sceny

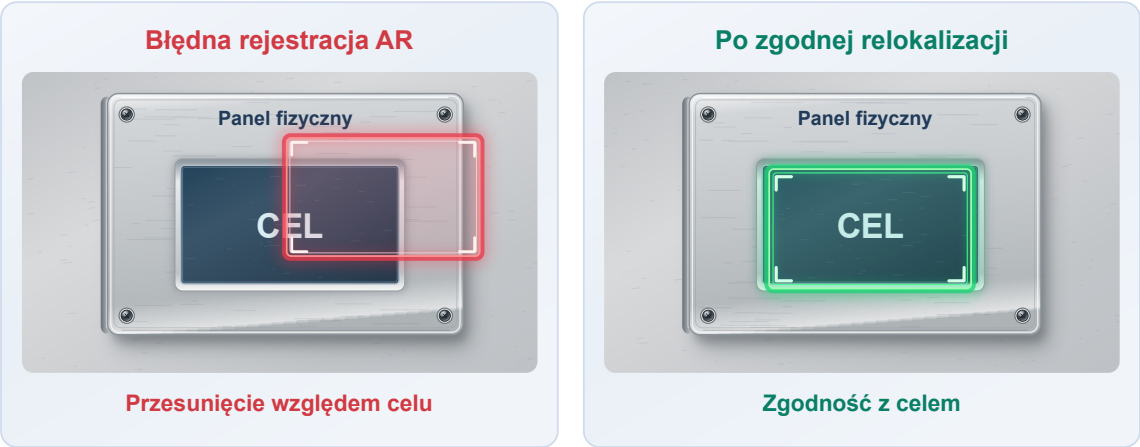
WiRR · Wykład 04 · Automatyka i Robotyka · 3 ECTS · 2026/2027 · v1.2.3

dr inż. Mateusz Pomianek

Katedra Informatyki i Automatyki, Politechnika Rzeszowska

27.10.2026

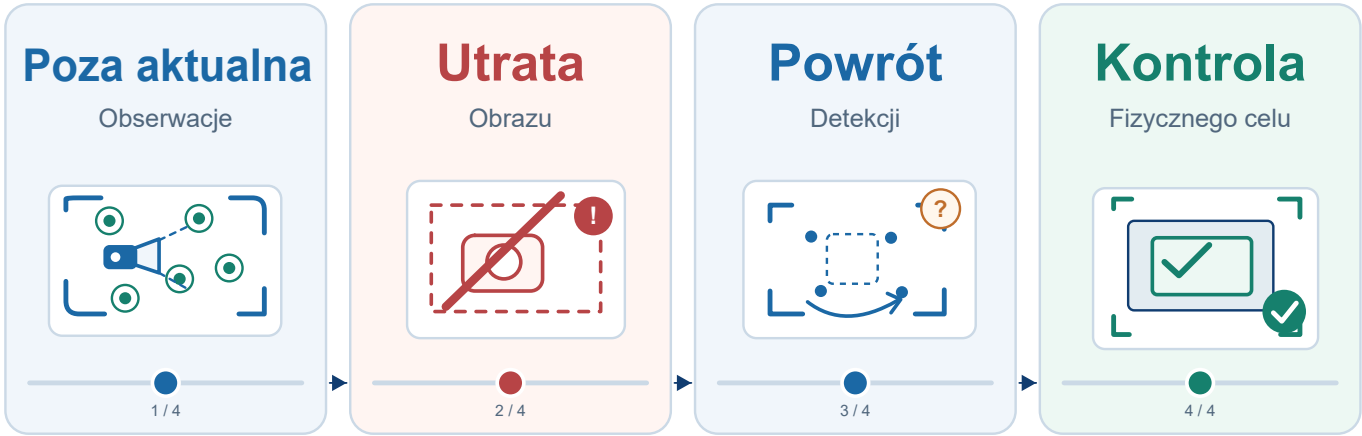
Utrata śledzenia może pozostawić nakładkę w złym miejscu



Jeden panel, dwa wyniki: nakładka może być płynna na ekranie, a równocześnie przesunięta o 35 cm względem obiektu

Unity, symulacja: czerwony obrys pokazuje niedopuszczoną pozycję, zielony oznacza pozycję po weryfikacji. Uruchamialny przykład C#.

Problem: obserwacja, utrata, powrót i weryfikacja



Odzyskanie obrazu nie oznacza jeszcze potwierdzenia poprawności kotwicy

Przebieg pokazu Unity: A: poprawna kotwica → B: brak obserwacji → C: ponowny `TRACKING`, ale bez mapy → D: błędne dopasowanie → E: test fizycznej referencji.

