Modele świata i polityki Vision–Language–Action w robotach humanoidalnych

Architektury WMA i VLA, implementacja na Unitree G1 EDU oraz protokół walidacji

Streszczenie

Przedmiotem opracowania są dwie klasy metod uczenia sterowania robotami: **World-Model–Action (WMA)**, w których predykcyjny model dynamiki środowiska wspomaga generowanie danych i wybór działania, oraz **Vision–Language–Action (VLA)**, w których zintegrowana polityka przekłada obserwacje wizyjne, instrukcję językową i stan robota na sekwencję akcji. Analiza dotyczy głównie architektur **UnifoLM-WMA-0** i **UnifoLM-VLA-0** oraz ich potencjalnej integracji z robotem humanoidalnym Unitree G1 EDU. Porównano przestrzenie wejść i wyjść, zależności czasowe, konfigurację modułu akcyjnego, wymagania programowe, formaty zbiorów danych, synchronizację czujników, schemat klient–serwer i ograniczenia czasu rzeczywistego. Zaproponowano wielowarstwową architekturę oddzielającą inferencję generatywną od deterministycznego wykonawcy umiejętności i zabezpieczeń funkcjonalnych. Omówiono procedury instalacji, przygotowania danych, kalibracji, walidacji poza robotem, eksperymentów HIL oraz raportowania skuteczności i awaryjności. Parametry podane w konfiguracjach modeli i przykładach wdrożeniowych są rozróżnione od wyników pomiarowych: **nie przedstawia się nowych eksperymentów ani nie przypisuje niewykonanym testom statusu walidacji empirycznej**. Wykazano, że VLA i WMA są architekturami komplementarnymi, jednak ich integracja wymaga niezależnej kontroli ograniczeń kinematycznych, kontaktu z człowiekiem, opóźnień sieciowych oraz zgodności licencyjnej.

Słowa kluczowe: robot humanoidalny, Unitree G1 EDU, world model, WMA, VLA, UnifoLM, uczenie przez demonstrację, multimodalna percepcja, action chunking, ROS 2, DDS, HIL, bezpieczeństwo robotyczne.

1. Wprowadzenie i problem badawczy

Sterowanie humanoidem w zmiennym środowisku wymaga współdziałania percepcji, reprezentacji zadania, rozumowania o skutkach interakcji oraz sterowania układami mechanicznymi. Klasyczne rozwiązania opierają się na rozdzieleniu planowania ruchu, identyfikacji obiektów, estymacji stanu i regulatorów sprzężenia zwrotnego. Modele uczenia reprezentacji multimodalnych wprowadzają alternatywny mechanizm: model uczony na demonstracjach może mapować obrazy i polecenia bezpośrednio na działania albo wykorzystywać predykcje przyszłych obserwacji jako pośrednią reprezentację dynamiki.

W architekturze **VLA** centralne jest odwzorowanie warunkowej polityki akcji. W architekturze **WMA** centralne jest modelowanie przejścia między stanami, potencjalnie umożliwiające symulowanie wielu następstw działania. Nie jest to podział rozłączny: współczesne polityki VLA mogą mieć komponenty predykcyjne, natomiast WMA może zawierać głowicę wyznaczającą akcje. Rozróżnienie dotyczy dominującej roli modelu świata w potoku decyzyjnym, a nie wymogu zastosowania konkretnego typu sieci neuronowej [1–3].

Problem wdrożeniowy dla Unitree G1 EDU obejmuje co najmniej cztery niezależne ograniczenia. Po pierwsze, wyjście modelu musi zostać przetłumaczone na układ współrzędnych i przestrzeń sterowania dostępnego efektora. Po drugie, obserwacje obrazu, IMU i stawów muszą reprezentować spójny moment czasu. Po trzecie, obliczenia na serwerze i transfer pakietów nie mogą powodować niekontrolowanej kontynuacji ruchu po utracie łączności. Po czwarte, generatywna polityka akcji nie jest równoznaczna z niezależnym układem bezpieczeństwa funkcjonalnego. Te wymagania powinny zostać spełnione przed wykonaniem zadań kontaktowych.

1.1. Cel i pytania badawcze

Celem jest zaprojektowanie odtwarzalnej ścieżki implementacyjnej pozwalającej porównać: (i) politykę VLA, (ii) wariant WMA z wyborem działania na podstawie predykcji oraz (iii) układ hybrydowy, w którym VLA proponuje działanie, a WMA ocenia jego możliwe następstwa. Z perspektywy naukowej istotne są następujące pytania:

1. W jakim stopniu predykcja przyszłych interakcji zwiększa skuteczność zadań wieloetapowych w porównaniu z bezpośrednią polityką VLA?
2. Jaki koszt czasowy i obliczeniowy wprowadza WMA względem wariantu VLA-only?
3. Jak zmieniają się odsetki błędów i odmów wykonania po zastosowaniu niezależnej warstwy Safety Gate?
4. Jak synchronizacja danych i opóźnienie sieciowe wpływają na jakość wykonania pakietów akcji?
5. Czy właściwości polityki wytrenowanej na danych manipulacyjnych są zachowywane w zadaniach poza rozkładem treningowym?

1.2. Charakter opracowania i granice wnioskowania

Przedstawiono **przegląd krytyczny architektur i dokumentacji technicznej**, projekt systemu i protokół walidacji. Wartości wynikające z kart modeli, plików konfiguracyjnych i przykładów uruchomieniowych mają status danych dokumentacyjnych. Liczby dotyczące częstotliwości, horyzontu, rozmiaru modeli i wymiaru akcji nie są wynikami pomiarów na konkretnym robocie. Nie należy utożsamiać deklarowanych demonstracji producenta z niezależną replikacją ani interpretować braku tabel benchmarków jako potwierdzenia skuteczności w innych zadaniach.

2. Formalizacja problemu sterowania

2.1. Reprezentacja stanu i obserwacji

Niech s_t oznacza stan fizyczny robota i środowiska w chwili t, o_t jego obserwację, u instrukcję zadaniową, a a_t działanie zadane kontrolerowi. Obserwacja multimodalna może zawierać obrazy z kamer głowy i nadgarstków, położenia i prędkości stawów, orientację tułowia, stan efektora i wskaźniki jakości sensorycznej:

$$o_t = \{I_t^{head}, I_t^{wrist}, q_t, \dot{q}_t, \mathrm{IMU}_t, c_t\}.$$

W praktyce zakres tych pól zależy od wyposażenia G1 EDU i wersji sterowników. Dane dokumentacyjne zbiorów manipulacyjnych wskazują na użycie kamer montowanych przy nadgarstkach oraz przy głowie, ale nie dowodzą, że identyczny zestaw jest zamontowany lub dostępny na każdym egzemplarzu robota.

2.2. Polityka VLA

Politykę VLA można ująć jako model warunkowy generujący sekwencję działań na podstawie obrazu, stanu i instrukcji:

$$\pi_\theta(a_{t:t+H-1} \mid o_{t-k:t}, u).$$

Parametr H to **action horizon**, a k określa długość historii obserwacji. Głowica akcyjna może wyznaczać cały pakiet działań, który następnie wykonuje się stopniowo. Taki mechanizm ogranicza częstotliwość kosztownej inferencji, ale wprowadza problem dezaktualizacji pakietu, gdy warunki środowiskowe zmienią się w trakcie jego realizacji.

2.3. Model WMA

Model świata estymuje przyszły stan lub obserwację warunkowo względem historii i potencjalnych akcji:

$$\hat{o}_{t+1:t+H} = f_\phi(o_{\le t}, a_{t:t+H-1}).$$

W wariancie z oceną kilku kandydatów model może posłużyć do porównania prognozowanych trajektorii. Przykładowa zasada projektowa polega na wyborze akcji minimalizującej koszt przewidywanego stanu, z zachowaniem twardych ograniczeń ruchu:

$$a^* = \arg\min_{a \in \mathcal{A}_{safe}} J\big(f_\phi(o_t, a), a\big).$$

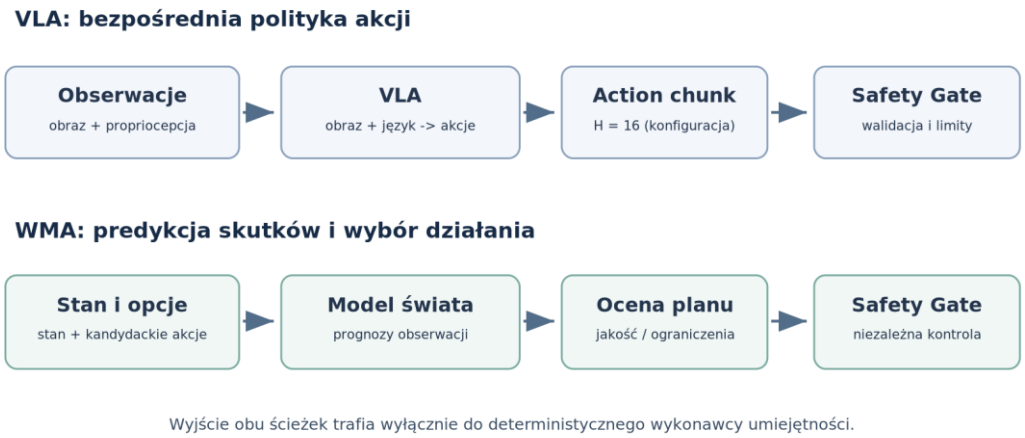
Jest to **formalizacja ogólnej architektury predykcyjnej**, a nie stwierdzenie, że każdy checkpoint UnifoLM-WMA-0 implementuje dokładnie tak zdefiniowaną funkcję kosztu. Przewidywana bezkolizyjność nie zastępuje deterministycznego sprawdzania ograniczeń.

