Terbit *online* pada laman web jurnal : http://metal.ft.unand.ac.id

METAL: Jurnal Sistem Mekanik dan Termal

| ISSN (Print) 2598-1137 | ISSN (Online) 2597-4483 |

Artikel Penelitian

pyScrew4Mobility: Modul Python untuk Penentuan Mobilitas Manipulator Paralel Berbasis Teori *Screw*

Adriyan[a]

[a] *Program Studi Teknik Mesin, Sekolah Tinggi Teknologi Nasional, Jambi, 36124, Indonesia*

INFORMASI ARTIKEL

Sejarah Artikel:
Diterima Redaksi: 03 Februari 2020
Revisi Akhir: 23 Maret 2020
Diterbitkan *Online*: 05 April 2020

KATA KUNCI

Teori *screw*
Manipulator paralel
Mobilitas
Sympy
pyScrew4Mobility

KORESPONDENSI

E-mail: adriyan0686@gmail.com

A B S T R A C T

This paper addresses the mobility determination of parallel manipulators (PMs) using screw theory with algebra methods on an open-source computer algebra package: Sympy (symbolic python). The screw theory can specify the number and the type of motions owned by PMs with over-constrained or non-over constrained kinematic structures. The algebra methods are applied to obtain reciprocal of a screw or a screw system and basis of a screw system using null-spaces and the row/column-spaces technique, respectively. Hence, an object-oriented python module, called a `pyScrew4Mobility` module, is designed to realize such implementation. The module consists of three classes namely a `Screw`, a `ScrewSystem` and a `ManipulatorMobility`. A screw is constructed by the class of `Screw`, including its algebraic calculation such as negation, addition, multiplication, and product of two screws. Then, the `ScrewSystem` is used to construct a system of screws. It can be used to find reciprocal screws, unique screws within the system of screws, and calculate the products of two systems of screws. The last class or `ManipulatorMobility` has a direct implementation to determine the mobility of PMs. It uses information from the list of unit screw direction, position, and pitch. Finally, the designed screw module is tested to demonstrate its capability to determine the mobility of four well-known PMs, i.e. 3-PRRR; 3-PR(Pa)R; 4-PRRU; and 6-UPS, including the respective time spent for calculation.

1. PENDAHULUAN

Manipulator paralel (*parallel manipulators*) merupakan jenis manipulator robotik yang dikonstruksi oleh satu batang tetap sebagai *base* dan satu batang apung yang disebut *platform* yang keduanya dihubungkan dengan minimal dua buah kaki yang disebut juga dengan *limb* atau *legs* [1]. Setiap kaki terdiri setidaknya atas dua buah batang kaku dan 3 buah sambungan (*joints*). Dengan mengacu pada konstruksi rantai kinematiknya dapat ditentukan kemampuan manipulator paralel dalam menghasilkan gerakan pada *platform*-nya. Kemampuan ini disebut dengan mobilitas (*mobility*) atau derajat kebebasan atau *degree of freedom* (disingkat dengan DOF) manipulator paralel.

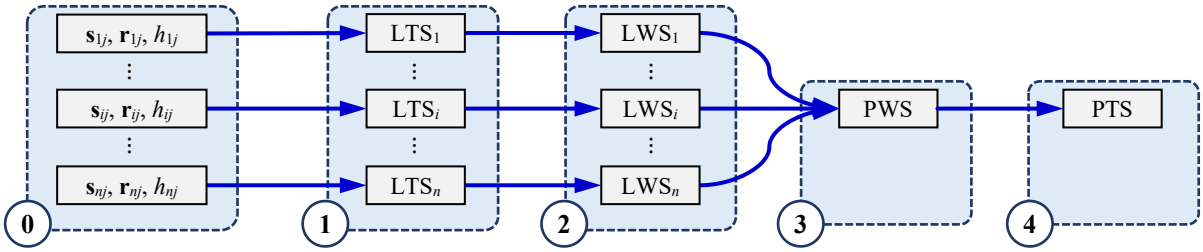
Cara umum yang digunakan dalam menentukan mobilitas manipulator paralel adalah dengan menerapkan formulasi Grübler-Kutzbach (disingkat GK). Dalam hal ini, formulasi GK membutuhkan jumlah batang, jumlah sambungan, dan jumlah gerakan yang diizinkan oleh setiap sambungan untuk menentukan mobilitas suatu manipulator paralel. Namun, mobilitas suatu manipulator paralel yang diperoleh dengan

