
Lokalizacja drona na podstawie dzwieku

Kacper Błaszczyk 251485

17 czerwca 2025

Spis treści

1 Cel projektu	1
2 Preliminaria	1
2.1 Definicje	1
2.1.1 Uśrednianie	1
2.1.2 Norma L_p	1
2.1.3 Sprzężenie hermitowskie macierzy	2
2.2 Algorytm DAS	2
2.2.1 Dziedzina Czasu	3
2.2.2 Dziedzina Częstotliwości	3
2.3 Algorytm Ortogonalny	4
2.3.1 Modelowanie Ciśnienia i Macierzy Korelacji Krzyżowej (CSM)	4
2.3.2 Dekompozycja Wartości Własnych	4
2.4 Algorytm Clean-SC	5
2.4.1 Macierz Korelacji Krzyżowej (CSM)	5
2.4.2 Wektor Sterujący	5
2.4.3 Moc Źródła (Konwencjonalny Beamforming)	5
2.4.4 Iteracyjny Proces Oczyszczania Mapy	6
2.5 Algorytm CMF	7
2.5.1 Podstawowe definicje	7
2.5.2 Rozwiązywanie problemu i założenia	8
2.6 Algorytm RANSAC	8
2.6.1 Idea Algorytmu	8
2.6.2 Kroki algorytmu	8
2.6.3 Parametry Algorytmu	9
3 Opis projektu	11
4 Technologie i narzędzia	12
5 Implementacja	12
6 Porównanie skuteczności i wydajności	13
6.1 Miara wydajności	13
6.2 Miara skuteczności	13
6.3 Nagranie wall1	14
6.3.1 Trajektorie	14
6.3.2 Analiza błędów	15
6.3.3 Dystrybuanta	16
6.4 Nagranie wall2	17
6.4.1 Trajektorie	17
6.4.2 Analiza błędów	18
6.4.3 Dystrybuanta	19
6.5 Nagranie wall3	20

6.5.1	Trajektorie	20
6.5.2	Analiza błędów	21
6.5.3	Dystrybuanta	22
7	Podsumowanie	23
7.1	Wnioski	23
7.2	Dystrybuanty	23
	Bibliografia	24

1 Cel projektu

Projekt polegał na implementacji, uruchomieniu i wykonaniu eksperymentów polegających na porównaniu skuteczności i wydajności lokalizacji drona przy użyciu algorytmów: DAS, Ortogonalny, Clean-SC, CMF. Implementacja została oparta o bibliotekę Acoular.

2 Preliminaria

2.1 Definicje

2.1.1 Uśrednianie

Dla wektora sygnału $\xi(t)$, macierz korelacji krzyżowej $\langle \xi \xi^\dagger \rangle$ jest obliczana jako:

$$\langle \xi \xi^\dagger \rangle = \frac{1}{T} \int_0^T \xi(t) \xi^\dagger(t) dt, \quad (1)$$

gdzie:

- $\xi(t)$ – wektor sygnału w czasie t ,
- $\xi^\dagger(t)$ – sprzężenie hermitowskie wektora $\xi(t)$,
- T – całkowity czas obserwacji,

Dla sygnałów dyskretnych uśrednianie w czasie przyjmuje postać sumy:

$$\langle \xi \xi^\dagger \rangle = \frac{1}{N} \sum_{n=1}^N \xi[n] \xi^\dagger[n], \quad (2)$$

gdzie:

- $\xi[n]$ – wektor sygnału w n -tej próbce czasowej
- $\xi^\dagger[n]$ – sprzężenie hermitowskie wektora $\xi[n]$
- N – liczba próbek w czasie

2.1.2 Norma L_p

Dla wektora $\mathbf{x} = [x_1 \ x_2 \ \dots \ x_n]^T \in \mathbb{R}^n$ norma L_p ($p \in \mathbb{R} \wedge p \geq 1$) jest definiowana jako:

$$\|\mathbf{x}\|_p = \left(\sum_{i=1}^n |x_i|^p \right)^{\frac{1}{p}} \quad (3)$$

W szczególności:

- Norma L_1 (norma Manhattan):

$$\|\mathbf{x}\|_1 = \sum_{i=1}^n |x_i|$$

- Norma L_2 (norma Euklidesowa):

$$\|\mathbf{x}\|_2 = \left(\sum_{i=1}^n x_i^2 \right)^{\frac{1}{2}}$$

- Norma L_∞ (norma maksimum, Czebyszewa):

$$\|\mathbf{x}\|_\infty = \max_i |x_i|$$

2.1.3 Sprzężenie hermitowskie macierzy

Dla macierzy $\mathbb{M}_{n \times k}(\mathbb{C})$ złożenie operacji transpozycji i sprzężenia zespolonego:

$$A^\dagger = \overline{A^T} \quad (4)$$

$$(A^\dagger)_{ij} = \overline{A_{ji}}$$

2.2 Algorytm DAS

Algorytm Delay and Sum (DAS), znany również jako konwencjonalny beamforming, stanowi podstawową i pierwotną technikę beamformingu. Jest to metoda przetwarzania danych służąca do przestrzennego filtrowania sygnałów pochodzących z macierzy czujników. Głównym celem DAS, w kontekście tego opracowania, jest lokalizacja źródeł dźwięku.

DAS opiera się na zależności między źródłem a odbiornikiem. Algorytm ten dąży do wykrycia opóźnienia fazowego poprzez wirtualne sterowanie macierzy w określonym kierunku lub punkcie w przestrzeni.

Proces polega na:

1. Przyjęciu siatki N_{points} punktów kontrolnych (\vec{x}_p), w których potencjalnie mogą znajdować się źródła dźwięku, oraz dyskretnego zestawu M mikrofonów (\vec{x}_m), które próbują pole dźwiękowe.
2. Propagacji wstępnej każdego sygnału czasowego zarejestrowanego przez mikrofon do każdego punktu kontrolnego. Uwzględnia się przy tym wzelną pozycję punktu kontrolnego i mikrofonu oraz prędkość propagacji fali akustycznej w danym medium. Te dwa czynniki dostarczają informacji o opóźnieniu czasowym (lub przesunięciu fazowym), które należy uwzględnić podczas wirtualnego ogniskowania macierzy na określony punkt kontrolny.

- Zsumowaniu opóźnionych danych w celu uzyskania wyjścia beamformera dla danego punktu kontrolnego. Jeśli źródło znajduje się w punkcie kontrolnym, opóźnione sygnały są w fazie, a wynik beamformera jest maksymalizowany. Jeśli macierz ogniskowana jest na punkt odległy od rzeczywistego źródła, sygnały nie są w fazie, co skutkuje niższym wynikiem.

W praktyce, punkty kontrolne są skanowane, a wartość wyjścia beamformera ($\text{bf}(\vec{x}_p, t)$) jest obliczana i rejestrowana dla każdego z nich. W ten sposób tworzona jest mapa amplitud, a pozycja źródła jest szacowana na podstawie punktów kontrolnych, które dają maksymalne wartości $\text{bf}(\vec{x}_p, t)$. Opóźnienia umożliwiają konstruktywną interferencję z określonych kątów i destrukcyjną interferencję z innych.

2.2.1 Dziedzina Czasu

W dziedzinie czasu wyjście beamformera $\text{bf}(\vec{x}_p, t)$ w p -tym punkcie kontrolnym w t -tej chwili czasowej można obliczyć za pomocą następującego wzoru:

$$\text{bf}(\vec{x}_p, t) = \frac{1}{M} \sum_{m=1}^M w_m A_m(\vec{x}_p, \vec{x}_m) p_m \left(t - \frac{|\vec{x}_p - \vec{x}_m|}{c} \right) \quad (5)$$

gdzie:

- p_m : - sygnał ciśnienia zarejestrowany przez m -ty mikrofon.
- w_m - współczynnik wagowy (ang. weighting factor) zastosowany do pojedynczego m -tego mikrofonu, używany do optymalizacji ogólnego wyniku. Konwencjonalne algorytmy beamformingu mają stałe wagi czujników.
- $A_m(\vec{x}_p, \vec{x}_m)$ - współczynnik skali, który może uwzględniać redukcję amplitudy, np. $A_m(\vec{x}_p, \vec{x}_m) = \frac{1}{4\pi k |\vec{x}_p - \vec{x}_m|}$. Współczynnik ten bierze pod uwagę tłumienie amplitudy fali z powodu odległości między źródłem a mikrofonem.
- c - prędkość propagacji fali akustycznej w danym medium.
- \vec{x}_p - położenie p -tego punktu kontrolnego.
- \vec{x}_m - położenie m -tego mikrofonu.