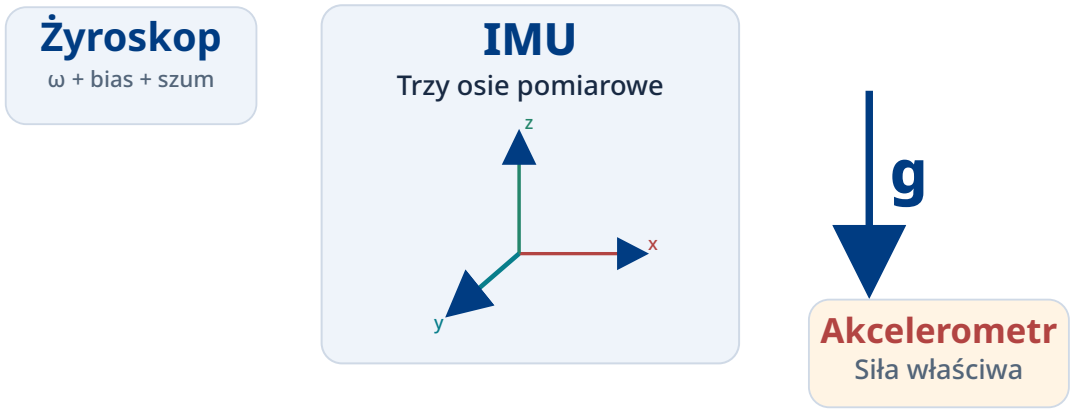
Skąd system otrzymuje pozę i odniesienie przestrzenne?



Aktualność pozy i zgodność z mapą sprawdzamy oddzielnie

Estymacja bieżącej pozy, powiązanie z mapą oraz poprawność kotwicy to **trzy odrębne zagadnienia**

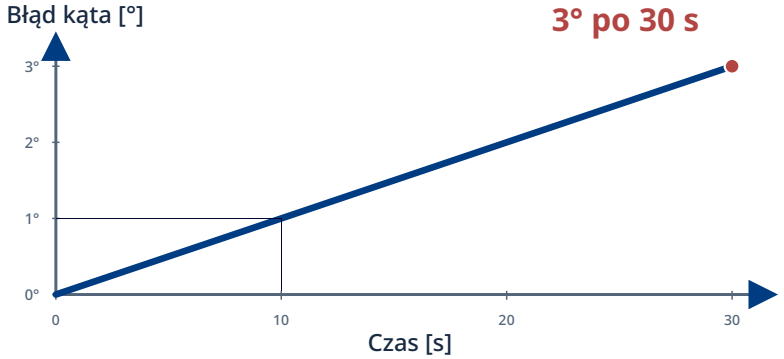
Co mierzy inercyjna jednostka pomiarowa?



$$\boldsymbol{\omega}_m = \boldsymbol{\omega} + \mathbf{b}_g + \mathbf{n}_g, \qquad \mathbf{f}_m = R^T(\mathbf{a} - \mathbf{g}) + \mathbf{b}_a + \mathbf{n}_a.$$

Żyroskop mierzy prędkość kątową, a akcelerometr **siłę właściwą**. Samo całkowanie nie zapewnia trwałej dokładności.

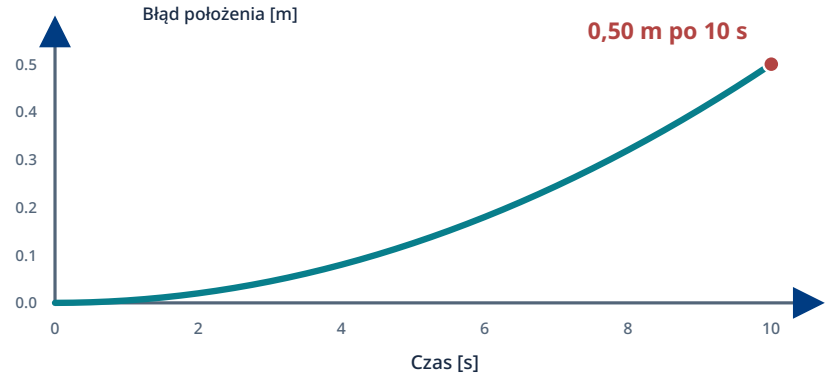
Stały bias żyroskopu powoduje narastanie błędu kąta



$$\Delta\theta(t) \approx b_g t \qquad 0,1^\circ/\mathrm{s} \cdot 30\,\mathrm{s} = \boxed{3^\circ}$$

Unity, tryb IMU: suwak czasu pokazuje obrót czerwonej estymaty względem szarej referencji; `angleErrorDeg = gyroBiasDegPerSec * imuTimeSeconds`. Kod pokazu.