2.4. Horyzont predykcji a sterowanie czasu rzeczywistego

Należy odróżnić częstotliwość akwizycji obrazu, częstotliwość uzyskiwania odpowiedzi modelu, częstotliwość przesyłania komend i częstotliwość wewnętrznych regulatorów robota. Przykład konfiguracji klienta WMA zawiera `--action_horizon 16`, `--exe_steps 16` i `--control_freq 15`. Tych wartości nie wolno bezpośrednio interpretować jako częstotliwości sterowania napędami. Ich znaczenie należy odczytać w implementacji klienta i zweryfikować pomiarem zegarów. Przy założeniu wykonania 16 kroków w tempie 15 Hz pakiet odpowiadałby nominalnie około 1,07 s, jednak jest to **wyliczenie warunkowe**, a nie pomiar opóźnienia inferencji [2].

3. Porównanie koncepcji WMA i VLA

Wymiar	WMA	VLA
Dominująca funkcja	Predykcja przyszłych obserwacji / wspomaganie decyzji	Bezpośrednie generowanie pakietów akcji
Wejście	Obserwacje, stany, kandydackie działania; instrukcja zależnie od modelu	Obrazy, instrukcja, opcjonalnie propriocepcja
Wyjście	Prognozy stanów/obserwacji, oceny lub działania z głowicy	Ciąg wartości sterujących lub przemieszczeń
Horyzont	Może obejmować rozwój sceny i sekwencje wieloetapowe	Zwykle odcinek wykonawczy action chunk
Mocna strona	Analiza następstw, augmentacja danych, planowanie	Jeden model wielu zadań manipulacyjnych
Słabe punkty	Błąd modelu świata, narastanie błędu prognozy, koszt symulacji	Wrażliwość na zmianę rozkładu, koszt inferencji, sterowanie otwartopętlowe w chunku
Warunek bezpieczeństwa	Niezależny filtr kandydatów i monitor wykonania	Niezależny filtr wygenerowanych działań i monitor wykonania



Zestawienie dwóch ścieżek decyzyjnych WMA i VLA.

Rysunek 1. Schemat porównawczy generowania decyzji w WMA i VLA. Oba wyjścia podlegają wspólnej kontroli bezpieczeństwa; schemat ma charakter projektowy.

3.1. Model świata jako symulator

UnifoLM-WMA-0 ma dwa jawnie opisane zastosowania. Jako **Simulation Engine** model generuje syntetyczne przebiegi interakcji pomocne w uczeniu polityki. Jako **Policy Enhancement** łączy przewidywanie przyszłych interakcji z głowicą akcyjną, aby wspomóc decyzję. Udostępniono bazowy checkpoint po dostrojeniu na zbiorze Open-X oraz wariant `Dual` trenowany na wybranych zbiorach Unitree w trybach decyzji i symulacji [2, 4].

Model generatywny można wykorzystać do wytwarzania dodatkowych trajektorii lub ocen scenariuszy „co jeśli", ale generacja obrazu poprawnego wizualnie nie gwarantuje poprawności fizycznej, zachowania kontaktów ani spójności kinematycznej. Trzeba oceniać zarówno trafność predykcji, jak i jakość działania. W szczególności model może tworzyć wizualnie przekonujące, lecz dynamicznie niemożliwe przyszłe stany, co prowadzi do błędnych decyzji w układzie planującym.

3.2. VLA jako polityka manipulacyjna

UnifoLM-VLA-0 opisuje wspólne przetwarzanie percepcji przestrzennej i instrukcji językowej, z dostrojeniem na danych manipulacyjnych. Producent wskazuje demonstracje dwunastu kategorii zadań na prawdziwym robocie G1 przy użyciu jednej polityki [1]. Takie demonstracje potwierdzają zakres pokazanych zastosowań, lecz bez niezależnych protokołów i przedziałów ufności nie stanowią podstawy do wyznaczenia ogólnego odsetka sukcesu w nowych warunkach.

Wspólny model może obsługiwać kilka poleceń bez ręcznego tworzenia osobnej reguły dla każdego zadania. Trudnością pozostają trajektorie o nietypowej geometrii, inne obiekty i przyrządy, zasłonięcia, błędy semantyczne instrukcji i niepewność kontaktu. Model musi współpracować z klasyczną warstwą sterowania i monitorowania stanu, zwłaszcza gdy manipulacja może spowodować utratę stabilności humanoida.

4. UnifoLM-VLA-0: konfiguracja techniczna

4.1. Backbone wizyjno-językowy i głowica akcyjna

W publicznym pliku konfiguracyjnym `UnifoLM-VLA-Base` zastosowano rodzinę `qwen2_5_vl` jako backbone przetwarzania wizualno-językowego. Dla akcji zdefiniowano osobny blok parametryzowany m.in. wymiarem wejściowym, ukrytym, oknem przyszłych działań oraz ustawieniami iteracyjnego próbkowania [5]. Oznacza to, że opis „VLM + działanie" nie sprowadza się do prostego dopisania tokena akcji do odpowiedzi tekstowej: architektura zawiera specjalizowany interfejs akcyjny.

Parametr konfiguracji UnifoLM-VLA-Base	Wartość	Znaczenie techniczne
model_type	qwen2_5_vl	Rodzina bazowego VLM
vl_hidden_dim	2048	Wymiar reprezentacji VLM
input_embedding_dim	1536	Wymiar wejściowy modelu akcji
hidden_size	1024	Wymiar ukryty modułu akcyjnego
action_dim	7	Wymiar wyjścia akcyjnego w tej konfiguracji
state_dim	7	Wymiar wejściowego stanu w tej konfiguracji
action_horizon	16	Liczba kroków generowanego horyzontu
future_action_window_size	15	Okno przyszłych działań
repeated_diffusion_steps	8	Parametr powtórzeń generacji iteracyjnej
num_inference_timesteps	4	Kroki inferencyjne w komponencie iteracyjnym
num_timestep_buckets	1000	Dyskretyzacja kroku czasowego w konfiguracji
num_target_vision_tokens	32	Docelowa liczba tokenów wizualnych
use_wrist_image	true	Włączone wejście obrazu z nadgarstka
use_proprio	true	Włączone wejście proprioceptywne
image_size	224 × 224	Rozmiar obrazów stosowanych podczas treningu
attn_implementation	flash_attention_2	Zadeklarowany mechanizm uwagi

Wartości `action_dim = 7` i `state_dim = 7` dotyczą tego checkpointu i nie oznaczają, że cały G1 EDU ma siedem stopni swobody. Potrzebne jest jawne mapowanie tych wymiarów na właściwą rękę, chwytak, konfigurację zadania i normalizację danych. Zakres 16 kroków nie stanowi gwarancji, że wszystkie 16 kroków powinno zostać wykonanych bez ponownego odczytu sensorycznego [5].

4.2. Parametry uczenia

Konfiguracja zawiera m.in. `epochs: 100`, `max_train_steps: 300000`, `per_device_batch_size: 16`, scheduler typu cosine, AdamW oraz oddzielne współczynniki uczenia dla bazowego bloku VLM i modułu akcji. Wybrana para wartości `qwen_vl_interface: 1e-5` i `action_model: 4e-4` wskazuje różne tempo adaptacji elementów architektury. Część parametrów dotyczy procedury treningu autora i nie powinna być bezpośrednio kopiowana do eksperymentów z mniejszym zbiorem danych bez przeglądu harmonogramu uczenia.

Dostępne repozytorium zawiera checkpoint rzędu **19 GB plików**. Wielkość repozytorium na dysku nie jest tożsama z wymaganiami VRAM podczas inferencji. Rzeczywiste zapotrzebowanie zależy m.in. od typów tensorów, optymalizacji, liczby kamer, długości sekwencji, sposobu przechowywania aktywacji oraz modelu wykonawczego [5, 6].