Ciśnienia akustyczne mierzone w każdej lokalizacji mikrofonu \vec{x}_m są ważone, skalowane i opóźniane zgodnie z względową odlegością między każdym mikrofonem a rozważanym p -tym punktem kontrolnym, a następnie sumowane, co daje wynik beamformera, stąd nazwa Delay & Sum Beamforming (Opóźnienie i Sumowanie).

2.2.2 Dziedzina Częstotliwości

Równanie beamformingu w dziedzinie czasu można również przenieść na dziedzinę częstotliwości. Transformatę Fouriera sygnału ciśnienia $p_m(t)$ dla m -tego mikrofonu w dziedzinie częstotliwości oznacza się jako $P_m(\omega_k)$. W dziedzinie częstotliwości, DAS można zapisać jako:

$$\text{BF}(\vec{x}_p, \omega_k) = \frac{1}{M} \sum_{m=1}^M w_m A_m(\vec{x}_p, \vec{x}_m) P_m(\omega_k) e^{j\omega_k \Delta_{p,m}} \quad (6)$$

gdzie:

- $P_m(\omega_k)$ - sygnał ciśnienia m -tego mikrofonu przy częstotliwości kątowej ω_k .
- $e^{j\omega_k \Delta_{p,m}}$ - przesunięcie fazowe używane do sterowania macierzy w kierunku \vec{x}_p .
- $\Delta_{p,m} = \frac{|\vec{x}_p - \vec{x}_m|}{c}$ - opóźnienie czasowe/przesunięcie fazowe dla m -tego mikrofonu względem punktu kontrolnego \vec{x}_p .

2.3 Algorytm Ortogonalny

Główna idea polega na dekompozycji pola dźwiękowego na ortogonalne składowe, z których każda odpowiada innemu mechanizmowi źródłowemu. Odbywa się to poprzez analizę macierzy korelacji krzyżowej (CSM) sygnałów zarejestrowanych przez mikrofony. Wyniki algorytmu konwencjonalnego beamformingu (CB) mogą być post-przetwarzane, aby uzyskać „wyczyszczoną” mapę akustyczną, pozbawioną fałszywych źródeł, pozostawiając tylko te rzeczywiste.

2.3.1 Modelowanie Ciśnienia i Macierzy Korelacji Krzyżowej (CSM)

Ciśnienie p_m mierzone przez m -ty mikrofon można modelować jako sumę P ortogonalnych mechanizmów źródłowych, gdzie f_{mj} to czynniki zależne od lokalizacji źródła i mikrofonu, a q_j to siła j -tego źródła:

$$p_m = \sum_{j=1}^P f_{mj} q_j \quad (7)$$

Warunek ortogonalności powoduje, że macierz korelacji krzyżowej $C_{mm'}$ (CSM) może być przepisana jako suma macierzy $C_{mm',j}$, gdzie każda z nich odpowiada pojedynczemu j -temu mechanizmowi źródłowemu:

$$C_{mm'} = \sum_{j=1}^P C_{mm',j} \quad (8)$$

Dzięki temu każdej j -taj składowej źródłowej można mapować oddzielnie, zastępując $C_{mm'}$ w równaniu konwencjonalnego beamformingu (które ma postać $BF_{cb} = \frac{g^\dagger C g}{\|g\|^4}$) przez $C_{mm',j}$.

2.3.2 Dekompozycja Wartości Własnych

Obliczanie każdej ortogonalnej składowej poprzez wykorzystanie dekompozycji wartości własnych macierzy korelacji krzyżowej (CSM). CSM jest macierzą hermitowską i dodatnio półokreślona. Wektory własne są ortogonalne, a wartości własne, które są dodatnie i rzeczywiste, aproksymują siły źródeł. Składową $C_{mm',j}$ można wyrazić za pomocą wektora własnego v_j i wartości własnej λ_j w następujący sposób:

$$C_{mm',j} = v_j \lambda_j v_j^\dagger \quad (9)$$

gdzie v_j^\dagger oznacza sprzężenie hermitowskie wektora własnego.

2.4 Algorytm Clean-SC

Działanie CLEAN-SC polega na iteracyjnym usuwaniu części mapy akustycznej, która jest przestrzennie spójna z wykrytym głównym źródłem dźwięku. Proces ten poprawia rozdzielczość przestrzenną i zakres dynamiczny, co pozwala na odkrywanie wtórnego źródła hałasu, które mogłyby być zamaskowane przez płaty boczne. Proces zaczyna się od wstępnej mapy akustycznej uzyskanej konwencjonalnym beamformingiem, a następnie iteracyjnie "czyści" mapę, usuwając przyczyniające się do niej artefakty.

2.4.1 Macierz Korelacji Krzyżowej (CSM)

Punktem wyjścia dla algorytmu CLEAN-SC jest macierz korelacji krzyżowej (CSM) sygnałów akustycznych, mierzone przez mikrofony. Zakłada się, że całkowita macierz CSM jest sumą nieskorelowanych źródeł:

$$C = \sum_{k=1}^K p_k p_k^\dagger \quad (10)$$

gdzie K jest liczbą źródeł, a p_k to N -wymiarowy wektor zawierający transformaty Fouriera sygnałów z N mikrofonów dla danej częstotliwości f . Powyższe równanie jest prawdziwe przy założeniu braku dekoherencji sygnałów z tego samego źródła między różnymi mikrofonami oraz braku dodatkowego niespójnego szumu. W praktyce stosuje się również "przyjętą" macierz CSM, gdzie elementy na głównej przekątnej są zerowane w celu redukcji szumów własnych mikrofonów, takich jak szum wiatru czy szum elektroniczny, które nie zawierają informacji fazowej:

$$\bar{C}_{m,n} = \begin{cases} C_{m,n} & \text{jeśli } m \neq n \\ 0 & \text{jeśli } m = n \end{cases} \quad \text{dla } m, m' = 1 \dots M \quad (11)$$

2.4.2 Wektor Sterujący

Wektor sterujący $g_n(\xi)$ dla n -tego mikrofonu, odnoszący się do źródła w pozycji ξ , jest dany jako:

$$g_n(\xi) = -\frac{1}{4\pi\|\vec{x}_n - \xi\|} \exp\left(-2\pi i f \frac{\|\vec{x}_n - \xi\|}{c}\right) \quad (12)$$

gdzie f jest częstotliwością, c jest prędkością dźwięku, \vec{x}_n jest pozycją n -tego mikrofonu, a ξ jest lokalizacją źródła. Idealnie, wektory sterujące g_k są proporcjonalne do wektorów źródłowych $p_k = a_k g_k$, gdzie a_k są zespolonymi amplitudami źródeł.

2.4.3 Moc Źródła (Konwencjonalny Beamforming)

Moc źródła A dla danej pozycji jest szacowana za pomocą konwencjonalnego beamformingu. Normalizowany wektor sterujący w jest definiowany jako $w = \frac{g}{(\sum_{(m,n) \in S} \|g_m\|^2 \|g_n\|^2)^{1/2}}$. Moc źródła A może być wówczas wyrażona jako:

$$A = w^\dagger \bar{C} w \quad (13)$$

gdzie w^\dagger oznacza sprzężone hermitowskie wektora w .

2.4.4 Iteracyjny Proces Oczyszczania Mapy

Algorytm CLEAN-SC wykorzystuje iteracyjny proces do oczyszczania mapy akustycznej. Kluczową cechą jest to, że płaty boczne są przestrzennie spójne z głównymi źródłami, co pozwala na ich iteracyjne usuwanie. Główne kroki algorytmu to:

- 1. Generowanie początkowej "brudnej" mapy:** Na początku procesu definiuje się zdegradowaną macierz CSM $D^{(i)}$, gdzie $D^{(0)} = C$ (całkowita macierz CSM). Dla każdej lokalizacji j , początkowa "brudna mapa" $P_j^{(0)}$ jest obliczana jako:

$$P_j^{(0)} = w_j^\dagger \bar{D}^{(0)} w_j = w_j^\dagger \bar{C} w_j \quad (14)$$

- 2. Wyszukiwanie piku:** W każdej iteracji (i), algorytm lokalizuje pozycję piku $\xi_{\max}^{(i)}$ w bieżącej brudnej mapie $P_j^{(i)}$.
- 3. Aktualizacja mapy (subtrakcja):** Zdegradowana macierz CSM jest aktualizowana, a następnie nowa mapa jest obliczana poprzez odjęcie "PSF-opodobnego" członu od bieżącej mapy. W literaturze, choć CLEAN-SC zazwyczaj nie używa "syntetycznych" PSF, to jednak w oryginalnych formułach do odejmowania stosuje się termin podobny do funkcji rozmycia punktu, dynamicznie wyznaczany na podstawie wektora sterującego zidentyfikowanego piku:

$$D^{(i+1)} = D^{(i)} - P_{\max}^{(i)} g_{\max}^{(i)} g_{\max}^{(i)\dagger} \quad (15)$$

Następnie mapa akustyczna jest aktualizowana:

$$P_j^{(i+1)} = P_j^{(i)} - P_{\max}^{(i)} w_j^\dagger \bar{D}^{(i)} [g_{\max}^{(i)} g_{\max}^{(i)\dagger}] w_j \quad (16)$$

(Należy zauważyć, że $w_j^\dagger \bar{D}^{(i)} [g_{\max}^{(i)} g_{\max}^{(i)\dagger}] w_j$ stanowi tu iloczyn formy kwadratowej, który de facto opisuje, jak źródło o charakterystyce $g_{\max}^{(i)}$ "rozmywa" się w mapie wygenerowanej z $D^{(i)}$).