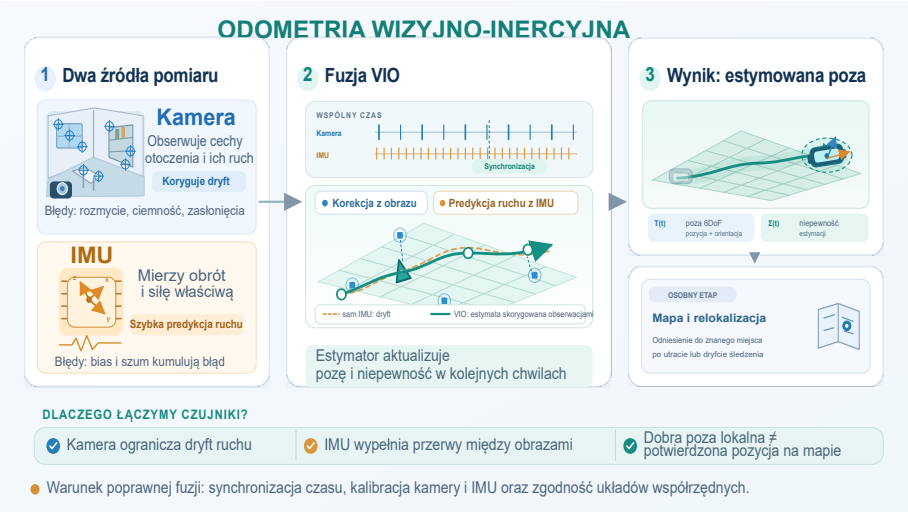
Bias akcelerometru daje błąd położenia rosnący z kwadratem czasu



$$\Delta p(t) \approx \frac{1}{2} b_a t^2. \qquad b_a = 0,01\,\mathrm{m/s^2}, t = 10\,\mathrm{s} \Rightarrow \boxed{0,50\,\mathrm{m}}\ .$$

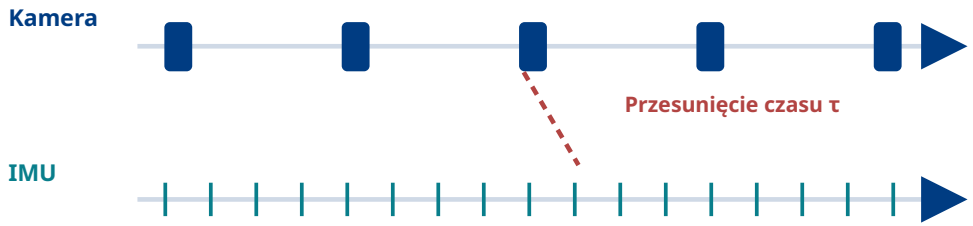
Unity, tryb IMU: ten sam czas, inny mechanizm: `posErrorM = 0.5f * accelBiasMps2 * imuTimeSeconds * imuTimeSeconds`. Kod pokazu.

Kamera i IMU uzupełniają swoje ograniczenia



- IMU daje szybkie informacje o zmianach ruchu, ale integruje błędy
- Kamera dostarcza obserwacji cech otoczenia, zależnych od tekstury i oświetlenia

Synchronizacja kamery i IMU jest częścią kalibracji



Porównujemy obserwacje tej samej chwili i tej samej ramy

$$\Delta\theta \approx \omega\tau, \qquad 100^\circ/\mathrm{s} \cdot 0{,}020\,\mathrm{s} = \boxed{2^\circ}\,.$$

Takie samo przesunięcie czasu jest bardziej dotkliwe przy szybszym ruchu. Przed integracją potrzebna jest kalibracja czasowa i przestrzenna.

Meta Quest 3: obraz świata przez kamery passthrough



W trybie *passthrough* obraz otoczenia przechodzi przez kamery, przetwarzanie i wyświetlacze. **Nie jest to optyczna przezierność.**