4.3. Przedtrening i adaptacja między zadaniami

Z punktu widzenia transferu umiejętności odróżnić należy reprezentacje ogólne (język, obraz, semantyka sceny) i reprezentacje ucieleśnione (konkretna kinematyka, przestrzeń działań, kontakt, specyficzna geometria efektora). Przeniesienie modelu na inny wariant dłoni lub na inne rozmieszczenie kamer wymaga

co najmniej walidacji mapowania stanów i działań oraz ponownej kalibracji obserwacji. Samo dopasowanie wymiaru wejścia do konfiguracji sieci nie potwierdza transferowalności polityki.

5. UnifoLM-WMA-0: mechanizm i wdrożenie

5.1. Etapy trenowania modelu świata

W dokumentacji UnifoLM-WMA-0 opisano dostrajanie bazowego modelu generacji wideo z wykorzystaniem Open-X, następnie trening w trybie `decision-making`, a opcjonalnie także w trybie `simulation`. W przypadku własnych demonstracji kluczowe jest przygotowanie zsynchronizowanych obserwacji i akcji oraz ustalenie jednoznacznej konwencji sterowania. Przewidzenie obrazu przyszłej sceny nie gwarantuje poprawnego oszacowania kontaktu, dlatego model świata wymaga walidacji również na stanach kinestetycznych i zdarzeniach zadaniowych [2].

5.2. Warianty checkpointów

Wariant	Deklarowana specjalizacja	Wniosek wdrożeniowy
UnifoLM-WMA-0-Base	Model bazowy dostrojony na Open-X	Punkt odniesienia przy ocenie transferu
UnifoLM-WMA-0-Dual	Dodatkowy trening na zbiorach Unitree w trybach decision-making i simulation	Wariant interesujący dla eksperymentów na danych producenta
WMA + zewnętrzna polityka	Świat-model jako selektor/oceniający	Eksperyment hybrydowy z niezależnym generatorem akcji

Modele WMA mogą być dostępne w trybie wymagającym akceptacji warunków dostępu (**gated**). Konieczność uwierzytelnienia należy uwzględnić w procedurze pozyskania artefaktów; nie wolno zakładać, że każdy publicznie opisany checkpoint można anonimowo pobrać i użyć w zautomatyzowanym środowisku [4].

5.3. Architektura klient–serwer

Oficjalna implementacja decyzyjna przewiduje uruchomienie inferencji na komputerze serwerowym, przy jednoczesnym prowadzeniu zbierania obserwacji i wykonywania działań przez klienta po stronie robota. Taka separacja upraszcza użycie cięższych modeli GPU, ale wprowadza opóźnienie sieciowe, ryzyko przestarzałych akcji oraz potrzebę ograniczenia domeny sterowania [2]. Klient musi zawsze zachować zdolność zatrzymania i bezpiecznego przejścia do stanu awaryjnego niezależnie od dostępności serwera.

W serwerowym scenariuszu uruchomieniowym występują konfiguracje `ckpt`, `res_dir`, `datasets`, `data_dir` i `dataset_and_weights`; należy je uzupełnić przed wykonaniem `scripts/run_real_eval_server.sh`. Po stronie klienta udostępniono skrypt `robot_client.py` i tunelowanie portu 8000. Jest to przykład publicznej ścieżki integracji, a nie uniwersalny zestaw parametrów dla dowolnego firmware.

6. Unitree G1 EDU: platforma i urządzenia wykonawcze

6.1. Zakres dostępnego rozwoju

Producent odróżnia podstawowy model Unitree G1 i wariant **G1 EDU**. W informacji handlowej standardowy G1 jest wskazywany jako model bez obsługi secondary development; do rozbudowy aplikacyjnej producent kieruje użytkownika do edycji EDU [7]. Rozpoczęcie integracji wymaga jednak potwierdzenia rzeczywistej konfiguracji G1 EDU, w szczególności rodzaju dłoni, liczby dostępnych przegubów, wersji sterowników, oprogramowania wbudowanego, dostępu do kamer i topologii sieci.

W publicznych specyfikacjach dla rodziny G1 pojawia się zakres **23–43 stopni swobody**, zależny od konfiguracji, oraz masa około 35 kg. Te dane nie zastępują inwentaryzacji egzemplarza. Dostępne opcjonalnie dłonie Dex3-1 mają po siedem stopni swobody i mogą występować w wersjach wyposażonych

w czujniki dotykowe. Manipulacja przedmiotami zależy od tego, czy skonfigurowano właściwy wariant efektorów oraz czy API udostępnia odpowiednie tryby sterowania [7].

6.2. Rozdzielenie warstw sterowania

Architektura bezpiecznego systemu powinna oddzielać warstwę obliczeń wysokiego poziomu od warstw stabilizacji i sterowania napędami. Inferencja WMA/VLA ma charakter niedeterministyczny czasowo; nie należy włączać jej w bezpośrednią pętlę sterowania wymagającą twardego czasu rzeczywistego. Model może proponować ruch lub referencję zadania, ale weryfikacja stanu robota, nasycenie prędkości, ograniczenia kinematyczne i procedura E-Stop muszą być wykonywane poza modelem.

Nie należy przyjmować, że istnieje jedna niezmienna nazwa lub pełna zgodność wysokopoziomowych interfejsów Unitree między wersjami firmware. Dostępność usług, formatów wiadomości i predefiniowanych umiejętności wymaga testu odczytowego na konkretnym stanowisku oraz porównania z właściwą wersją SDK [8, 9].

6.3. Profil urządzeń do inferencji

Dla dużych checkpointów UnifoLM uzasadnione jest uruchomienie inferencji na oddzielnym komputerze ze sprzętową akceleracją, a samego klienta sensoryczno-sterującego na komputerze developerskim robota. Konfigurację referencyjną można rozdzielić na: G1 EDU z SDK2/DDS, proces ROS 2 do normalizacji obserwacji, serwer GPU z VLA/WMA oraz niezależny proces kontroli bezpieczeństwa. Wariant w pełni pokładowy wymaga dodatkowej weryfikacji zużycia RAM/VRAM, opóźnienia i termiki; rozmiar repozytorium z modelem nie jest wystarczającym kryterium.

7. Akwizycja i synchronizacja multimodalna

7.1. Kamery

W zbiorach manipulacyjnych Unitree opisano konfigurację kamer obejmującą monokularne kamery przy nadgarstkach i stereoskopową konfigurację kamer przy głowie. Przykładowe nagrania mają rozdzielczość **640 × 480 px** i częstotliwość **30 Hz**. Informacje te stanowią punkt odniesienia dla projektowania akwizycji, nie są natomiast potwierdzeniem sprzętowych możliwości wszystkich wariantów G1 EDU [1, 10].

Dla każdego strumienia należy rejestrować: identyfikator urządzenia, stempel czasu akwizycji, stempel czasu odbioru, rozdzielczość, identyfikator kalibracji, parametry intrynsyczne oraz orientację kamery względem układu bazowego. W okresie treningu transformacje `crop`, `resize` i normalizacja obrazu muszą odpowiadać transformacjom używanym w inferencji. W przeciwnym razie zmiana skali i pola widzenia powoduje przesunięcie rozkładu wejściowego.

7.2. IMU i propriocepcja

Warstwa stanu powinna integrować orientację tułowia, prędkości kątowe, położenia i prędkości stawów oraz stan dłoni. W ekosystemie Unitree SDK2 i `unitree_mujoco` pojawia się dedykowany temat `rt/secondary_imu` dla G1 oraz wiadomości stanu niskopoziomowego (`lowstate`) [8, 11]. Należy odrębnie ustalić częstotliwość publikacji oraz to, które sygnały są bezpieczne do odczytu bez ingerowania w zamknięte kontrolery robota.

Synchronizacja danych stanowi warunek interpretowalności etykiet akcji. Obserwacja z kamery sprzed kilkuset milisekund i bieżące położenia stawów nie opisują tego samego stanu. Należy zastosować stałą referencję zegara i zdefiniować maksymalny dopuszczalny wiek danych; po jego przekroczeniu system powinien odrzucić obserwację i wstrzymać nowy plan. Dla sterowania poprzez sieć potrzebne są osobne metryki czasu akwizycji, preprocessingu, inferencji, transmisji i wykonania.

