

Spis treści

[1. Wstęp 3](#_Toc89618554)

[2. Sterowanie robotem 4](#_Toc89618555)

[1. Gcode 4](#_Toc89618556)

[3. Kinematyka odwrotna robotów 6-cio osiowych 4](#_Toc89618557)

[4. Generowanie modelu obrabianego przedmiotu 4](#_Toc89618558)

[Rys.4.1 Graficzna wizualizacja chmury punktów 5](#_Toc89618559)

[Rys.4.2 Fragment tablicy wierzchołków 5](#_Toc89618560)

[Rys.4.3 Indywidualne konfiguracje punktów 6](#_Toc89618561)

[Rys.4.4 Kolejność przetważania danych przez algorytm Marching Cubes 6](#_Toc89618562)

[5. Podsumowanie wyników 7](#_Toc89618563)

[6. Bibliografia 7](#_Toc89618564)

# Wstęp

Obecnie obrabiarki CNC są w stanie sprostać większości powierzonych im zadań, ale ich największym ograniczeniem jest niewielkie pole robocze. Istnieją frezarki o dużym polu roboczym, ale są one bardzo drogie. Istnieje tańsze rozwiązanie na skrawanie dużych obiektów, które można zrealizować montując wrzeciono na robocie przemysłowym. Mają one duży zasięg ramienia, który przekłada się na duże pole robocze. Dodatkowym atutem jest szósta oś, zapewniająca dodatkowy stopień swobody. Umożliwia to dotarcie narzędziem do miejsc do których dotarcie byłoby niemożliwe przy konwencjonalnych obrabiarkach CNC. Zastosowanie obrotnika lub prowadnic liniowych, jeszcze bardziej zwiększa elastyczność systemu. Roboty przemysłowe są jednak rzadziej stosowane w obróbce skrawaniem niż konwencjonalne centra obróbcze. Składa się na to kilka czynników, m.in. niewystarczająca sztywność konstrukcji spowodowana długim ramieniem robota, która skutkuje błędami pozycjonowania. Podczas skrawania dokładność pozycji narzędzia wynosi 0.5 - 2mm oraz jest zależna od aktualnej odległości narzędzia od podstawy robota. Pomimo wad roboty wykorzystywane są do obróbki łatwiej skrawalnych materiałów, gdy uzyskiwana dokładność mieści się w zadanej tolerancji wymiarów.

Problematyczne okazuje się również zaprogramowanie robota. Programowanie online jest praktycznie nie wykonalne w kontekście obróbki skrawaniem. Ręczne programowanie offline również nie jest najlepszym sposobem - duży przedmiot będzie wymagał długiego programu, który zajmie dużo czasu do napisania oraz prawdopodobnie będzie zawierał błędy. Najbardziej pożądaną formą zapisu procesu jest G-code, który jest stosowany w programowaniu obrabiarek sterowanych numerycznie. Aby nie pisać programu ręcznie można go wygenerować za pomocą oprogramowania CAM.

Uruchomienie programu na rzeczywistym robocie bez wcześniejszego sprawdzenia jest ryzykowne i może prowadzić do kosztownych uszkodzeń. W takiej sytuacji pomocne są symulatory pozwalające na odtworzenie programu bez obaw o robota. Podczas symulacji wirtualny robot zachowuje się jak fizyczny, możliwe jest wykrycie np. kolizji ramienia robota z blokiem materiału. Po zakończeniu symulacji otrzymujemy również podgląd finalnego produktu.

Celem pracy było napisanie oprogramowania zdolnego do zasymulowania obróbki skrawaniem używając robota przemysłowego. Symulacja odbywa się dla robota KUKA KR 60 L30 HR. Oprogramowanie umożliwi sterowanie ręczne robotem za pomocą TCP i zmianą kąta przegubów, oraz wykonanie programu g-code. Aby możliwe było umieszczenie narzędzia w zadanym punkcie w przestrzeni, program oblicza kinematykę odwrotną. Aby zwizualizować efekty pracy robota, w czasie rzeczywistym generowany jest model 3D obrabianego przedmiotu.