VITURE Luma Ultra: optyczna przezierność

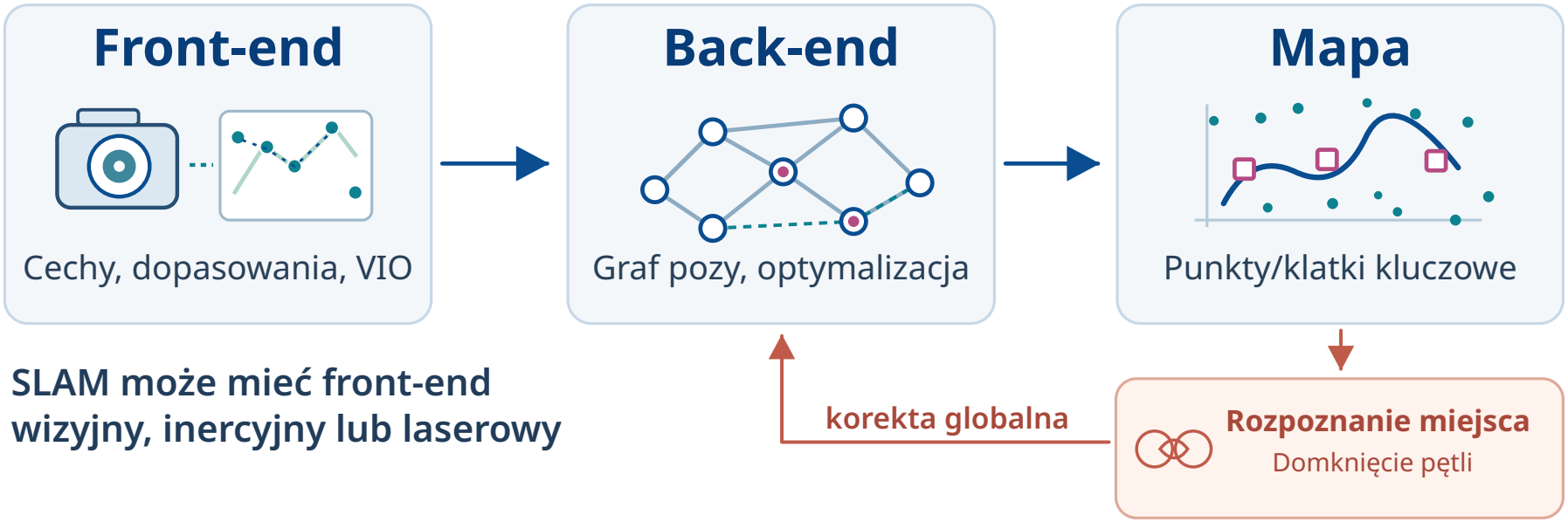


W torze przeziernym użytkownik ogląda otoczenie **bezpośrednio przez optykę**.

Dwie architektury obrazu, inne ograniczenia rejestracji

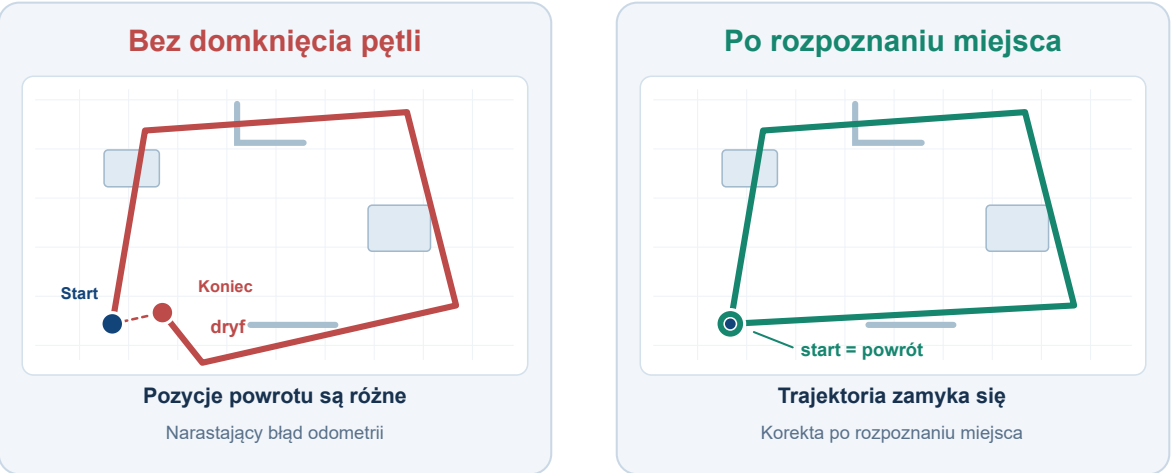
Kryterium	Meta Quest 3	VITURE Luma Ultra
Widok otoczenia	Tor wizyjny *passthrough*	Tor optyczny przezierny
Nałożenie treści	Kompozycja obrazu na wyświetlaczu	Dodanie obrazu wirtualnego do widoku
Okluzja i pomiar	Zależne od dostępnej głębi, śledzenia i aplikacji	Zależne od optyki oraz rzeczywistych możliwości zestawu
Weryfikacja	Obraz i cel fizyczny w tym samym czasie	Rejestracja względem widocznego celu