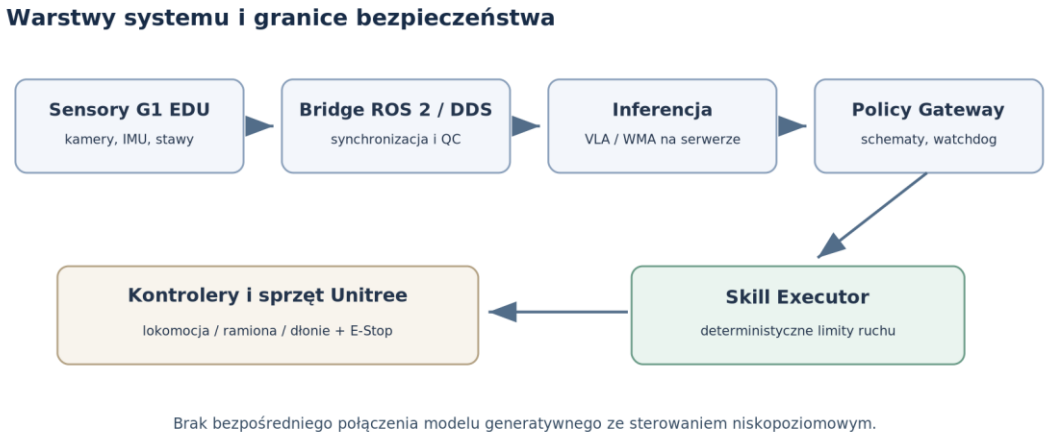
7.3. Rejestracja semantyki kontaktu

Dla manipulacji celowe jest rejestrowanie punktów styku, siły chwytu lub dostępnego wskaźnika kontaktu, statusu chwytu, utraty obiektu i błędów granicznych. Jeżeli sprzęt nie udostępnia czujników dotykowych, należy jawnie odnotować ten brak i nie zastępować go domniemaną informacją z obrazu. W raportowaniu badań trzeba rozróżnić wynik percepcji, oszacowanie stanu kontaktu i rzeczywisty odczyt czujnikowy.

7.4. Jakość i spójność danych

Proponowane kryteria kontroli jakości obejmują kompletność sekwencji, liczbę pominiętych klatek, błędy synchronizacji, zakresy dopuszczalnych wartości stawów, spójność układów współrzędnych oraz identyfikację zresetowanych sensorów. Eksperyment, w którym obraz i akcje zostały zestawione z błędem synchronizacji, powinien być oznaczony jako wadliwy, a nie wykorzystywany do oceniania błędu modelu.

8. Architektura systemowa dla G1 EDU



Modułowa architektura integracji WMA i VLA z kontrolerami Unitree.

Rysunek 2. Architektura modułowa z wydzieleniem percepcji, inferencji, Safety Gate i deterministycznej warstwy wykonawczej.

8.1. Warstwy i odpowiedzialności

Akwizycja. Kamery, IMU i stany przegubów są zbierane z oznaczeniem czasu i identyfikatora konfiguracji. Węzeł nie wysyła poleceń ruchu; jego zadaniem jest zapewnienie wiarygodnych danych i sygnalizowanie niekompletności obserwacji.

Agregacja i przygotowanie obserwacji. Węzeł normalizuje obrazy, transformuje współrzędne, sprawdza zakresy i buduje niezmienny pakiet obserwacji. Każdy pakiet ma numer sekwencji i czas ważności. W ten sposób serwer inferencji otrzymuje jednoznaczny kontekst danego wywołania.

Inferencja. Serwer WMA/VLA otrzymuje wyłącznie dane niezbędne do zadania. Dla VLA zwraca pakiet akcji; dla WMA może zwracać predykcje i wariant działania. Interfejs jest typowany i wersjonowany; odpowiedzi niezgodne ze schematem są odrzucane.

Policy Gateway / Safety Gate. Bramka waliduje typ działania, jego czas przydatności, ograniczenia przestrzeni pracy, strefy człowieka, limity ruchu i stan stabilności robota. Zabezpieczenie musi działać nawet przy niedostępnym modelu, bez dostępu do Internetu i bez odpowiedzi serwera. W szczególności nie wolno wykonywać kodu arbitralnego przekazanego w odpowiedzi modelu.

Skill Executor. Wykonawca działań mapuje zatwierdzony plan na wspierany interfejs robota. Utrzymuje watchdog potwierdzeń i umożliwia natychmiastowe przejście do stanu bezpiecznego. Odwrócenie

mapowania wyjścia z `action_dim=7` na komendy realnego efektora musi zostać przetestowane na sucho przed pierwszym ruchem.

8.2. Zalecany kontrakt wiadomości

Poniższy schemat służy jako **propozycja integracyjna** i nie jest natywną definicją komunikatów SDK2. Wersje serializacji mogą być zrealizowane jako wiadomości ROS 2, protobuf lub walidowany JSON, pod warunkiem zapewnienia typowania i limitów rozmiarów.

```yaml
observation:
  schema_version: "1.0"
  robot_id: "g1-edu-lab"
  sequence_id: 1042
  acquired_at_ns: 0         # w produkcji: rzeczywisty znacznik czasu
  calibration_id: "cam-calib-v1"
  frames: [head_left, head_right, wrist_left, wrist_right]
  proprio: [joint_positions, joint_velocities, imu]
  instruction: "przesuń miękki obiekt do strefy A"
  quality:
    synchronized: true
    missing_frames: 0

policy_response:
  schema_version: "1.0"
  source_sequence_id: 1042
  action_space: "arm_ee_delta_v1"
  horizon: 16
  model_version: "checkpoint-sha256"
  expires_after_ms: 200     # PRZYKŁADOWA wartość projektowa
  actions: []              # pełny blok wartości numerycznych
```

Wartość `expires_after_ms` jest przykładem elementu kontraktu, a nie parametrem zweryfikowanym dla UnifoLM. Produkcyjny próg ważności należy dobrać na podstawie pomiaru opóźnień, charakterystyki kontrolera i ryzyka zadania. W systemach kontaktowych może być konieczne jeszcze krótsze okno oraz potwierdzanie stanu przed każdą akcją.

8.3. Automat stanów bezpieczeństwa

Rekomendowany automat wykonawczy przyjmuje stany `DISARMED`, `READY`, `INFERENCE_PENDING`, `ACTION_VALIDATED`, `EXECUTING`, `HOLD` i `FAULT`. Przejście do `EXECUTING` wymaga aktualnego potwierdzenia operatora, spójnego stanu sensorycznego oraz pozytywnej walidacji granic. Niewystarczająco świeża obserwacja, nieprawidłowa liczba elementów akcji, brak potwierdzenia kontrolera, utrata synchronizacji lub wykrycie niebezpiecznej konfiguracji powodują odrzucenie działania. Urządzenie powinno posiadać odrębny, niezależny od procesu AI mechanizm zatrzymania awaryjnego.

9. Wymagania sprzętowe i programowe

9.1. Macierz wdrożeniowa

Komponent	Wymaganie / środowisko referencyjne	Status i ograniczenia
Robot	Unitree G1 EDU, właściwy zestaw efektorów	Potwierdzić numer wariantu i zakres API
SDK2	unitree_sdk2 / unitree_sdk2_python, DDS	Test wersji biblioteki i obsługiwanych usług
ROS 2	Ubuntu 22.04 + Humble jako znana konfiguracja	Dobór dystrybucji do BSP i sterowników
Python WMA	3.10.18 w środowisku conda	Wskazany przez repo WMA
Pinocchio WMA	3.2.0	Zgodność z konfiguracją środowiska
FFmpeg WMA	7.1.1	Przetwarzanie sekwencji danych

Komponent	Wymaganie / środowisko referencyjne	Status i ograniczenia
VLA	CUDA 12.4, FlashAttention 2.5.6	Wymagania repo bazowego; sprawdzić sterownik
LeRobot	Rewizja zależna od implementacji; w VLA wskazana 0878c68	Nie utożsamiać bez testów z najnowszą wersją
Serwer inferencji	x86-64 z akceleracją GPU	Dokładne VRAM/RAM ustalić pomiarem
Zbiór danych WMA	LeRobot v2.1 dla opisanej ścieżki WMA	Wersjonować konwersję z v3.0

Repozytorium `unifolm-vla` rekomenduje CUDA 12.4, Python 3.10.18 oraz `flash-attn==2.5.6`; repo `unifolm-world-model-action` podaje zestaw zależności obejmujący `pinocchio=3.2.0` i `ffmpeg=7.1.1` [1, 2]. Takie powiązanie wersji jest ważniejsze dla powtarzalności niż instalowanie najnowszych pakietów bez ograniczeń.

9.2. Rozmiar checkpointu i budżet pamięci

Przykładowa karta `UnifoLM-VLA-Base` wskazuje repozytorium około 19 GB, natomiast checkpoint WMA-Base to plik rzędu 10 GB [4, 6]. Wymagania wykonawcze mogą być większe niż masa samych wag z powodu aktywacji, buforów, pamięci uwagi i współbieżnych procesów. Nie należy ustalać „minimum 24 GB VRAM" jako wymogu producenta bez benchmarku. Ocena zasobów powinna rozróżniać: rozmiar na dysku, szczytowe zużycie pamięci GPU, RAM systemowy, czas inicjalizacji, koszt inferencji jednego pakietu oraz pobór mocy i temperaturę pod obciążeniem.