- 4. Dodawanie "czystego" piku:** Jednocześnie z odejmowaniem artefaktów, do "czystej" mapy $Q_j^{(i)}$ dodawany jest "czysty pik" w miejscu zidentyfikowanego źródła:

$$Q_j^{(i)} = P_{\max}^{(i)} \Phi(\xi_j - \xi_{\max}) \quad (17)$$

gdzie Φ to funkcja "czystego piku", zazwyczaj funkcja Diraca, z $\Phi(0) = 1$.

- 5. Warunek stopu:** Proces iteracyjny jest kontynuowany, dopóki nie zostanie spełniony warunek stopu. Typowym warunkiem jest, gdy norma (lub "moc") zdegradowanej macierzy CSM przestaje się zmniejszać, co oznacza, że dalsze odejmowanie artefaktów przestaje być efektywne:

$$\|\bar{D}^{(I+1)}\| \geq \|\bar{D}^{(I)}\| \quad (18)$$

gdzie I jest końcową liczbą iteracji.

6. **Końcowa oczyszczona mapa:** Po zakończeniu iteracji, oczyszczona mapa akustyczna A_j jest sumą wszystkich "czystych pików" i ostatniej (potencjalnie wciąż "brudnej") mapy:

$$A_j = \sum_{i=0}^{I-1} Q_j^{(i)} + P_j^{(I)} \quad (19)$$

Oczekuje się, że ta mapa zawiera jedynie rzeczywiste źródła.

2.5 Algorytm CMF

Algorytm Covariance Matrix Fitting (CMF) jest podejściem stosowanym w dziedzinie beamformingu akustycznego do rozwiązywania problemu odwrotnego lokalizacji źródeł hałasu. W ramach problemu odwrotnego, celem jest określenie rozkładu źródeł akustycznych na podstawie pomiarów ciśnienia akustycznego zebranych przez zestaw mikrofonów. CMF ma na celu znalezienie kombinacji źródeł, która najlepiej pasuje do zmierzonej macierzy korelacji krzyżowej (CSM) sygnałów akustycznych. Jest to metoda, która dąży do uwzględnienia wszystkich źródeł jednocześnie, co jest cechą charakterystyczną metod odwrotnych.

2.5.1 Podstawowe definicje

Punktem wyjścia jest związek między ciśnieniem mierzone przez mikrofony a siłą źródeł. Zakłada się, że ciśnienie p mierzone przez M mikrofonów jest wynikiem superpozycji sygnałów z S źródeł, gdzie p jest wektorem ciśnień mikrofonowych, a q jest wektorem sił źródeł. Zależność tę można opisać za pomocą macierzy radiacyjnej G :

$$p = Gq \quad (20)$$

gdzie G jest macierzą o wymiarach $M \times S$.

Macierz korelacji krzyżowej (CSM) dla ciśnień mikrofonowych, P , jest definiowana jako:

$$P = \langle pp^\dagger \rangle \quad (21)$$

gdzie $\langle \cdot \rangle$ oznacza uśrednianie w czasie, a p^\dagger oznacza sprzężone transponowanie. Macierz P ma wymiary $M \times M$.

Podobnie, macierz korelacji krzyżowej dla sił źródeł, Q , jest definiowana jako:

$$Q = \langle qq^\dagger \rangle \quad (22)$$

gdzie Q ma wymiary $S \times S$.

Podstawowe równanie problemu odwrotnego, wyrażone w formie kwadratowej macierzy CSM, jest następujące:

$$P = GQG^\dagger \quad (23)$$

Celem algorytmu CMF jest rozwiązanie tego równania w celu oszacowania Q (macierzy sił źródeł) na podstawie zmierzonej macierzy P (macierzy ciśnień mikrofonowych) i znanej macierzy G (macierzy radiacyjnej).

2.5.2 Rozwiążanie problemu i założenia

Rozwiążanie problemu odwrotnego, tj. estymacja macierzy Q , jest często przedstawiane w ogólnej formie jako:

$$\hat{Q} = HPH^\dagger \quad (24)$$

gdzie H jest operatorem odwrotnym.

Kluczowym założeniem w algorytmie CMF, które znacznie upraszcza problem, jest założenie o **nieskorelowanych źródłach**. Oznacza to, że macierz mocy źródeł Q staje się macierzą diagonalną, co pozwala na przekształcenie kwadratowej formy problemu akustycznego w standardowy układ liniowy, dla którego można uzyskać rozwiązanie w postaci rzeczywistej.

2.6 Algorytm RANSAC

2.6.1 Idea Algorytmu

RANSAC zaczyna od możliwie najmniejszego początkowego zbioru danych i rozszerza go o spójne dane. Podstawowym założeniem jest, że dane składają się z "inlierów" (danych, których rozkład może być wyjaśniony przez dany zestaw parametrów modelu, choć mogą być obciążone szumem) oraz "outlierów" (danych, które nie pasują do modelu). Outliery mogą pochodzić z ekstremalnych wartości szumu, błędnych pomiarów lub niepoprawnych hipotez.

2.6.2 Kroki algorytmu

Algorytm RANSAC jest algorytmem iteracyjnym, która powtarza dwa główne kroki:

1. Generowanie hipotezy (Wybór próbki i dopasowanie modelu):

- Losowo wybierany jest minimalny podzbiór danych obserwacyjnych $S_k \subset U$, o rozmiarze $|S_k| = m$. Minimalna liczba próbek m jest równa złożoności modelu geometrycznego, tj. minimalnej liczbie danych potrzebnych do obliczenia rozwiązania.
- Na podstawie tego podzbioru obliczane są parametry modelu $p_k = f(S_k)$.

2. Weryfikacja (Sprawdzenie spójności i wyznaczenie zbioru zgodności):

- Obliczana jest wartość kosztu $C_k = \sum_{x \in U} \rho(p_k, x)$ dla wszystkich punktów danych w zbiorze U , względem obliczonych parametrów p_k . Funkcja kosztu $\rho(p, x)$ mierzy dopasowanie pojedynczego punktu danych x do modelu z parametrami p .
- Punkty danych, które są spójne z modelem w ramach określonego progu błędu (tolerancji), tworzą tzw. zbiór konsensusu (consensus set).
- Jeśli rozmiar zbioru konsensusu (np. liczba inlierów) jest większy niż zadany próg d , lub jeśli C_k jest lepsze niż dotychczasowe C^* (maksymalizacja funkcji kosztu), parametry p_k zostają uznane za najlepsze dotychczasowe i zapisane jako p^* .
- Model może być następnie ulepszony poprzez ponowną estymację parametrów, wykorzystując wszystkich członków nowo znalezionego zbioru konsensusu.

3. Kryterium zakończenia: Procedura jest powtarzana, aż prawdopodobieństwo znalezienia lepszego rozwiązania spadnie poniżej zdefiniowanego progu η , lub po ustalonej z góry maksymalnej liczbie iteracji k_{max} . Jeśli po pewnej liczbie prób nie zostanie znaleziony zbiór konsensusu o wystarczającej liczbie członków, algorytm zwraca model z największym dotychczas znalezionym zbiorem konsensusu lub kończy się niepowodzeniem.

Funkcja kosztu ρ : Dla przykładu detekcji linii, funkcja błędu ρ jest odległością (Euklidową) punktu od linii.

2.6.3 Parametry Algorytmu

Algorytm RANSAC wymaga zdefiniowania kilku kluczowych parametrów, które mają wpływ na jego działanie, skuteczność i prawdopodobieństwo znalezienia poprawnego modelu.