VIO wyznacza ruch; SLAM estymuje ruch i mapę



VIO może dostarczać krótkoterminową odometrię. SLAM może wykorzystywać obserwacje do budowy mapy, rozpoznawania miejsc i globalnej korekty. **SLAM nie wymaga obecności IMU w każdej implementacji.**

Domknięcie pętli może skorygować mapę i trajektorię



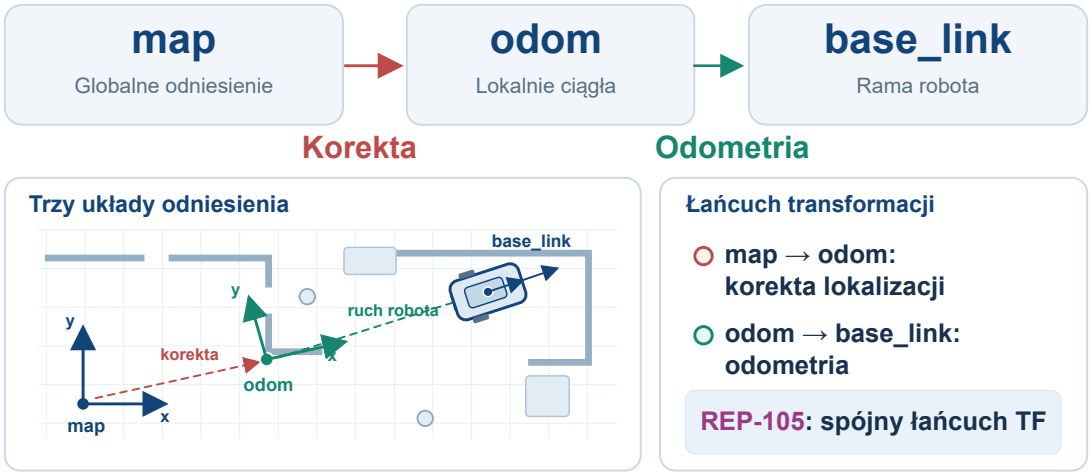
Mechanizm: ponowne rozpoznanie znanego miejsca pozwala skorygować dryf trajektorii i mapy. Fałszywe dopasowanie może zwiększyć błąd.

Trzy różne operacje lokalizacji



- **Domknięcie pętli:** ponowne rozpoznanie miejsca podczas mapowania.
- **Relokalizacja:** odzyskanie pozy względem istniejącej mapy po utracie odniesienia.
- **Lokalizacja globalna:** wyznaczenie położenia bez znanego przybliżenia początkowego.

Ramy `map`, `odom` i `base_link` pełnią inne funkcje



Według REP-105: **`odom`** jest lokalnie ciągła, lecz może dryfować; **`map`** zapewnia długoterminowe odniesienie, lecz korekty mogą powodować skoki jej relacji do `odom`.

5 minut przerwy

Pytania do samodzielnego powtórzenia

1. Jaki błąd kąta powstaje po 30 s przy stałym biasie żyroskopu $0,1^\circ/\mathrm{s}$ bez korekcji?
2. Jak zmieni się głębia stereo $Z = fB/d$, gdy dysparycja zmniejszy się dwukrotnie przy stałych f i B?
3. Czym różnią się domknięcie pętli, relokalizacja i ciągła odometria?
4. Dlaczego poprawny stan śledzenia i wiarygodna głębia nie gwarantują poprawnego położenia kotwicy AR?

Frameworki AR w Unity: sześć różnych ról

Rozwiązanie	Główna rola	Zastosowania / cechy
AR Foundation	Warstwa wspólna Unity	Jedno API dla różnych platform wywołania do providerów Android / iOS
ARCore **ARKit**	Dostawcy natywni	Tracking, płaszczyzny i kotwice Android / iOS, zależnie od sprzętu
Vuforia **VisionLib**	Rozpoznawanie obiektów	Targety obrazów, modele i CAD identyfikacja obiektów i otoczenia
MRTK **Snapdragon Spaces**	Interakcje w XR	Dłonie, wzrok, kotwice, UI okulary AR i mixed reality
Lightship ARDK **Zapworks**	Mapowanie / WebAR	Lightship: lokalizacja przestrzenna Zapworks: aplikacje AR w przeglądarce
Unity MARS	Authoring w edytorze	Projektowanie reguł i scen AR narzędzie autorskie, nie tracker

Nie są zamienne 1:1: różnią się poziomem integracji, funkcjami i urządzeniami.

Wybór SDK zależy od zadania: dostawca śledzenia, rozpoznawanie obiektów, interakcja XR i narzędzia edytora to różne warstwy.