9.3. Topologia sieci i separacja środowisk

Sieć robota i serwera powinna ograniczać ruch tylko do niezbędnych portów. Tunel SSH w przykładzie WMA służy do połączenia z usługą inferencji, lecz nie zabezpiecza automatycznie całego łańcucha sterowania. Dostęp zdalny nie może umożliwiać obejścia Safety Gate; narzędzia debugowania oraz tryby niskopoziomowego sterowania trzeba oddzielić od procesów eksperymentalnych. Dla testów modelu początkowo należy używać symulatora i atrap usług sterowania, nie fizycznego robota.

10. Instalacja i uruchomienie środowiska badawczego

10.1. Konfiguracja WMA

Poniższa sekwencja przedstawia instalację opisaną przez repozytorium WMA. Konkretne wersje pakietów i ścieżki należy zamrozić w pliku środowiska. Komendy instalacyjne nie wykonują ruchu fizycznego robota [2].

```
conda create -n unifolm-wma python==3.10.18
conda activate unifolm-wma
conda install pinocchio=3.2.0 -c conda-forge -y
conda install ffmpeg=7.1.1 -c conda-forge

git clone --recurse-submodules \
  https://github.com/unitreerobotics/unifolm-world-model-action.git
cd unifolm-world-model-action
pip install -e .
cd external/dlimp
pip install -e .
```

Po instalacji należy wykonać test importów bez dostępu do robota, spisać wersje bibliotek oraz rozwiązać zależności formatu danych. Zgodnie z instrukcją producenta pliki konfiguracyjne serwera zawierają ścieżki checkpointów i datasetów. Ich struktura nie może być zastąpiona wartościami domyślnymi domniemanymi przez użytkownika.

10.2. Uruchomienie serwera WMA

W wariancie klient–serwer serwer inferencji uruchamia się po skonfigurowaniu checkpointu, katalogu wyników i zbiorów danych. Poniższy przykład podaje dokumentowaną nazwę skryptu, ale nie gwarantuje poprawności ustawień na danym stanowisku:

```
conda activate unifolm-wma
cd unifolm-world-model-action
# Najpierw: konfiguracja ścieżek w skrypcie i pliku YAML
bash scripts/run_real_eval_server.sh
```

Po stronie klienta dokumentowany jest tunel SSH do serwera:

```
ssh user_name@remote_server_IP -CNg -L 8000:127.0.0.1:8000
```

W repozytorium znajduje się także przykład argumentów klienta:

```
python scripts/robot_client.py \
  --robot_type "g1_dex1" \
  --action_horizon 16 \
  --exe_steps 16 \
  --observation_horizon 2 \
  --language_instruction "pack black camera into box" \
  --output_dir ./results \
  --control_freq 15
```

Ten ostatni fragment jest poleceniem dla klienta robota. Nie należy uruchamiać go na fizycznym G1 przed potwierdzeniem mapowania efektorów, usług, trybu kontroli, przestrzeni pracy, zewnętrznego E-Stop i testów symulacyjnych. Parametry `g1_dex1`, `exe_steps` i `control_freq` odnoszą się do przykładu autora i nie są uniwersalną konfiguracją bezpieczeństwa [2].

10.3. Konfiguracja VLA

Dla VLA producent opisuje osobne środowisko i odwołanie do konkretnej rewizji LeRobot [1]:

```
conda create -n unifolm-vla python==3.10.18
conda activate unifolm-vla

git clone https://github.com/unitreerobotics/unifolm-vla.git
cd unifolm-vla
pip install --no-deps \
  "lerobot @ git+https://github.com/huggingface/lerobot.git@0878c68"
pip install -e .
pip install "flash-attn==2.5.6" --no-build-isolation
```

Tego stosu nie należy mieszać bez weryfikacji z innym środowiskiem WMA. W praktyce rozsądne są osobne środowiska, osobne manifesty checkpointów, jawny obraz kontenera lub środowisko conda oraz niezależne testy importów i wyjścia modelu. Przed pobraniem danych i checkpointów należy sprawdzić dostęp i licencje każdego artefaktu.

10.4. Zależności SDK, symulacji i LeRobot

Repozytoria Unitree udostępniają `unitree_sdk2`, `unitree_sdk2_python`, `unitree_mujoco` i warianty integracji z LeRobot. Aktualna publiczna nazwa odpowiedniego repozytorium treningowego występuje jako `unitree_lerobot`, natomiast w starszych materiałach spotyka się `unitree_IL_lerobot`. Zmienność nazewnictwa wymaga pinowania konkretnego URL i SHA, a nie polegania na samym opisie [8, 11, 12].

10.5. Kontrola wersji i reprodukowalność

Minimalny manifest eksperymentu powinien zawierać SHA wszystkich repozytoriów, wersję modelu, sumę SHA-256 checkpointu, wersję sterowników GPU, pakiety środowiska, dane urządzenia, wariant dłoni, mapowanie akcji, identyfikator kalibracji kamer i format zbioru danych. Każdy wynik powinien być odtwarzalny bez pobierania niekontrolowanych nowych rewizji z gałęzi `main`.

11. Zbiory danych, transfer i przygotowanie trajektorii

11.1. Publiczne zadania manipulacyjne

W ekosystemie UnifoLM-VLA-0 udostępniono dwanaście rodzin demonstracji manipulatora G1. Obejmują one układanie klocków, wkładanie przedmiotów do torby, ścieranie tablicy, czyszczenie blatu, pakowanie piórnika, przelewanie, pakowanie piłeczek, przygotowanie owoców, porządkowanie narzędzi, składanie ręcznika, wycieranie stołu oraz współpracę dwóch robotów przy czyszczeniu stołu. Domena tych zadań jest silnie skoncentrowana na manipulacji stołowej i stanowi użyteczny, ale niepełny opis możliwych zachowań humanoida [1].

Zbiór / scenariusz	Charakter działania	Podstawowe trudności oceny
G1_Stack_Block	Układanie i stabilizacja	Precyzja chwytu, stabilność ułożenia
G1_Bag_Insert	Umieszczanie obiektu w pojemniku podatnym	Odkształcenia, zasłonięcia
G1_Erase_Board	Kontakt z powierzchnią	Siła nacisku, dokładność trajektorii
G1_Clean_Table	Przemieszczanie przedmiotów	Kolizje i przeplanowanie
G1_Pack_PencilBox	Sekwencja układania	Kolejność i relacje przestrzenne
G1_Pour_Medicine	Manipulacja naczyniami	Utrzymanie orientacji, ryzyko rozsypania
G1_Pack_PingPong	Chwyt lekkich przedmiotów	Delikatność chwytu, powtarzalność
G1_Prepare_Fruit	Sekwencja czynności	Zmienność kształtu przedmiotów
G1_Organize_Tools	Selekcja i sortowanie	Rozpoznanie i lokalizacja
G1_Fold_Towel	Manipulacja materiałem odkształcalnym	Wielokontaktowość, brak sztywnej geometrii
G1_Wipe_Table	Przesuwanie narzędzia po podłożu	Powierzchnia kontaktu i ocena zakończenia
G1_DualRobot_Clean_Table	Kooperacja wielorobotowa	Synchronizacja dwóch agentów

Dla własnych eksperymentów nie należy łączyć bez kontroli wszystkich zbiorów z jedną skalą normalizacji. Skala stawów i przemieszczeń, układ odniesienia efektora oraz konwencje reprezentacji chwytu mogą różnić się między zadaniami. Należy utrzymywać jawny słownik `robot_variant`, `action_space`, `state_space`, `control_rate` i `calibration_id`.

11.2. LeRobot v2.1, HDF5, RLDS i LeRobot v3.0

Repozytoria WMA i VLA zawierają procedury przygotowania własnych danych z formatu **LeRobot v2.1**, lecz ich dalsze potoki przetwarzania nie są identyczne. WMA stosuje skrypt `prepare_training_data.py`, tworząc strukturę z filmami, przejściami w HDF5 i plikiem CSV; VLA opisuje przepływ **LeRobot → HDF5 → RLDS**, w tym wywołanie `convert_lerobot_to_hdf5.py` i budowanie datasetu przez `tfds build` [1, 2]. W obu przypadkach ważna jest spójność indeksowania i czasu obserwacji względem akcji.

```
# Przykład ścieżki przygotowania danych UnifoLM-WMA-0
cd prepare_data
python prepare_training_data.py \
```

```
  --source_dir /path/to/source_dir \
  --target_dir /path/to/output \
  --dataset_name "dataset1_name" \
  --robot_name "Unitree G1 Robot with Gripper"
# Przykład etapów konwersji dla UnifoLM-VLA-0
cd prepare_data
python convert_lerobot_to_hdf5.py \
  --data_path /path/to/lerobot_dataset \
  --target_path /path/to/hdf5_output
cd hdf5_to_rlds/rlds_dataset
tfds build --data_dir /path/to/rlds_output
```

Nowszy format **LeRobotDataset v3.0** organizuje wiele epizodów w większych kontenerach danych, upraszcza indeksowanie i wspiera streaming. Nie jest automatycznie kompatybilny z narzędziami oczekującymi v2.1. Poprawna ścieżka wymaga wersjonowanej konwersji i regresyjnego sprawdzenia zgodności sekwencji obserwacja–akcja; należy zweryfikować również obecność metadanych epizodów i wywołanie procedur finalizujących zbiór [13].