- **Minimalna liczba punktów danych (n):** Jest to minimalna liczba punktów danych wymagana do estymacji parametrów modelu. Kardynalność podpróbki (czyli liczba danych w tej podpróbce) musi być wystarczająca do wyznaczenia parametrów modelu. Na przykład, jeśli celem jest dopasowanie linii w dwóch wymiarach, minimalna liczba punktów wynosi dwa, ponieważ linia jest jednoznacznie określona przez dwa punkty.
- **Maksymalna liczba iteracji (k):** Określa maksymalną liczbę iteracji dozwolonych w algorytmie. Algorytm RANSAC jest niedeterministyczny, co oznacza, że daje rozsądny wynik tylko z pewnym prawdopodobieństwem, które wzrasta wraz z liczbą iteracji. Liczba iteracji k może być w przybliżeniu określona jako funkcja pożądanego prawdopodobieństwa sukcesu (p).

Niech p będzie pożdanym prawdopodobieństwem, że algorytm RANSAC dostarczy co najmniej jeden użyteczny wynik. W uproszczeniu, RANSAC zwraca pomyślny wynik, jeśli w którejś iteracji wybierze wyłącznie inliery z wejściowego zbioru danych, gdy wybiera n punktów do estymacji parametrów modelu.

Niech ε będzie prawdopodobieństwem wybrania inliera za każdym razem, gdy wybierany jest pojedynczy punkt danych:

$$\varepsilon = \frac{\text{liczba inlierów w danych}}{\text{liczba punktów w danych}} \quad (25)$$

Prawdopodobieństwo, że wszystkie n punktów wybranych do estymacji modelu są inlierami, wynosi ε^n . Prawdopodobieństwo, że co najmniej jeden z n punktów jest elementem odstającym, wynosi $1 - \varepsilon^n$.

Prawdopodobieństwo, że algorytm nigdy nie wybierze zbioru n punktów, które są wyłącznie inlierami, po k iteracjach wynosi $(1 - \varepsilon^n)^k$. Jest to równoważne z $1 - p$ (prawdopodobieństwem, że algorytm nie zakończy się sukcesem) w skrajnym przypadku. Stąd, aby osiągnąć pożądane prawdopodobieństwo sukcesu p , liczba iteracji k musi spełniać:

$$1 - p \leq (1 - \varepsilon^n)^k \quad (26)$$

Co po zlogarytmowaniu obu stron prowadzi do wzoru na k :

$$k \geq \frac{\log(1 - p)}{\log(1 - \varepsilon^n)} \quad (27)$$

Należy zauważyć, że ten wzór zakłada niezależny wybór punktów, co oznacza, że wybrany punkt może być ponownie wybrany w tej samej iteracji.

- **Próg błędu (t):** Jest to wartość progowa używana do określenia, czy punkty danych są dobrze dopasowane do modelu i są uznawane za inliery. Element danych zostanie uznany za element odstający, jeśli nie pasuje do modelu w ramach tego progu błędu, który definiuje maksymalne odchylenie danych dla inlierów.
- **Wymagana liczba punktów zgodnych (d):** Jest to liczba bliskich punktów danych (inlierów) wymaganych do stwierdzenia, że model dobrze pasuje do danych. Zbiór inlierów uzyskanych dla dopasowanego modelu nazywany jest zbiorem konsensusu (consensus set). Jeśli liczba elementów w zbiorze konsensusu jest większa niż d , algorytm uznaje, że znaleziono odpowiedni model.

3 Opis projektu

Poniżej przedstawiono kluczowe kroki realizacji projektu:

1. Przetwarzanie sygnału dźwiękowego:

- Wczytanie plików dźwiękowych w formacie WAV (nagrania wall1, wall2, wall3) o częstotliwości próbkowania 96 kHz.
- Podział danych dźwiękowych na ramki o długości odpowiadającej 30 klatkom na sekundę (FPS), co umożliwiło analizę w czasie rzeczywistym.
- Zdefiniowanie geometrii mikrofonów na podstawie pliku XML oraz siatki prostokątnej o wymiarach od -2 do $+2$ m w osiach x i y , z krokiem 0.1 m, na wysokości $z = 1$ m.

2. Zastosowanie algorytmów beamformingu:

- Obliczenie spektrum mocy (PowerSpectra) dla każdej ramki sygnału z użyciem okna Hanninga, rozmiaru bloku 128 próbek i 50% nakładania.
- Zastosowanie wybranego algorytmu beamformingu (DAS, Functional Beamforming, MUSIC, Capon) do wyznaczenia mapy akustycznej.
- Określenie punktu maksymalnej amplitudy na siatce, reprezentującego szacowaną pozycję źródła dźwięku (drona), oraz utworzenie powiększonej siatki ogniskowania (RectGrid) o rozmiarze ± 0.2 m wokół tego punktu.

3. Generowanie wizualizacji map akustycznych:

- Wykorzystanie biblioteki matplotlib do tworzenia animacji map akustycznych dla każdej ramki, zapisanych jako pliki MP4 (ffmpeg).
- Oznaczenie na mapach szacowanej pozycji drona oraz wyświetlenie współrzędnych punktu maksymalnej amplitudy.
- Zapisanie współrzędnych punktów ogniskowania i szacowanych pozycji drona do plików .npy dla dalszej analizy.

4. Korekta trajektorii za pomocą transformacji afonicznej:

- Wczytanie referencyjnych trajektorii drona oraz szacowanych punktów ogniskowania dla każdego algorytmu.
- Zastosowanie algorytmu RANSAC do wyznaczenia macierzy transformacji afonicznej, mapującej szacowane punkty na trajektorie referencyjne.
- Przekształcenie szacowanych punktów za pomocą wyznaczonej macierzy, a następnie zastosowanie filtrów: ograniczenie zakresu do rozdzielczości obrazu oraz eliminacja wartości odstających.

5. Obliczenie błędu lokalizacji:

- Obliczenie błędu lokalizacji jako normy L_∞ między przekształconymi punktami ogniskowania a referencyjną trajektorią dla każdego nagrania i algorytmu.
- Zapisanie błędów do plików .npy oraz obliczenie statystyk: średniej, mediany, odchylenia standardowego i RMSE.

6. Wizualizacja wyników:

- Wygenerowanie wykresów rozkładu błędów oraz różnic między kolejnymi błędami dla każdego algorytmu i nagrania.
- Utworzenie porównawczych wykresów trajektorii szacowanych i referencyjnych w formacie SVG, z użyciem różnych map kolorów dla wizualizacji czasowej.
- Sporządzenie dystrybuant błędów (CED) dla każdego algorytmu indywidualnie oraz dla wszystkich algorytmów dla każdego nagrania, zapisując je jako pliki SVG.

7. Porównanie skuteczności i wydajności:

- Pomiar czasu przetwarzania dla każdej ramki i całkowitego czasu dla każdego nagrania, zapisanie wyników w tabelach (np. Tabela 1–3).
- Ocena skuteczności na podstawie rozkładu błędów i dystrybuant, co umożliwiło porównanie precyzji lokalizacji między algorytmami.

4 Technologie i narzędzia

Python 3.12.4

Biblioteki:

- Acoular 25.3.post1 [1]
- Numpy 2.1.3 [7]
- Scipy 1.15.2 [8]
- Matplotlib 3.10.1 [6]

MATLAB 24.2.0.2863752 Ground Truth Labeler [5]

ffmpeg 7.1.1 [3]

Audacity 2.4.2 [2]

Obliczenia zostały przeprowadzone na maszynie z procesorem Intel® Core™ i5-9300H [4].

5 Implementacja

Utworzono repozytorium kodu na platformie GitHub:

<https://github.com/zasilacz-szklanki/DroneLocalizationProject>

6 Porównanie skuteczności i wydajności

6.1 Miara wydajności

Wydajność mierzona jest przez rzeczywisty czas życia procesu powstałego na skutek uruchomienia skryptu w języku Python.

6.2 Miara skuteczności

Skuteczność jest mierzona poprzez różnicę odległości punktów zaznaczonych ręcznie od punktów wyznaczonych przez program (norma L_∞).