Źródło zestawienia: Reko3D (2023). Zakres i zgodność wersji SDK wymagają osobnej weryfikacji.

Utrata i odzyskanie śledzenia to różne stany



Unity: warunek dopuszczenia nakładki (fragment demonstratora):

```
private bool IsOverlayApproved =>
    PoseFresh && mapMatched && referenceConfirmed;
```

Ostatnia poza staje się **nieaktualna** po utracie obserwacji. `TRACKING` ≠ poprawna rejestracja do poprzedniej mapy.

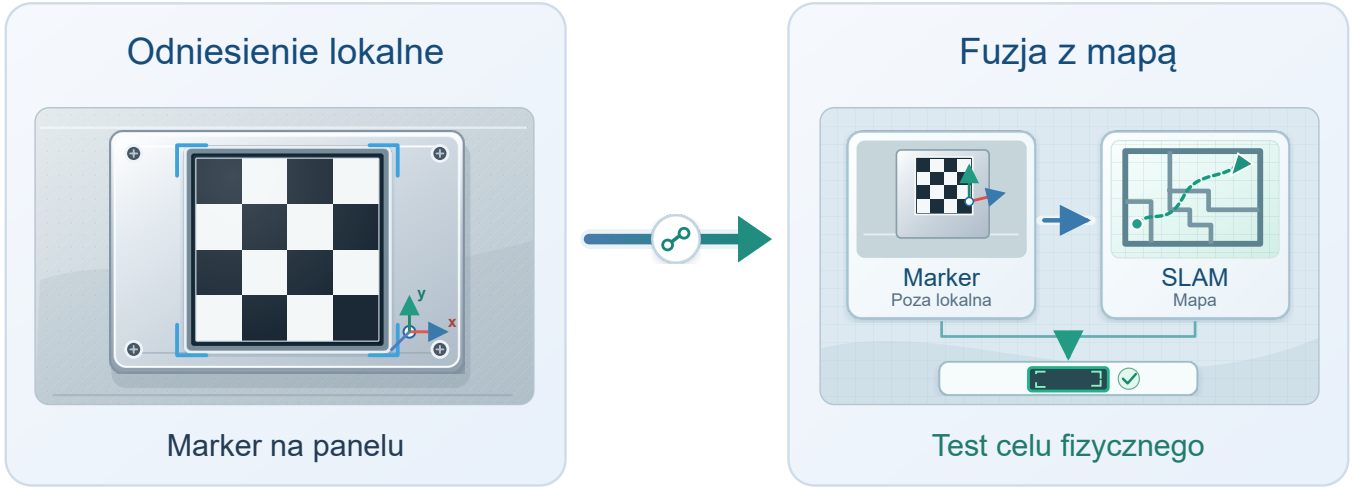
Kotwica AR ma pozę i zależy od jakości mapy



Unity: kontrolowana symulacja błędnej pozy kotwicy

```
diagnosticOverlay.transform.position =
    physicalTarget.transform.position + new Vector3(errorMeters, 0f, -0.14f);
```

Marker zapewnia lokalne odniesienie przy stanowisku



AprilTag lub ArUco może pomóc w odzyskaniu pozy względem znanego elementu.

Dlaczego płynny obraz może pokazywać źle położoną kotwicę?

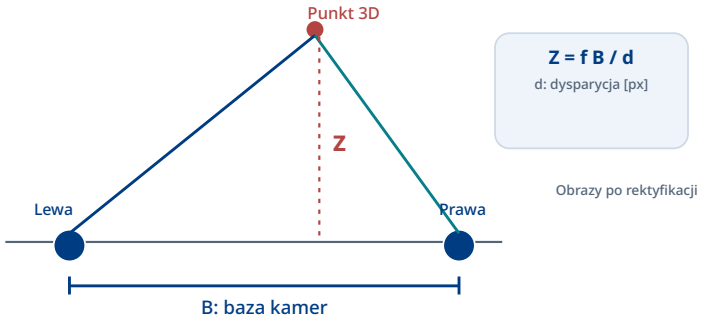


Trzy scenariusze błędnej relokalizacji kotwicy AR

Unity: krok D → E. Estymator zgłasza dopasowanie mapy, ale symulowana kotwica jest przesunięta o **0,35 m**. Test względem niezależnego celu przy tolerancji **0,05 m** odmawia akceptacji.

Hipotezy obejmują fałszywe dopasowanie, zmienioną scenę i starą transformację. Progi są **dydaktyczne**, nie normatywne.