11.3. Ograniczenie kamer w niektórych ścieżkach WMA

Instrukcja przygotowania danych WMA wskazuje, że określona ścieżka treningowa obsługuje **tylko wejście z kamery głównej**. W przypadku danych wielowidokowych konieczne jest dostosowanie `data_dir` w pliku CSV lub przebudowanie ładowarki danych. To istotna różnica wobec konfiguracji UnifoLM-VLA-Base, która jawnie dopuszcza obraz z nadgarstka [2, 5]. Nie należy zatem porównywać WMA i VLA jako identycznie wyposażonych metod bez wyrównania liczby kamer oraz budżetu wejściowych danych.

11.4. Projekt podziału zbioru danych

Proponowany protokół rozdziela epizody na podzbiory treningowe, walidacyjne i testowe tak, aby nie wystąpiło przeciekanie niemal identycznych klatek z tej samej trajektorii. Zmiana warunków może obejmować nowe obiekty, inne oświetlenie, odmienne położenia startowe oraz rozbieżne instrukcje. Dobór próby należy ustalić przed eksperymentem; wyniki z tego samego stanowiska i bardzo podobnych trajektorii nie są dowodem ogólnej odporności na przesunięcie domeny.

11.5. Reprodukowalność danych

Każdemu epizodowi należy przypisać identyfikator, identyfikator konfiguracji sprzętowej, numer wersji programu sterującego, czas rozpoczęcia, wersję modelu, statystyki synchronizacji i wynik zadania. Etykiety sukcesu powinny wynikać z jawnych warunków, np. „obiekt umieszczono w strefie docelowej bez upuszczenia i bez naruszenia ograniczeń”, a nie z deklaracji modelu, że zadanie zakończono pomyślnie.

12. Przykładowe scenariusze implementacyjne

12.1. Podążanie za obiektem jako test integracyjny

Przed zastosowaniem VLA warto sprawdzić podstawową ścieżkę kamera → detekcja obiektu → decyzja → symulowana komenda prędkości. Prosty regulator oparty na odchyleniu obiektu od środka kadru może służyć jako test infrastruktury, ale nie jest alternatywą dla pełnego kontrolera bezpiecznego ruchu humanoida. Należy zaczynać od demonstracji w symulacji, przy wyłączonych napędach i z ograniczonym zakresem ruchu.

```
# Pseudokod, nie interfejs SDK2. Bez wyjścia na napędy.
class TrackingDecision:
    def __init__(self, kp_yaw=0.8):
        self.kp_yaw = kp_yaw

    def propose(self, width_px, target_center_px):
```

```
        if target_center_px is None:
            return {"action": "HOLD", "reason": "target_lost"}
        error_x = (target_center_px - width_px / 2) / (width_px / 2)
        return {"action": "ROTATE_SUGGESTION",
                "normalized_error": error_x,
                "proposed_yaw": -self.kp_yaw * error_x}

# Propozycję weryfikuje niezależny kontroler symulacyjny.
```

Tego rodzaju baseline umożliwia oddzielenie awarii warstwy komunikacyjnej od błędu modelu uczonego. W eksperymencie należy dodatkowo uwzględnić przypadki braku celu, nakładających się obiektów i szybkiej zmiany kierunku ruchu.

12.2. VLA do wykonania polecenia manipulacyjnego

W scenariuszu językowym moduł tekstowy lub ASR przekazuje instrukcję w postaci ujednoliconego napisu. Polityka otrzymuje obraz, wektor stanu i instrukcję, następnie produkuje **action chunk**. Przed wykonaniem każda akcja musi być sprawdzona, a realizacja przerwana, jeżeli zmieni się stan robota lub spadnie wiarygodność obserwacji.

```
# Pseudokod testowy: bezpośrednio nie steruje robotem.
def propose_vla_action(model, observation, instruction, safety):
    if not safety.observation_fresh(observation):
        return {"status": "REJECTED", "reason": "stale_observation"}

    chunk = model.predict(observation, instruction)
    if not safety.schema_valid(chunk):
        return {"status": "REJECTED", "reason": "invalid_schema"}
    if not safety.within_limits(chunk, observation):
        return {"status": "REJECTED", "reason": "safety_limits"}

    return {"status": "VALIDATED", "candidate": chunk}
```

Najważniejsze jest rozdzielenie dwóch etapów: **model proponuje**, a **deterministyczna warstwa decyduje o dopuszczeniu**. W wyniku oceny bezpieczeństwa część poprawnie semantycznych instrukcji może zostać odrzucona. Takie odrzucenie należy raportować oddzielnie od błędu percepcji lub języka.

12.3. WMA do oceny kandydatów

W wariancie hybrydowym VLA może wygenerować jeden lub wiele kandydackich pakietów akcji, a WMA ocenia ich potencjalne skutki. Zestaw kryteriów obejmuje przewidywane naruszenia strefy roboczej, utratę przedmiotu, zbyt duże przemieszczenia i czas osiągnięcia celu. Nie należy traktować predykcji WMA jako wystarczającego dowodu bezpieczeństwa; ograniczenia geometryczne i dynamiczne muszą być sprawdzone osobno.

```
# Pseudokod oceny kandydatów w symulacji.
def select_candidate(world_model, state, candidates, hard_gate):
    admissible = []
    for candidate in candidates:
        if not hard_gate.within_limits(candidate, state):
            continue
        prediction = world_model.rollout(state, candidate)
        score = evaluate_task_and_risk(prediction)  # eksperymentalny koszt
        admissible.append((score, candidate))
    if not admissible:
        return None
    return min(admissible, key=lambda item: item[0])[1]
```

Wariant powinien być oceniany w ablacjach: bez WMA, z WMA jako oceniającym, z WMA jako generatorem danych oraz z WMA i kontrolą ryzyka. Tylko w ten sposób można ustalić, czy wzrost skuteczności wynika z predykcji, większego czasu obliczeń czy innego sposobu doboru akcji.

12.4. Polecenia głosowe

Ścieżka mowa → tekst → model może ułatwić obsługę, ale dodatkowo wprowadza niepewność rozpoznania mowy. Przed zastosowaniem polecenia w ruchu należy zdefiniować sposób potwierdzenia niejednoznacznych komend i odrzucania instrukcji o niebezpiecznej semantyce. Jako osobne metryki warto raportować Word Error Rate, liczbę powtórzeń i odsetek błędnych komend wywołanych przez ASR. Obsługa audio w SDK może zależeć od wersji usług, dlatego implementacja powinna dopuszczać niezależny lokalny moduł ASR/TTS zamiast zakładać obecność określonej funkcji `AudioClient`.

13. Protokół eksperymentalny i wskaźniki jakości

13.1. Układ badań i warianty

Proponowany eksperyment porównuje trzy główne warianty: VLA bez modelu świata, WMA z dedykowaną głowicą akcyjną oraz hybrydę VLA+WMA. Wszystkie warianty powinny korzystać z tego samego zestawu zadań, wspólnej definicji sukcesu i takich samych ograniczeń bezpieczeństwa. Zmienne kontrolowane obejmują konfigurację kamer, położenie obiektów, typ efektora, liczbę demonstracji, rozdzielczość, częstotliwość sterowania i dostępny budżet inferencji.

Plan pomiarowy obejmuje następujące etapy: testy unitowe formatu danych; odtwarzanie nagranych trajektorii offline; zamknięty test w symulatorze; test hardware-in-the-loop z emulowanym wyjściem napędów; eksperymenty na prawdziwym robocie pod kontrolą operatora. Dla każdego poziomu należy zdefiniować odrębne kryteria przejścia, tak aby niespełniona próba offline nie była kompensowana przez przypadkowy sukces w eksperymencie fizycznym.