# Sterowanie robotem

Rzeczywistym robotem można sterować na kilka sposobów. Wykorzystując panel sterowania można na żywo przemieszczać TCP w globalnym układzie współrzędnych robota lub lokalnym TCP. Sterować można również bezpośrednio przegubami robota. Funkcje te zostały przeniesione do symulatora, pozwalając na ręczne sterowanie wirtualnym robotem.

## Przemieszczanie TCP

TCP można przemieszczać w dwóch trybach. Pierwszy z nich to układ współrzędnych robota,

## Obrót przegubów robota

Sterowanie osiami polega na nadaniu każdej z nich określonego kąta do jakiego ma dążyć. Wirtualny robot bez problemu może osiągnąć każdy zadany kąt na każdej z osi. Rzeczywisty robot posiada ograniczenia kątów które każda z osi może przyjąć. Jest to dobre miejsce, aby upewnić się że ograniczenia fizycznego robota zostaną odwzorowane. Zadana wartość kąta jest korygowana. Odpowiedzialna jest za to funkcja przedstawiona na rysx.



* 1. Funkcja ustalająca kąt na osi robota

Na początku należy sprawdzić czy przekazany kąt jest liczbą. Będzie to przydatne później podczas obliczeń kinematyki odwrotnej. Niektóre osie mogą obracać się w 360° dlatego wprowadzono możliwość wyłączenia korekcji ustawiając oraz . Następnie funkcja nadaje już poprawny kąt dla osi robota głównego oraz jego ducha.

Aby uzyskać ruch ramienia kąt jest interpolowany. Nie może to być jednak zwykła interpolacja liniowa ponieważ prowadziłoby to do nie optymalnej drogi. Na przykład ruch z do wymagałby 300 sekund przy prędkości obrotowej dla zwykłej interpolacji liniowej kąta. Lepszym rozwiązaniem byłby ruch w przeciwną stronę. Przy tej samej prędkości zająłby tylko 30 sekund. W silniku Unity istnieje funkcja realizująca interpolację liniową dla kątów, która eliminuje przytoczony problem. Na rysXXX przedstawiono funkcję która jest wykonywana w każdą klatkę działania programu Zmienia ona kąt „Z” który jest równoznaczny z kątem . Jednak nie można zmienić pojedynczego składnika, konieczne jest stworzenie nowego wektora 3D przechowywującego informację o rotacji w postaci kątów Eulera. Jest on następnie konwertowany na kwaternion. Tej 4-ro wymiarowej struktury silnik używa do manipulacji obrotem obiektów.



* 1. Funkcja interpolująca kąt przegubu robota

## Gcode

G-code to język zapisu poleceń dla obrabiarek sterowanych numerycznie. Zapisana jest w nim dokładna ścieżka którą musi pokonać narzędzie, aby poprawnie wykonać przedmiot. Funkcji g-code jest bardzo dużo, do symulatora zostały zaimplementowanie tylko podstawowe:

* G0 – ruch szybki,
* G1 – ruch liniowy interpolowany,
* G2 – ruch kołowy zgodnie z ruchem wskazówek zegara,
* G3 – ruch kołowy przeciwnie do ruchu wskazówek zegara.

Oraz parametry oraz osie:

* X, Y, Z – położenie punktu,
* A, B, C – orientacja punktu,
* F – prędkość liniowa,
* S – prędkość obrotowa wrzeciona.