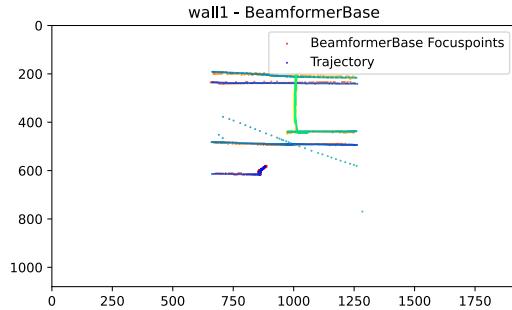
6.3 Nagranie wall1

Liczba ramek = 2411

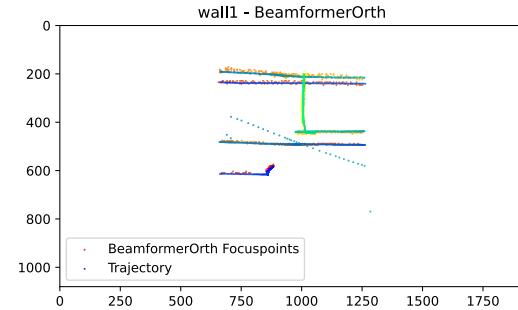
Algorytm	Całkowity czas dla ramek [s]	Średni czas dla ramki [s]	Maksymalny czas dla ramki [s]	Minimalny czas dla ramki [s]
DAS	60.1819	0.0249	0.1758	0.0181
Ortogonalny	62.1833	0.0257	0.1120	0.0199
Clean-SC	588.0144	0.2437	0.3855	0.2154
CMF	5763.4069	2.3894	3.4873	1.5540

Tabela 1: Podsumowanie czasu przetwarzania nagrania wall1

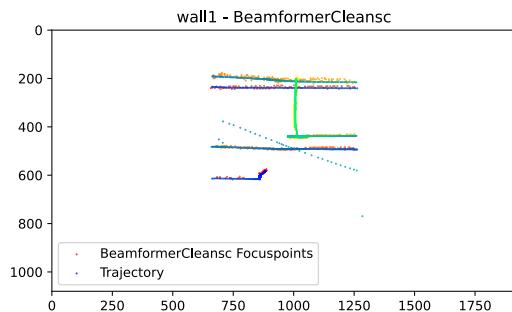
6.3.1 Trajektorie



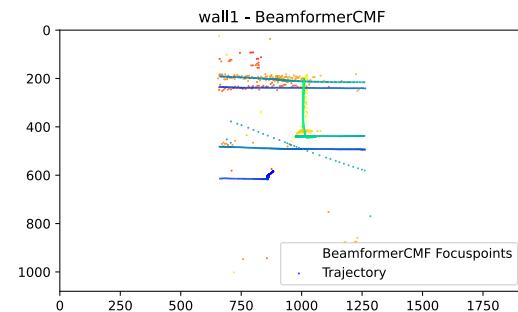
Rysunek 1: Trajektoria lotu drona dla algorytmu DAS dla nagrania wall1



Rysunek 2: Trajektoria lotu drona dla algorytmu Ortogonalnego dla nagrania wall1

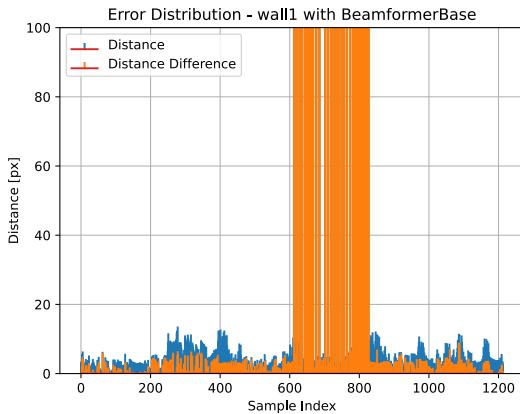


Rysunek 3: Trajektoria lotu drona dla algorytmu Clean-SC dla nagrania wall1

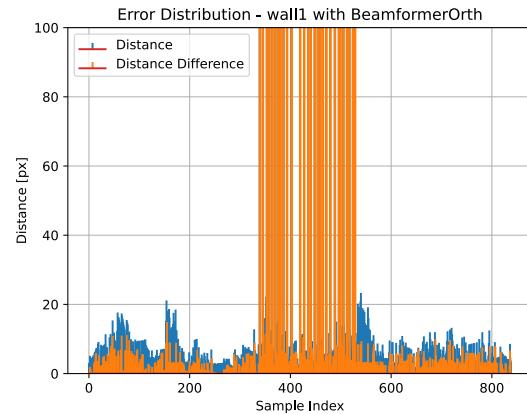


Rysunek 4: Trajektoria lotu drona dla algorytmu CMF dla nagrania wall1

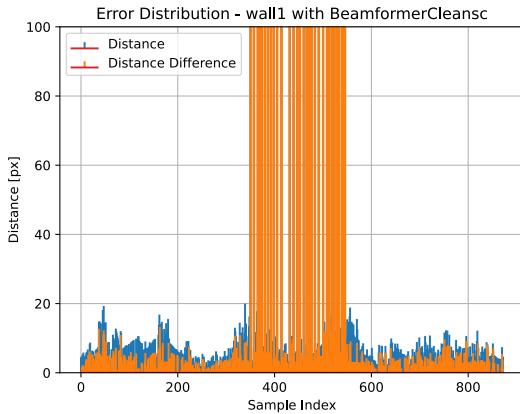
6.3.2 Analiza błędów



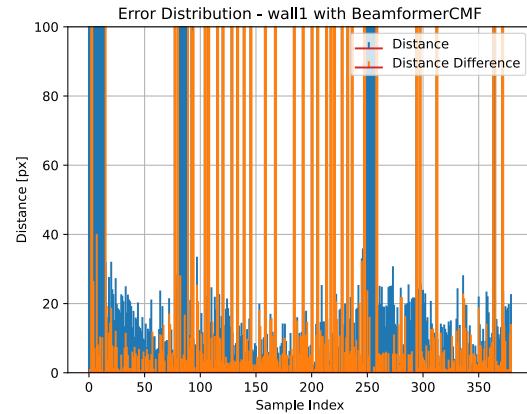
Rysunek 5: Rozkład błędu dla algorytmu DAS dla nagrania wall1



Rysunek 6: Rozkład błędu dla algorytmu Ortogonalnego dla nagrania wall1

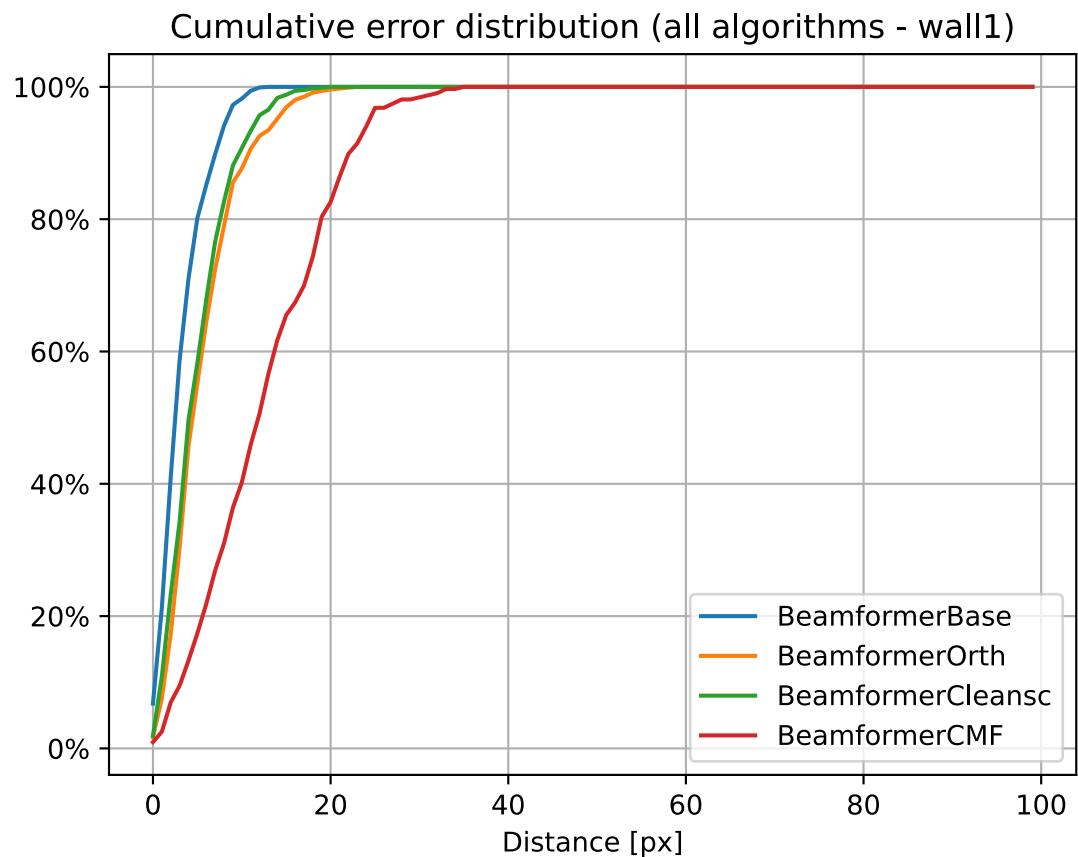


Rysunek 7: Rozkład błędu dla algorytmu Clean-SC dla nagrania wall1



Rysunek 8: Rozkład błędu dla algorytmu CMF dla nagrania wall1

6.3.3 Dystrybuanta



Rysunek 9: Dystrybuanta rozkładu błędu dla wszystkich algorytmów dla nagrania wall1