13.2. Wskaźniki jakości zadania

Wskaźnik	Definicja operacyjna	Jednostka / sposób raportowania
Task success rate	Odsetek epizodów spełniających warunek końcowy	% + 95% przedział ufności
Completion time	Czas od uruchomienia zadania do terminalnego stanu	s; mediana, IQR, P95
Intervention rate	Wymagana interwencja operatora	zdarzenia / epizod
Object drop rate	Odsetek epizodów z upuszczeniem obiektu	%
Collision/constraint rate	Naruszenia reguł lub kontakt niedozwolony	liczba / godzinę i / epizod
Action rejection rate	Pakiety odrzucone przez Safety Gate	% pakietów
Recovery success	Skuteczny powrót po perturbacji	% epizodów z perturbacją
OOD performance	Wynik na nowych obiektach i warunkach	osobne wyniki dla podgrup

Dla krótkich prób szczególnie ważne są liczby bezwzględne i przedziały ufności. Wynik „100% sukcesu na kilku epizodach” nie uzasadnia deklaracji niezawodności produkcyjnej. W badaniu należy przeprowadzić losowanie kolejności wariantów i, jeśli jest to możliwe, zaślepić ocenę wideo względem stosowanego modelu.

13.3. Wskaźniki obliczeniowe i sieciowe

Raportowanie musi rozróżniać: (a) przetwarzanie wejścia, (b) obliczenie modelu, (c) transmisję w obie strony, (d) czas zatwierdzenia Safety Gate oraz (e) opóźnienie od akcji zatwierdzonej do odpowiedzi kontrolera. W praktyce należy mierzyć P50, P95 i P99, ponieważ sporadyczne przekroczenia czasu mogą stanowić większe ryzyko niż nieznacznie podwyższona mediana. Pomiar FPS inferencji nie zastępuje pomiaru maksymalnego wieku wykonywanej akcji.

Warianty na serwerze zewnętrznym wymagają testów utraty pakietów, ograniczenia przepustowości, jitteru, zerwania połączenia i restartu procesu. Telemetria powinna obejmować pamięć GPU, RAM systemowy, temperaturę, pobór mocy, czas inicjalizacji, częstotliwość wyjść modelu, liczbę kolejek oczekujących i czas ostatniego poprawnego potwierdzenia stanu robota.

13.4. Ablacje

W celu zidentyfikowania istotnych źródeł poprawy należy zaplanować odrębne eksperymenty dla jednej i wielu kamer, włączenia oraz wyłączenia propriocepcji, zmiany rozdzielczości, horyzontu akcji, odsetka wykonywanego pakietu, użycia WMA i wariantu bez WMA. Ponadto można porównać działanie z cache'em reprezentacji wizualnych i bez niego oraz generowanie akcji w sposób synchroniczny i asynchroniczny. Wyniki ablacji trzeba interpretować przy zbliżonych budżetach obliczeniowych, ponieważ wzrost częstotliwości działania może wynikać z mniejszego modelu, a nie lepszej architektury.

13.5. Próby odporności

Do testów perturbacyjnych należą: zmiana pozycji obiektu w trakcie zadania, częściowe zasłonięcie, zmiana oświetlenia, upadek obiektu, utrata jednej kamery, odchylenie IMU, opóźnienie lub zanik sieci oraz przejście człowieka w strefie pracy. Każdy scenariusz powinien mieć jawny warunek wstrzymania pracy i oczekiwane zachowanie bezpieczne. Szczególnie istotne jest rozdzielenie błędu semantyki instrukcji od błędu percepcji i błędu wykonawcy.

14. Bezpieczeństwo, niezawodność i ryzyka

14.1. Macierz ryzyka

Zagrożenie	Skutek	Zabezpieczenie	Test weryfikacyjny
Nieaktualny action chunk	Błędny ruch po zmianie sceny	TTL akcji, kontrola znaczników czasu	Opóźnienie i zatrzymanie strumienia
Błędna predykcja WMA	Nieprawidłowy plan	Twarde limity, niezależne sprawdzanie geometrii	Scenariusze OOD
Polecenie poza zakresem	Ryzykowna próba czynności	Lista umiejętności i odmowa wykonania	Test komend nieznanych
Niezgodne jednostki sterowania	Ruch z błędną skalą	Wersjonowane mapowania, testy zakresów	Wstrzyknięcie złej jednostki
Utrata łączności	Kontynuacja przestarzałego planu	Watchdog i stan HOLD	Odłączenie sieci
Fałszywe odczyty stanu	Niezauważona utrata stabilności	Walidacja sensorów, reakcja na brak danych	Symulowana awaria IMU
Kolizja dłoni z człowiekiem	Uraz lub uszkodzenie	Strefy wyłączenia, nadzór operatora, E-Stop	Testy bezkontaktowe z przeszkodą
Atak na wejście modelu	Zamrożenie lub błędna akcja	Limit czasu, sanity-check, fallback	Zmodyfikowane obrazy / instrukcje
Nieautoryzowany dostęp	Niebezpieczne zdalne polecenia	Separacja sieci, uwierzytelnianie	Audyt portów i uprawnień

W badaniach nad bezpieczeństwem VLA opisuje się m.in. możliwość wymuszenia „zamrożenia" polityki przez perturbacje wizualne. Tego rodzaju ryzyko uzasadnia niezależny watchdog, jednak skala podatności zależy od badanego modelu i scenariusza; wyników jednego benchmarku nie należy przenosić bezpośrednio na UnifoLM [17].

14.2. Ograniczenia pola pracy i kontaktu

Parametry akcji muszą zostać przefiltrowane przez dopuszczalne zbiory położeń, prędkości, przyspieszeń oraz orientacji. W zadaniach kontaktowych potrzebna jest także kontrola siły lub momentu, jeżeli interfejs

sprzętowy daje takie możliwości. Brak pomiaru siły powinien ograniczać zakres badań do miękkich, lekkich obiektów i małych prędkości. Kwestie zatrzymania awaryjnego i przywracania systemu do stanu bezpiecznego nie mogą zależeć od tego, czy model generatywny poprawnie odczyta instrukcję językową.

14.3. Niezależność zabezpieczeń

Bezpieczeństwa nie należy definiować jako promptu „nie rób nic niebezpiecznego". Ograniczenia trzeba egzekwować poza modelem, w procesie o prostszej i audytowalnej logice. W sytuacji zaniku danych, błędu sieciowego lub awarii procesu inferencji reakcja układu powinna być deterministyczna. W badaniach można mierzyć liczbę wymuszonych zatrzymań, opóźnienie reakcji oraz poprawność zachowania stanu HOLD. Normy przemysłowe mogą dostarczyć zasad projektowania, lecz eksperymentalny filtr programowy sam w sobie nie stanowi certyfikowanego systemu bezpieczeństwa.

14.4. Sim2real i ograniczenia demonstracji

Transfer z symulatora do środowiska fizycznego wymaga uwzględnienia niedokładności masy i tarcia, odkształcalności obiektów, właściwości powierzchni, rozkładu szumów czujnikowych oraz opóźnień napędów. WMA może generować poprawne wizualnie predykcje przy jednocześnie błędnym odwzorowaniu fizyki kontaktu. Z tego powodu walidację należy stopniować: od symulacji, przez ruch bez przedmiotu, po zadania z miękkimi obiektami, przy aktywnym nadzorze operatora.

15. Optymalizacja inferencji i harmonogram wykonawczy

Duże polityki VLA wymagają optymalizacji przepływu danych i zarządzania akcjami, a nie wyłącznie zmniejszania wag sieci. Rozważane techniki obejmują kwantyzację po walidacji błędu, buforowanie wyników kodowania wizualnego, redukcję liczby klatek, cache reprezentacji, ograniczenie rozdzielczości oraz asynchroniczne generowanie kolejnego pakietu przed zakończeniem bieżącego. Techniki typu **Real-Time Chunking (RTC)** próbują ograniczać pauzy między pakietami akcji; powinny być oceniane pod kątem dezaktualizacji pakietu, a nie tylko średniej częstotliwości generacji.

Wariant z WMA może stosować predykcję kilku kandydatów, co zwiększa jakość rozważań o konsekwencjach, lecz może pogarszać czas reakcji. Optymalny kompromis nie musi oznaczać równoczesnego uruchomienia obu modeli dla każdej klatki. Można uruchamiać WMA tylko przy wysokiej niepewności VLA, zmianie sceny, zbliżeniu do strefy niebezpiecznej lub przed wykonaniem działań trudno odwracalnych. Jest to **hipoteza projektowa**, której poprawność wymaga zbadania kosztu wywołań i liczby niewykrytych zagrożeń.

16. Kwestie licencyjne, etyczne i odtwarzalność

Karty modeli UnifoLM-VLA-Base i UnifoLM-WMA-0-Base deklarują w treści ograniczenia licencji **CC BY-NC-SA 4.0**, co ma znaczenie dla wykorzystania komercyjnego i redystrybucji pochodnych artefaktów. Jednocześnie publiczne metadane Hugging Face dla WMA-Base mogą wskazywać **Apache-2.0**. Ta rozbieżność oznacza, że przed użyciem konieczne jest przeczytanie faktycznego pliku `LICENSE`, dokumentacji checkpointu i warunków dostępu; nie wolno automatycznie przyjmować korzystniejszego oznaczenia w metadanych [4, 5].