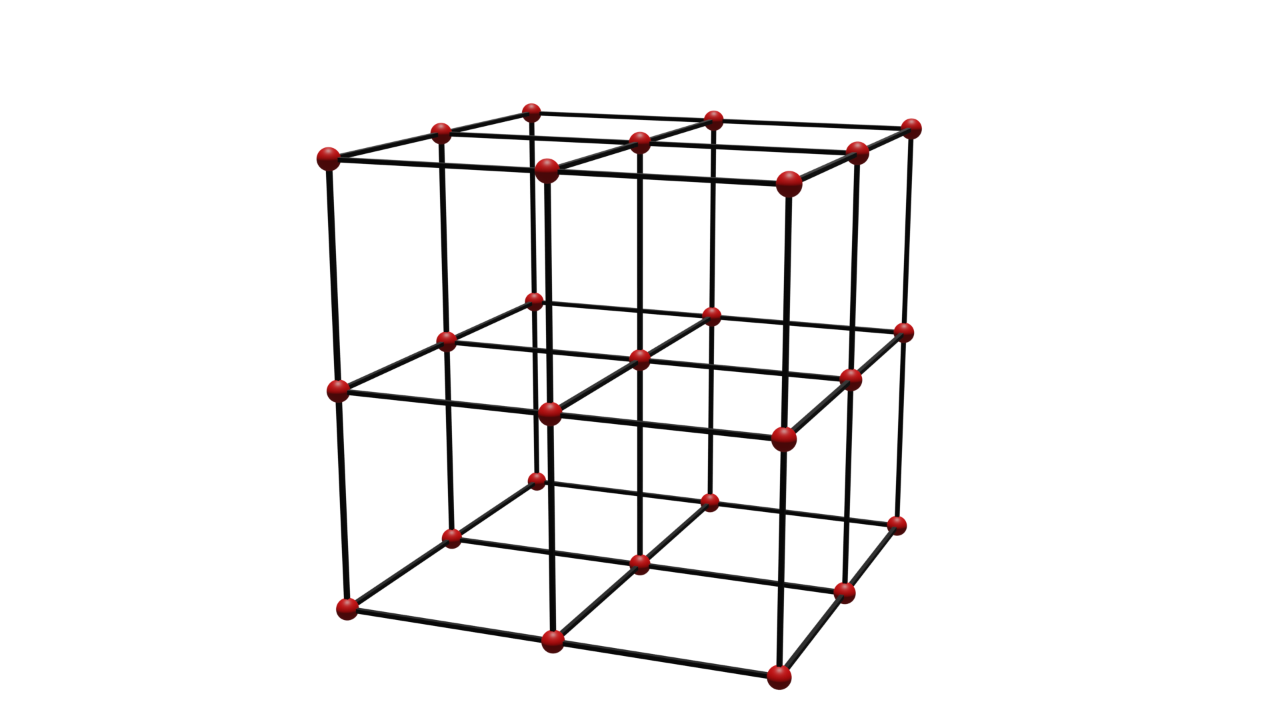
W przeciwieństwie do pozostałych parametrów, podczas symulacji prędkość wrzeciona nie ma większego znaczenia. Istotna jest informacja czy wrzeciono jest uruchomione – .

# Kinematyka odwrotna robotów 6-cio osiowych

# Generowanie modelu obrabianego przedmiotu

Sam ruch robota jest niewystarczający do weryfikacji napisanego programu. Wygenerowanie modelu 3D obrabianego przedmiotu pozwoli na wizualną ocenę wykonanej pracy narzędzia.

Kształcie bryły jest definiowany za pomocą trójwymiarowej chmury punktów (Rys.4.1). Są one rozłożone w równej odległości od siebie, tworząc sześcian. Ich ilość jest ustalona przy uruchomieniu programu. Informacja przechowywana jest w postaci „0” lub ”1”.



* 1. Graficzna wizualizacja chmury punktów

Proces skrawania jest symulowany poprzez sprawdzenie, które punkty znajdują się wewnątrz narzędzia. Jest to realizowane za pomocą serii obliczeń oraz instrukcji warunkowych.