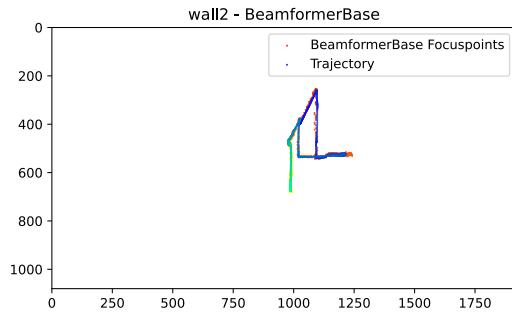
6.4 Nagranie wall2

Liczba ramek = 1252

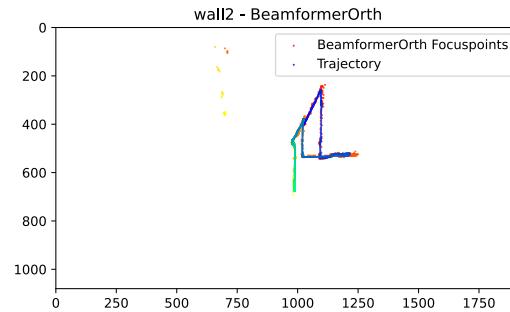
Algorytm	Całkowity czas dla ramek [s]	Średni czas dla ramki [s]	Maksymalny czas dla ramki [s]	Minimalny czas dla ramki [s]
DAS	28.8960	0.0230	0.1191	0.0181
Ortogonalny	31.9241	0.0254	0.1066	0.0209
Clean-SC	315.9844	0.2521	0.5653	0.2043
CMF	3045.7364	2.4307	3.8712	1.6587

Tabela 2: Podsumowanie czasu przetwarzania nagrania wall2

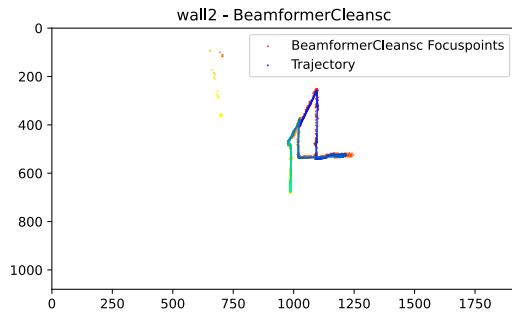
6.4.1 Trajektorie



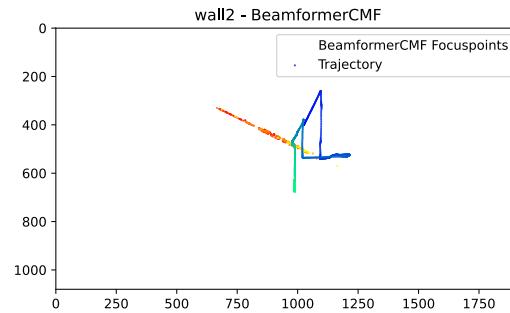
Rysunek 10: Trajektoria lotu drona dla algorytmu DAS dla nagrania wall2



Rysunek 11: Trajektoria lotu drona dla algorytmu Ortogonalnego dla nagrania wall2

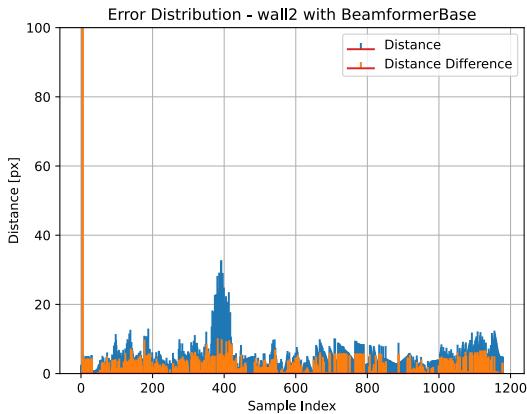


Rysunek 12: Trajektoria lotu drona dla algorytmu Clean-SC dla nagrania wall2

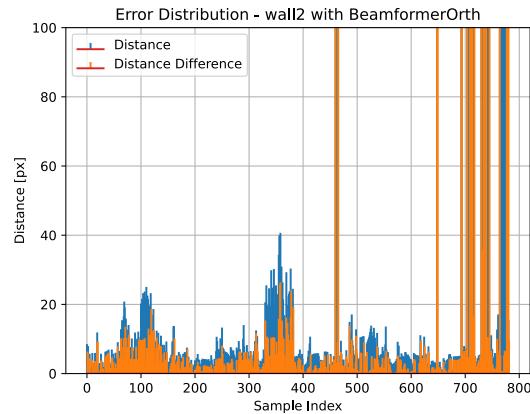


Rysunek 13: Trajektoria lotu drona dla algorytmu CMF dla nagrania wall2

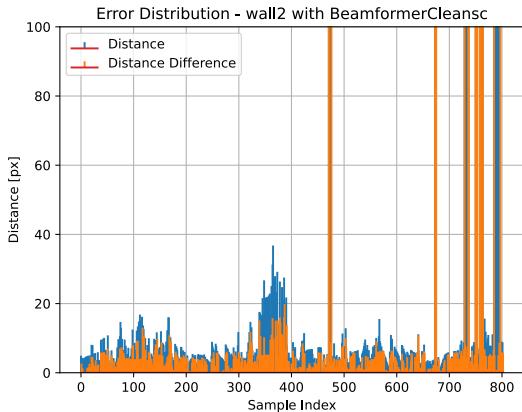
6.4.2 Analiza błędów



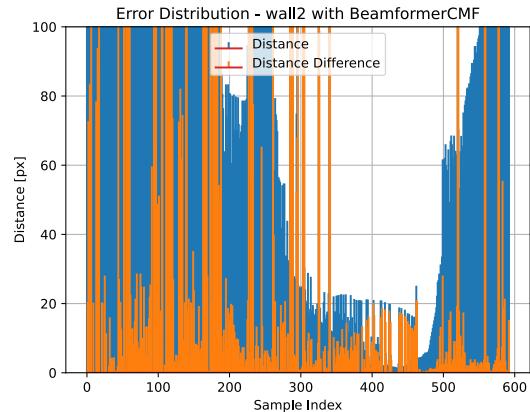
Rysunek 14: Rozkład błędu dla algorytmu DAS dla nagrania wall2