Przetwarzanie obrazów z kamer humanoida wymaga kontroli prywatności osób znajdujących się w otoczeniu. Rejestrowanie strumieni do uczenia i walidacji powinno być celowe, ograniczone czasowo, zabezpieczone dostępem oraz zgodne z zakresem zgód lub inną odpowiednią podstawą prawną. W wariancie wykorzystującym serwer zewnętrzny potrzebna jest jawna polityka transmisji danych i okresów ich przechowywania. Etyka badań dotyczy także narażenia operatora i obserwatorów na ruch fizycznego urządzenia.

Odtwarzalny raport eksperymentalny powinien zawierać datę pomiarów, SHA repozytoriów, wersję firmware, zestaw sensorów, mapowania współrzędnych, pełną konfigurację modelu, liczbę epizodów

w każdym wariancie, kryteria sukcesu, surowe rozkłady opóźnień i wszystkie incydenty bezpieczeństwa. Upublicznienie wyłącznie wybranych filmów demonstracyjnych nie wystarcza do ilościowego porównania modeli.

17. Plan badań i kryteria zakończenia etapów

Etap	Zakres	Wymagany rezultat przed przejściem dalej
E0	Inwentaryzacja robota i API	Zgodna mapa efektorów, sensorów, firmware i SDK
E1	Środowiska WMA/VLA offline	Powtarzalne uruchomienie checkpointów; zapis wersji
E2	Konwersja danych i QC	Spójne sekwencje obserwacja–akcja i testy konwersji
E3	Replay i symulacja	Brak naruszeń zakresów w testach regresyjnych
E4	HIL bez ruchu fizycznego	Watchdog, E-Stop i fail-closed sprawdzone
E5	Ruch o ograniczonej energii	Zatwierdzenie operatora i ocena ryzyka zadania
E6	Porównanie VLA/WMA/hybrydy	Pełne metryki, ablacjach i przedziały ufności
E7	Reprodukowalność	Archiwum konfiguracji, logów i wersji modeli

Wyniki etapów E0–E4 powinny być uzyskane bez konieczności eksperymentowania z niezweryfikowanymi komendami fizycznego robota. Wprowadzenie do testów obiektów podatnych i delikatnych zmniejsza potencjalne skutki błędów, ale nie zastępuje oceny ryzyka i nadzoru człowieka. Wykonanie etapu E6 jest uzasadnione dopiero po ustaleniu wspólnego protokołu badań oraz wiarygodnego rejestrowania przyczyn niepowodzeń.

18. Dyskusja

WMA i VLA reprezentują dwa komplementarne sposoby operacjonalizacji uczenia zachowania. Polityki VLA są naturalne w zadaniach, w których instrukcję da się przełożyć na sekwencję sterowań na podstawie aktualnej percepcji. Modele WMA stają się interesujące, kiedy decyzja zależy od oceny odległych konsekwencji albo gdy koszt eksploracji w fizycznym środowisku jest wysoki. Jednak posiadanie modelu świata nie jest równoznaczne z poprawnym modelowaniem pełnej fizyki, a wygenerowanie dłuższego pakietu akcji nie zapewnia samodzielnie niezawodnej realizacji zadania.

Dla G1 EDU istotna jest asymetria między skalą modelu i możliwościami platformy. Udostępnione duże checkpointy oraz serwerowe przykłady producenta wskazują, że inferencję należy traktować jako osobny proces obliczeniowy. Właściwą jednostką porównania nie jest tylko liczba parametrów lub rozmiar pliku, lecz cały układ sensoryka–sieć–model–Safety Gate–kontroler. Równocześnie modele trenowane na zadaniach manipulacyjnych nie muszą bez dodatkowego uczenia generalizować na lokomocję ani na nowe efektory.

Największą luką przy porównywaniu WMA i VLA pozostaje brak wspólnego, niezależnego benchmarku przeprowadzonego na identycznym wyposażeniu, zbiorach i warunkach. Przewagę danej architektury można stwierdzić dopiero na podstawie kontrolowanego badania uwzględniającego skuteczność zadania, czas, awaryjność i koszt zasobów. Przyszłe prace powinny obejmować również metody szacowania niepewności prognoz świata oraz adaptację horyzontu akcji do poziomu ryzyka.

19. Wnioski

Architektura **UnifoLM-VLA-0** stanowi technicznie uzasadniony punkt wyjścia do badania generowania pakietów akcji warunkowanych instrukcją i obrazem na platformie Unitree G1 EDU. **UnifoLM-WMA-0** umożliwia dodatkowe badanie predykcji przyszłych interakcji, generowania danych i wspomagania selekcji działania. Oba podejścia wymagają dostosowania przestrzeni akcji do rzeczywistego efektora i wspólnej procedury synchronizacji danych.

Praktyczna droga wdrożenia przebiega od odtworzenia środowisk programowych i przygotowania danych, przez walidację symulacyjną, do nadzorowanych prób na fizycznym robocie. Najistotniejsze ograniczenia to brak deterministycznej latencji generacji, podatność na zmianę domeny, niedokładna reprezentacja kontaktu oraz możliwe niezgodności formatu i licencji artefaktów. **Niezależny Safety Gate i deterministyczny wykonawca umiejętności powinny być obowiązkowym elementem architektury**, a nie opcjonalnym modułem dodatkowym.

Na podstawie dostępnych danych nie można rozstrzygnąć, że WMA przewyższa VLA albo odwrotnie w ogólnym sensie. Można natomiast zdefiniować falsyfikowalne hipotezy, spójne wskaźniki i kompletny protokół eksperymentalny pozwalający porównać obie strategie na G1 EDU.

Bibliografia

[1] Unitree Robotics. *UnifoLM-VLA-0: A Vision–Language–Action Framework*. Repozytorium projektu, 2026. https://github.com/unitreerobotics/unifolm-vla

[2] Unitree Robotics. *UnifoLM-WMA-0: A World-Model–Action Framework*. Repozytorium projektu, 2025. https://github.com/unitreerobotics/unifolm-world-model-action

[3] Kim M. J. i in. *OpenVLA: An Open-Source Vision-Language-Action Model*. Projekt i kod. https://github.com/openvla/openvla

[4] Unitree Robotics. *UnifoLM-WMA-0-Base*. Karta modelu i warunki użycia. https://huggingface.co/unitreerobotics/UnifoLM-WMA-0-Base

[5] Unitree Robotics. *UnifoLM-VLA-Base: config.yaml*. Konfiguracja modelu. https://huggingface.co/unitreerobotics/UnifoLM-VLA-Base/blob/main/config.yaml

[6] Unitree Robotics. *UnifoLM-VLA-Base*. Artefakty modelu i karta zastosowań. https://huggingface.co/unitreerobotics/UnifoLM-VLA-Base

[7] Unitree Robotics. *Unitree G1*. Specyfikacja i rozróżnienie wariantu EDU. https://shop.unitree.com/products/unitree-g1

[8] Unitree Robotics. *unitree_sdk2*. Interfejs SDK2 i komunikacja. https://github.com/unitreerobotics/unitree_sdk2

[9] Unitree Robotics. *unitree_sdk2_python*. Implementacja Python SDK. https://github.com/unitreerobotics/unitree_sdk2_python

[10] Unitree Robotics. *UnifoLM-VLA-0 datasets*, m.in. `G1_Clean_Table`, `G1_Fold_Towel` i `G1_Wipe_Table`. https://huggingface.co/collections/unitreerobotics/unifolm-vla-0

[11] Unitree Robotics. *unitree_mujoco*. Środowisko symulacyjne i interfejsy stanu robota. https://github.com/unitreerobotics/unitree_mujoco

[12] Unitree Robotics. *unitree_lerobot*. Integracja danych i uczenia przez demonstrację. https://github.com/unitreerobotics/unitree_lerobot

[13] Hugging Face. *LeRobotDataset v3.0*. Dokumentacja formatu danych i strumieniowania. https://huggingface.co/docs/lerobot/en/lerobot-dataset-v3

[14] Chi C. i in. *Diffusion Policy: Visuomotor Policy Learning via Action Diffusion*. 2023. https://arxiv.org/abs/2303.04137

[15] Zhao T. Z. i in. *Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware*. 2023. https://arxiv.org/abs/2304.13705

[16] Brohan A. i in. *RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control*. 2023. https://arxiv.org/abs/2307.15818

[17] *FreezeVLA: Adversarial Vulnerability in Vision-Language-Action Models*. 2025. https://huggingface.co/papers/2509.19870

[18] Ha D., Schmidhuber J. *World Models*. 2018. https://arxiv.org/abs/1803.10122