WYJAŚNIENIE SPRAWDZANIA PUNKTU

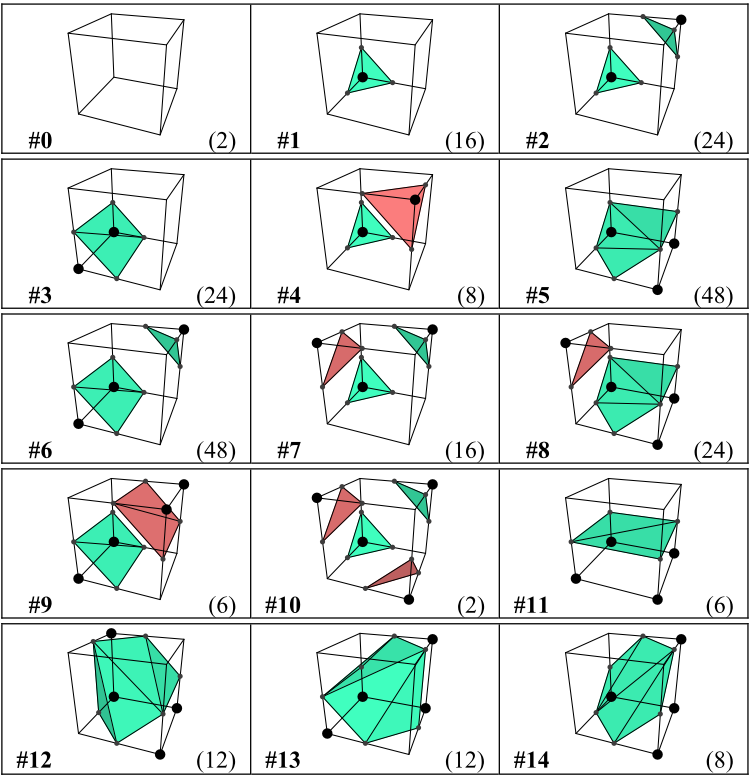
Aby poprawnie wyświetlić obrabiany przedmiot konieczne jest dodanie algorytmu generującego siatkę trójkątów na podstawie chmury punktów. Do tego celu został użyty algorytm „Marching Cubes”.

Algorytm generuje siatkę trójkątów zależną od 8 punktów znajdujących się na wierzchołkach sześcianu. Zatem na jeden sześcian przypada możliwych kombinacji. Informacja jak powinien wyglądać dana konfiguracja zakodowana jest w tablicy, której fragment przedstawiony jest na Rys.4.2.



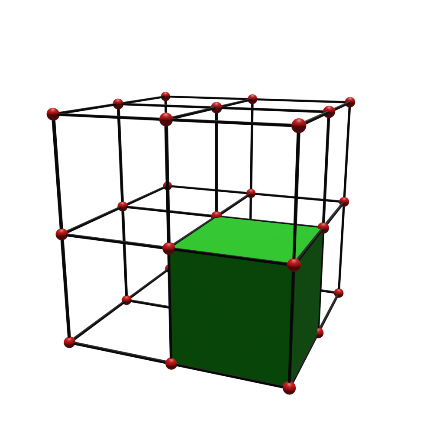
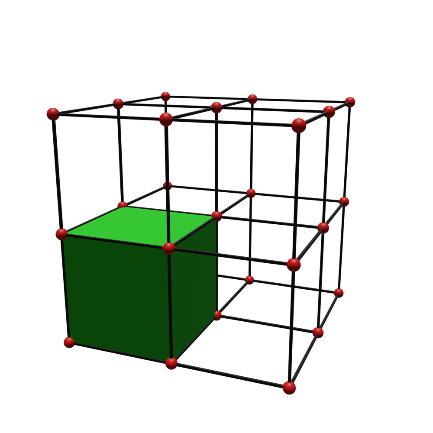
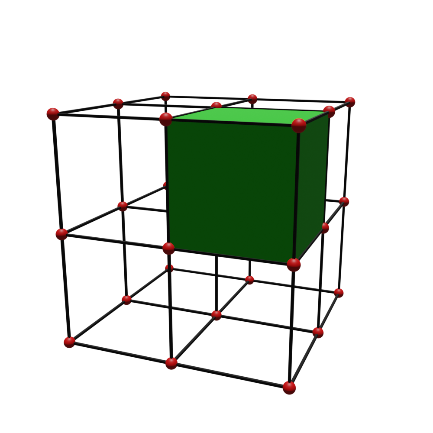
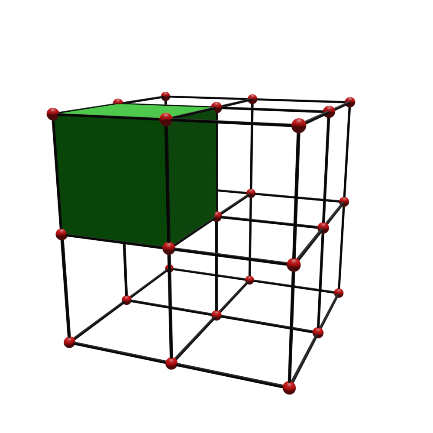
* 1. Fragment tablicy wierzchołków