Rysunek 15: Rozkład błędu dla algorytmu Ortogonalnego dla nagrania wall2

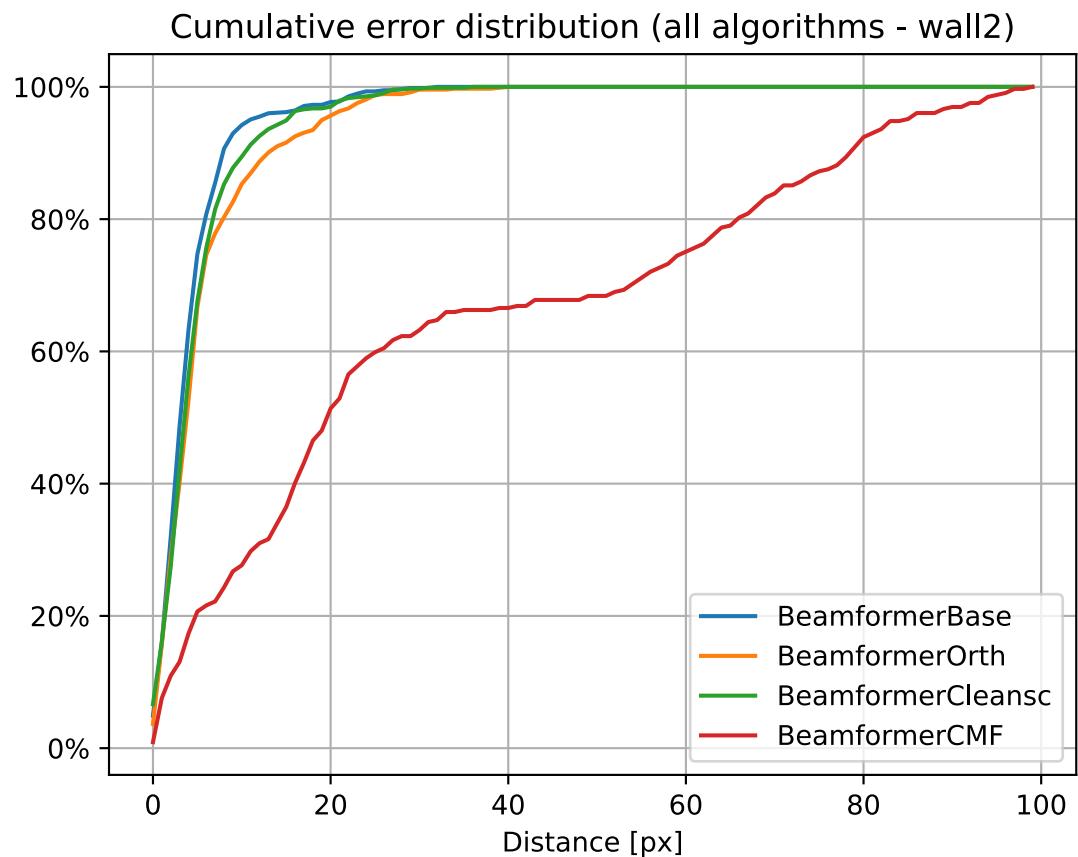


Rysunek 16: Rozkład błędu dla algorytmu Clean-SC dla nagrania wall2



Rysunek 17: Rozkład błędu dla algorytmu CMF dla nagrania wall2

6.4.3 Dystrybuanta



Rysunek 18: Dystrybuanta rozkładu błędu dla wszystkich algorytmów dla nagrania wall2

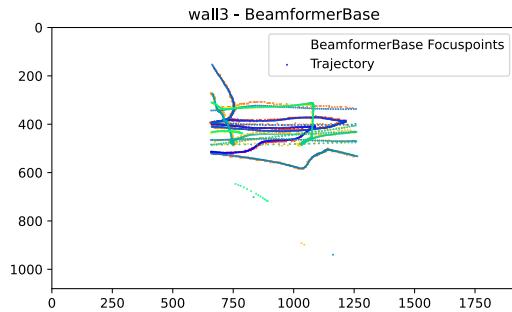
6.5 Nagranie wall3

Liczba ramek = 3410

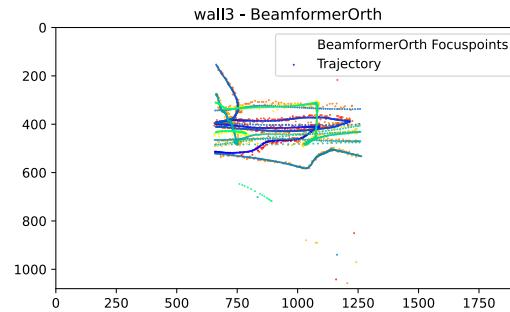
Algorytm	Całkowity czas dla ramek [s]	Średni czas dla ramki [s]	Maksymalny czas dla ramki [s]	Minimalny czas dla ramki [s]
DAS	79.6908	0.0233	0.1209	0.0180
Ortogonalny	87.0646	0.0255	0.1383	0.0208
Clean-SC	821.6074	0.2408	0.3684	0.1902
CMF	8700.8239	2.5508	3.7938	1.6426

Tabela 3: Podsumowanie czasu przetwarzania nagrania wall3

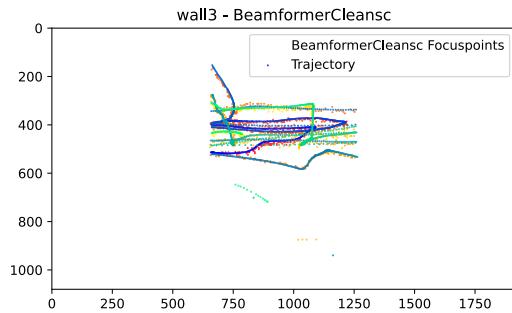
6.5.1 Trajektorie



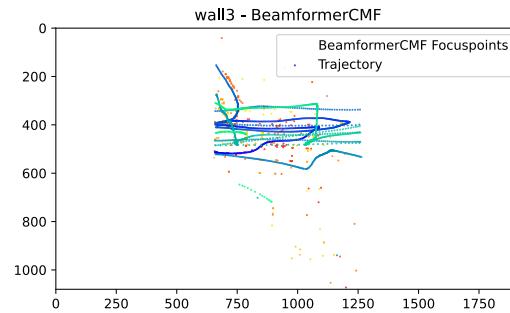
Rysunek 19: Trajektoria lotu drona dla algorytmu DAS dla nagrania wall3



Rysunek 20: Trajektoria lotu drona dla algorytmu Ortogonalnego dla nagrania wall3

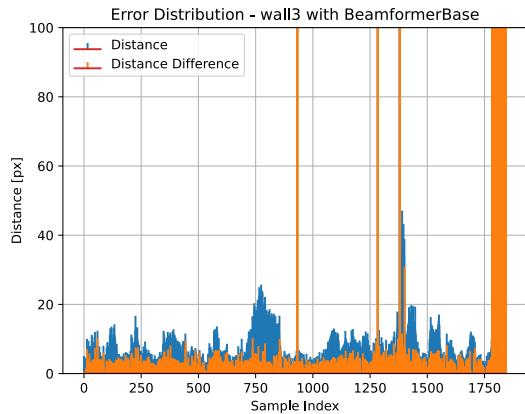


Rysunek 21: Trajektoria lotu drona dla algorytmu Clean-SC dla nagrania wall3

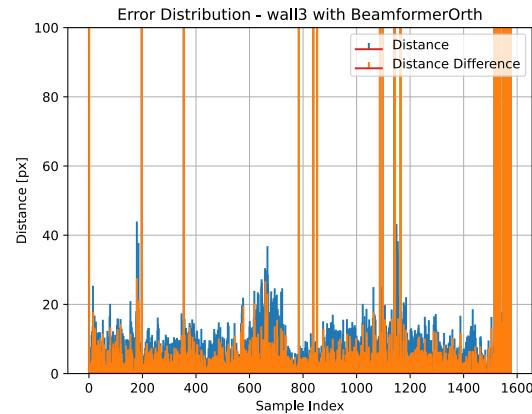


Rysunek 22: Trajektoria lotu drona dla algorytmu CMF dla nagrania wall3

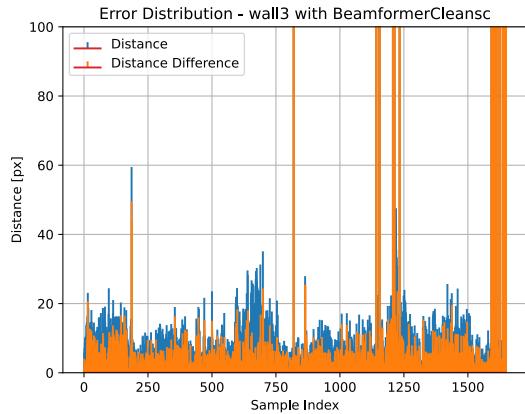
6.5.2 Analiza błędów



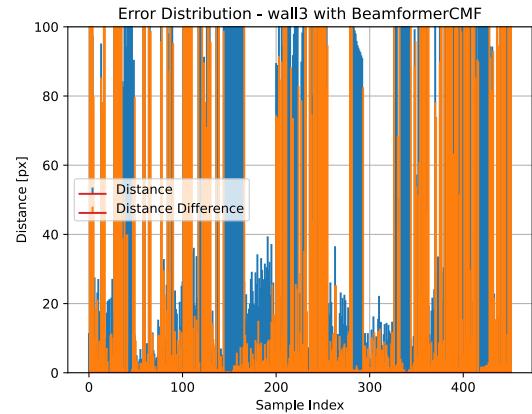
Rysunek 23: Rozkład błędu dla algorytmu DAS dla nagrania wall3