Stereo: dysparycja wyznacza głębię



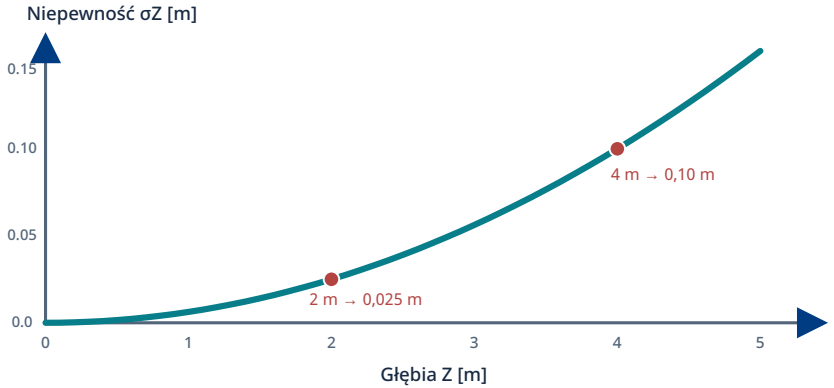
$$Z = \frac{fB}{d}, \qquad \frac{800\,\mathrm{px} \cdot 0,10\,\mathrm{m}}{40\,\mathrm{px}} = \boxed{2\,\mathrm{m}}\ .$$

```
float z = FocalPx * BaselineMeters / disparityPx;
```

Unity: suwak zmienia dysparycję w zakresie 10–80 px. Model do uruchomienia.

Założenie: kamera rektyfikowana, dane syntetyczne.

Niepewność głębi stereo narasta z kwadratem odległości



$$\sigma_Z \approx \frac{Z^2}{fB}\sigma_d.$$

Unity: zmniejszenie dysparycji z **40 px** do **20 px** zmienia głębię **2 m → 4 m** i modelową niepewność **2,5 cm → 10 cm**.

Założenia: $fB = 80\,\mathrm{px\,m}$, $\sigma_d = 0,5\,\mathrm{px}$, małe błędy i brak niepewności kalibracji.

Wynik **obliczeniowy**, nie pomiar.

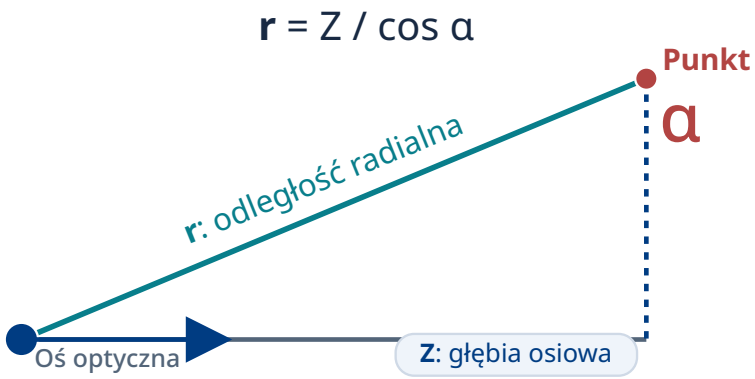
Metody pomiaru głębi mają różne źródła błędu



Stereo (także wspomagane projekcją wzorca) opiera się na korespondencjach. Światło strukturalne wykorzystuje znany wzór. **iToF** szacuje przesunięcie fazy, **dToF** mierzy czas impulsu; LiDAR może korzystać z metod czasu przelotu.

Nie istnieje jeden uniwersalny ranking sensorów niezależny od zasięgu, oświetlenia i materiału.

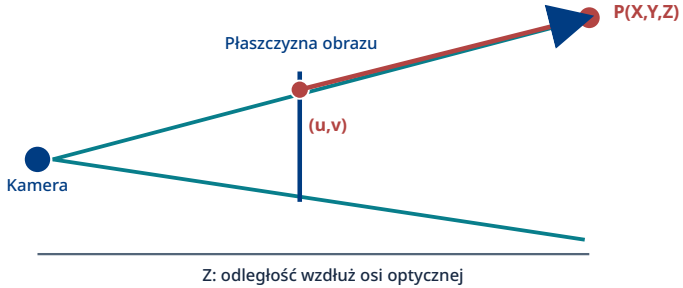
Głębia osiowa nie jest tym samym co odległość radialna



$$Z = r \cos \alpha.$$

Przed deprojekcją sprawdzamy, czy pomiar oznacza **głębię wzdłuż osi optycznej**, czy **odległość wzdłuż promienia**. Jednostki obrazu mogą wymagać konwersji mm na m.

Deprojekcja: piksel i głębia dają punkt przestrzenny



$$X = (u - c_x)\frac{Z}{f_x}, \qquad Y = (v - c_y)\frac{Z}{f_y}.$$

Dla $(u, v) = (410, 260)$, $Z = 1{,}35\,\mathrm{m}$, $f_x = f_y = 615\,\mathrm{px}$ i $(c_x, c_y) = (320, 240)$ otrzymujemy $(X, Y, Z) \approx (0{,}198;\, 0{,}044;\, 1{,}35)\,\mathrm{m}$.