Większość z nich jest rotacją lub negacją innych. Indywidualnych kombinacji jest 15 i zostały przedstawione zostały na Rys.4.3. Czarną kropką oznaczono wierzchołki które są w materiale, wierzchołki bez kropki znajdują się poza. Zielona ścianka widziana jest od przodu, czerwona od tyłu. W nawiasie podano liczbę podobnych kombinacji – obrócone lub/i zanegowane.



* 1. Indywidualne konfiguracje punktów

Algorytm jest powtażany dla każdej grupy punktów, jak przedstawiono na Rys.4.4. Generowana siatka trójkątów dla każdej grupy jest ułożona w taki sposób aby pasowała do sąsiadujących siatek, tworząc tym samym pełny model 3D obiektu.



* 1. Kolejność przetważania danych przez algorytm Marching Cubes

Dokładność modelu zależna jest od ilości punktów. Zwiększenie ilości punktów zwiększy również ilość obliczeń do wykonania, które muszą być wykonywane w każdej klatce działania programu. Wiec aby przyśpieszyć program przetwarzanie punktów oraz generowanie modelu odbywa się za pośrednictwem karty graficznej, która została stworzona do przeprowadzania wielu równoległych, ale prostych obliczeń.

PORÓWNANIE BLOKU O MAŁEJ ROZDZIELCZOŚCI ORAZ TEGO O DUŻEJ – FAJNIE BY BYŁO WYKONAĆ TEN SAM G-KOD

Ustalono wymiary chmury punktów na 100x100x100 punktów. W sumie jest to punktów, które trzeba sprawdzić oraz wygenerować na ich podstawie siatki trójkątów. Daje to akceptowalną rozdzielczość przy stabilnych 10 klatkach na sekundę.

Chmura punktów przyjmuje formę sześcianu, więc odległość między punktami zależna jest od najdłuższego boku półfabrykatu.

# Podsumowanie wyników

RZECZY KTÓRE MOŻNABY ZROBIĆ LEPIEJ:

- OPTYMALIZACJA GENEROWANIA MODELU 3D

# Bibliografia

x

|  |  |
| --- | --- |
| 1. | Lague, S.: Marching-Cubes.GitHub. (2019) <https://github.com/SebLague/Marching-Cubes> |
| 2. | Bourke, P.: Polygonising a scalar field. (1994) <http://paulbourke.net/geometry/polygonise/> |
| 3. | Automatic Addison: The Ultimate Guide to Inverse Kinematics for 6DOF Robot Arms. (2020) |
| 4. | KUKA Roboter GmbH: KRL Reference Guide. |
| 5. | Lague, S.: Terraforming.GitHub. (2021) <https://github.com/SebLague/Terraforming> |
| 6. | Lengyel, E.: Voxel-Based Terrain for Real-Time Virtual Simulations. (2010) |
| 7. | Łygas, K., Danilczuk, W.: Translator G-codu na język robota Kawasaki. |
| 8. | Khatamian, A.: Solving Kinematics Problems of a 6-DOF Robot Manipulator. |
| 9. | Pandremenos, J., Doukas, C., Stavropoulos, P., Chryssolouris, G.: Machining with robots: a critical review. (2011) |

x