Rysunek 24: Rozkład błędu dla algorytmu Ortogonalnego dla nagrania wall3

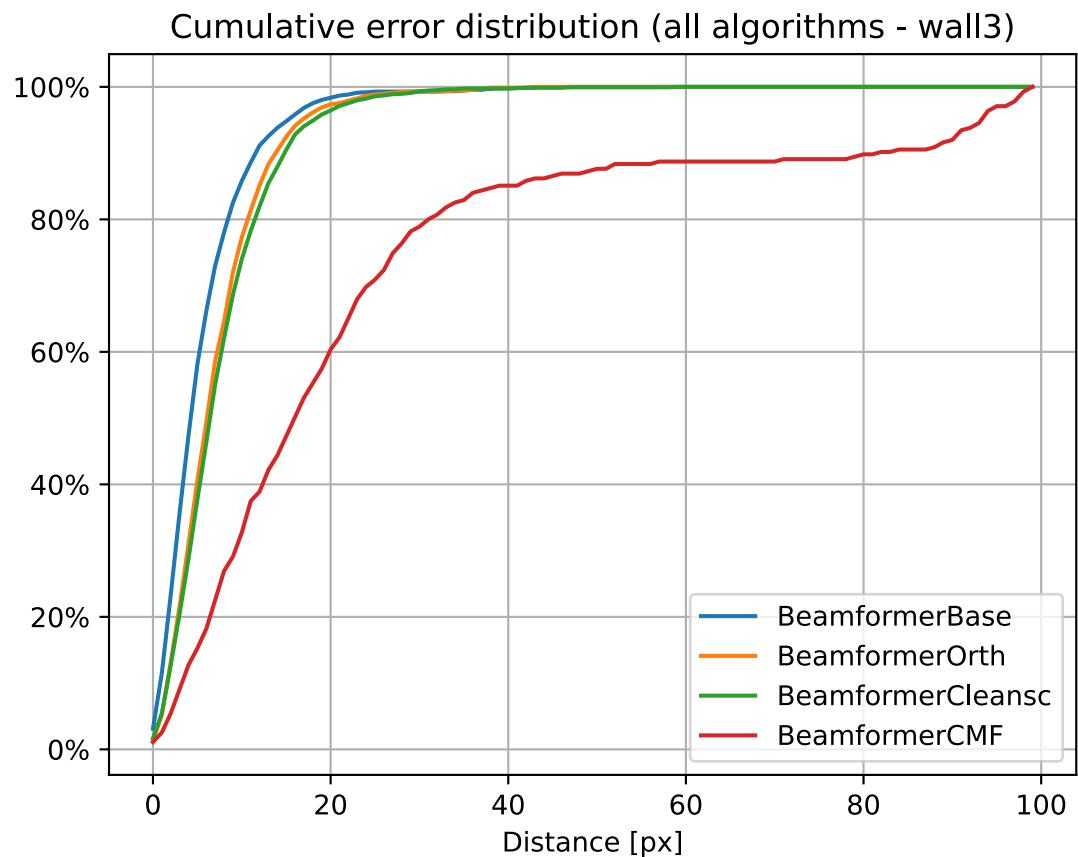


Rysunek 25: Rozkład błędu dla algorytmu Clean-SC dla nagrania wall3



Rysunek 26: Rozkład błędu dla algorytmu CMF dla nagrania wall3

6.5.3 Dystrybuanta



Rysunek 27: Dystrybuanta rozkładu błędu dla wszystkich algorytmów dla nagrania wall3

7 Podsumowanie

7.1 Wnioski

Na podstawie przeprowadzonych eksperymentów porównujących skuteczność i wydajność algorytmów lokalizacji drona (DAS, Ortogonalny, Clean-SC, CMF) przy użyciu biblioteki Acoular, można wyciągnąć następujące wnioski:

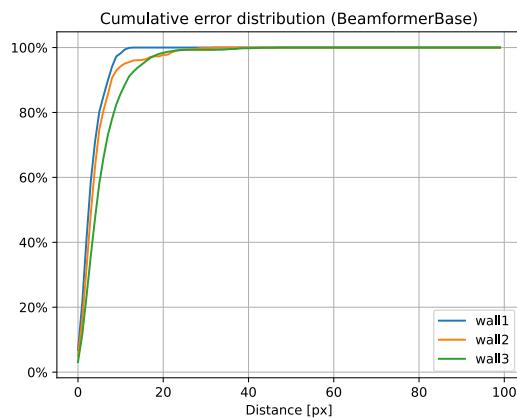
1. Wydajność obliczeniowa:

- Algorytm DAS wykazał się najkrótszym całkowitym czasem przetwarzania dla wszystkich testowanych nagrani, co czyni go najbardziej efektywnym pod względem czasu obliczeń.
- Algorytm Ortogonalny również osiągał konkurencyjne czasy, będąc minimalnie wolniejszym od DAS.
- Algorytm Clean-SC przetwarzał nagrania w dziesięciokrotnie dłuższy czasie.
- Algorytm CMF wykazał się najdłuższym czasem przetwarzania niemal stu-krotnie dłuższym od algorytmów DAS i Ortogonalnego.

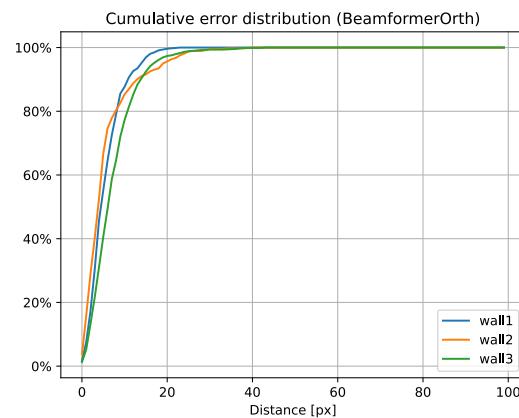
2. Skuteczność lokalizacji:

Skuteczność mierzona normą L_∞ (różnica między ręcznie wyznaczonymi a automatycznie określonymi punktami) wskazuje, że algorytmy różnią się w precyzji lokalizacji. Rozkłady błędów oraz dystrybuanty błędów sugerują, że algorytmy DAS i Ortogonalny oferują lepszą rozdrobnienie przestrzenną i mniejsze błędy w porównaniu do Clean-SC i CMF.

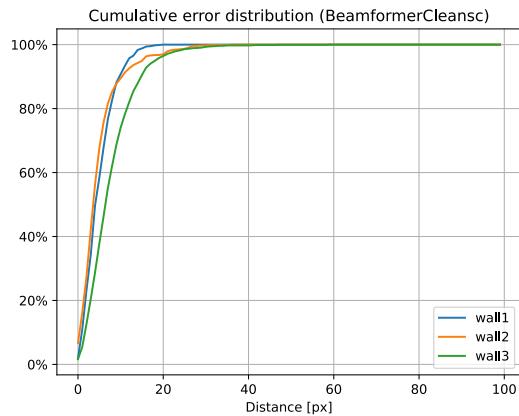
7.2 Dystrybuanty



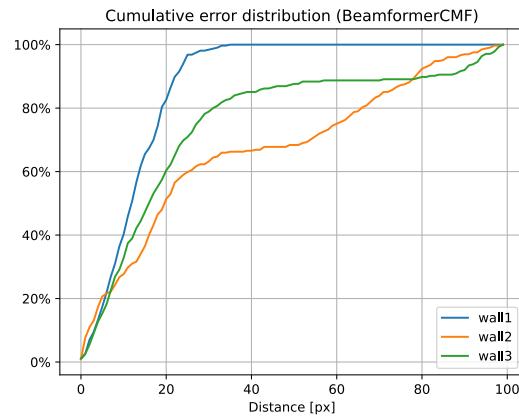
Rysunek 28: Dystrybuanty błędów dla algorytmu DAS dla wszystkich nagrani



Rysunek 29: Dystrybuanty błędów dla algorytmu Ortogonalnego dla wszystkich nagrani



Rysunek 30: Dystrybuanty błędów dla algorytmu Clean-SC dla wszystkich nagrań



Rysunek 31: Dystrybuanty błędów dla algorytmu CMF dla wszystkich nagrań

Bibliografia

- [1] Acouilar. *Strona internetowa biblioteki Acouilar*. Dostęp: 16 czerwca 2025. 2025. URL: <https://www.acouilar.org>.
- [2] Audacity. *Strona internetowa Audacity*. Dostęp: 16 czerwca 2025. 2025. URL: <https://www.audacityteam.org>.
- [3] Ffmpeg. *Strona projektu ffmpeg*. Dostęp: 16 czerwca 2025. 2025. URL: <https://ffmpeg.org>.
- [4] Intel. *Specyfikacja procesora i5-9300H*. Dostęp: 16 czerwca 2025. 2025. URL: <https://www.intel.com/content/www/us/en/products/sku/191075/intel-core-i59300h-processor-8m-cache-up-to-4-10-ghz/specifications.html>.
- [5] Matlab. *Strona internetowa Ground Truth Labeler*. Dostęp: 16 czerwca 2025. 2025. URL: <https://www.mathworks.com/help/driving/ref/groundtruthlabeler-app.html>.
- [6] Matplotlib. *Strona internetowa biblioteki Matplotlib*. Dostęp: 16 czerwca 2025. 2025. URL: <https://matplotlib.org>.
- [7] NumPy. *Strona internetowa biblioteki NumPy*. Dostęp: 16 czerwca 2025. 2025. URL: <https://numpy.org>.
- [8] SciPy. *Strona internetowa biblioteki SciPy*. Dostęp: 16 czerwca 2025. 2025. URL: <https://scipy.org>.