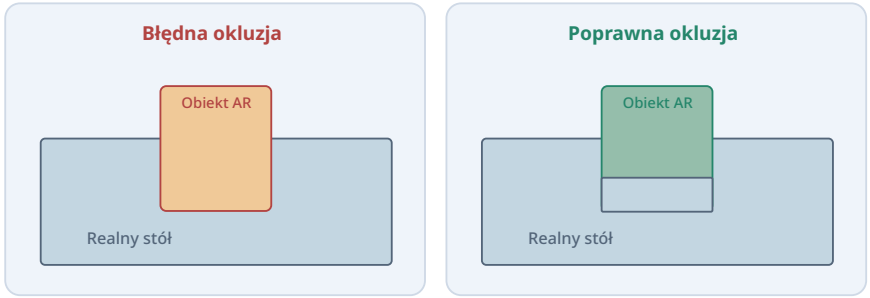
Uruchamialny kod z walidacją danych.

Chmura punktów opisuje geometrię; segmentacja obiekty



Geometria odpowiada na pytanie **gdzie znajduje się powierzchnia**. Segmentacja semantyczna odpowiada na pytanie **do jakiej klasy należy**, a instancyjna rozdziela poszczególne obiekty.

Okluzja wymaga porównywalnych głębokości



Unity: zgodne głębokości tego samego promienia i czasu

```
bool realOccludes = usable && zReal < zVirtual;
```

Gdy $z_{real} < z_{virtual}$, rzeczywista powierzchnia powinna zasłonić obiekt cyfrowy. Demonstrator geometrii i Z-bufora używa dwóch brył; **nie jest implementacją sensora ani shadera AR**.

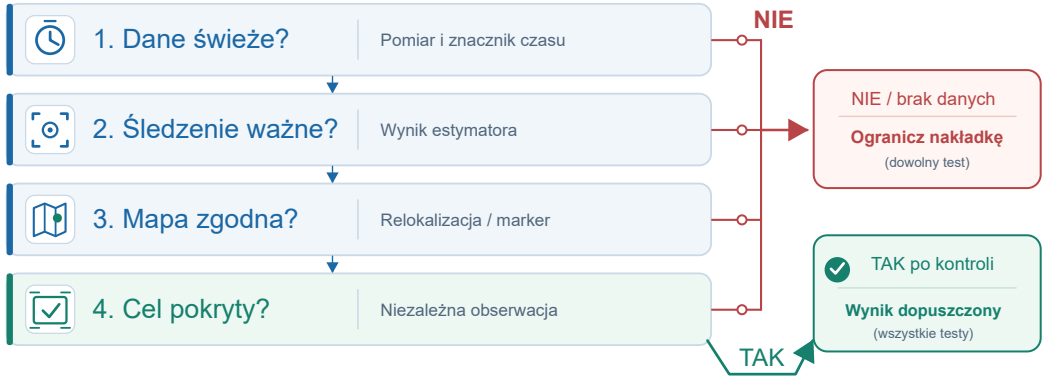
Brak wiarygodnej głębi nie oznacza pustej przestrzeni



Unity: wyłącz „Pomiar głębi dostępny“ lub „Głębia i obraz z tej samej chwili”. Poprawny wariant **ukrywa obiekt zależny od okluzji**; przełącznik błędnego wariantu pokazuje, co dzieje się bez bramkowania.

Brak próbki, desynchronizacja i błąd rejestracji nie są dowodem wolnej przestrzeni.

Procedura: kiedy nakładka może wrócić do użycia?



Implementacja Unity: `IsOverlayApproved` wymaga świeżej pozy, dopasowania mapy i niezależnie sprawdzonej referencji; po każdej korekcie rejestracji sprawdzenie jest ponawiane.

Wynik przypadku: błędna relokalizacja 0,35 m zostaje odrzucona; dopiero korekta oraz nowa kontrola celu pozwalają wyświetlić zieloną nakładkę.

Wideo demonstracyjne

Bibliografia

- O. Woodman, *An Introduction to Inertial Navigation*, 2007.
- T. Qin i in., *VINS-Mono*, 2018.
- C. Cadena i in., *Past, Present, and Future of SLAM*, 2016.
- ROS, REP-105: układy `map`, `odom`, `base_link`.
- Google ARCore, *Working with Anchors*.
- OpenCV, *Camera Calibration and 3D Reconstruction*.