menerapkan formulasi GK hanya menampilkan jumlah DOF manipulator paralel saja. Sementara itu, ketika konstrain redundan dimiliki oleh suatu manipulator paralel maka formulasi GK tidak dapat menentukan mobilitas yang seharusnya dimiliki oleh manipulator paralel, seperti yang dinyatakan oleh Dai, dkk [2]. Di samping itu, formulasi GK juga tidak dapat menghasilkan DOF/mobilitas pada manipulator paralel yang memiliki mobilitas lokal atau DOF pasif, seperti pada manipulator paralel 6-SPS. Meskipun demikian, modifikasi formulasi GK telah banyak dilakukan untuk mengatasi kekurangan ini, seperti yang dideskripsikan oleh [3]. Akan tetapi, jenis DOF pada manipulator paralel tidak dapat dihasilkan melalui modifikasi formulasi ini.

Di samping formulasi GK, teori *screw* juga dapat diterapkan untuk menentukan informasi mobilitas manipulator paralel seperti jumlah DOF, jenis DOF, dan arah gerakan dari DOF tersebut [2], [4]–[6], juga kondisi terkopel atau tidaknya DOF manipulator paralel itu [6]. Teori *screw* diperkenalkan pertama kalinya oleh Sir Robert Stawell Ball lebih dari satu abad yang lalu [7]. Dewasa ini, teori *screw* merupakan perangkat untuk bidang manipulator robotik atau mekanisme paralel. Ini dibuktikan dengan telah diterapkannya teori *screw* pada berbagai sub bidang pada kajian manipulator/mekanisme paralel, seperti penentuan mobilitas atau DOF, sintesis struktur [8], analisis kinematika orde tinggi (misal *jerk* dan *jounce*) [9], memperoleh Jacobian [1], dan menentukan transmisibilitas gerakan dan gaya [10].

Kong dan Gosselin telah mengajukan suatu proses penentuan mobilitas manipulator paralel dalam dua langkah, yaitu analisis mobilitas sesaat (*instantaneous mobility analysis*) dan inspeksi mobilitas penuh (*full-cycle mobility inspection*) [4]. Zhao, dkk telah memperlihatkan bahwa penggunaan metode geometri berbasis sistem *screw* (*screw systems*) manipulator paralel untuk memperoleh *screw* resiprokalnya (*reciprocal screws*) cukup efektif [5]. Dalam makalah mereka juga telah dinyatakan suatu prosedur dalam menentukan mobilitas paralel manipulator berbasis sistem *screw* dan resiprokal *screw*-nya, mulai dari setiap kaki (*limb*) hingga ke *platform* seperti ditunjukkan pada Gambar 1. Pada Gambar 1 ini , $i = 1 - n$; dan $j = 1 - m$, dengan n dan m menyatakan jumlah kaki (*limb*) dan sambungann (*joint*).

Penggabungan teori *screw* dan topologi suatu mekanisme telah digunakan oleh Wang, dkk dalam menentukan mobilitas manipulator paralel [6]. Melalui penelitian mereka ini telah diberikan tiga langkah utama dalam menentukan mobilitas sebuah manipulator paralel. Langkah pertama adalah identifikasi topologi dari manipulator paralel yang akan dikaji. Selanjutnya, penerapan teori *screw* dan resiprokal *screw* untuk memperoleh *motion screw* pada setiap kaki merupakan langkah kedua. Pada tahap ini, *motion screw* untuk setiap kaki harus didefinisikan pada kerangka acuan tetap (*fixed reference frame*). Akhirnya, langkah ketiga digunakan untuk memperoleh mobilitas manipulator paralel dari *motion screw* yang dimiliki oleh *platform* berupa nominal, jenis, arah gerakan, dan hubungan lengkap terkait ada atau tidaknya DOF terkopel [6].



Gambar 1. Proses untuk penentuan mobilitas manipulator paralel menggunakan teori *screw*

Artikel ini menyajikan suatu implementasi sederhana teori *screw* untuk menentukan mobilitas manipulator paralel dengan menggunakan perangkat lunak *open-source computer algebra system* yaitu Sympy [11], [12]. Implementasi ini dilakukan dengan mengembangkan sebuah modul dalam bahasa pemrograman python menggunakan pustaka Sympy yang mengacu pada [2], [5], [6], [13]. Realisasi penentuan mobilitas manipulator paralel ini dinyatakan ke dalam sebuah modul dalam bahasa python bernama `pyScrew4Mobility` berikut dengan *class* dan *methods*-nya. Bagian hasil dan pembahasan discussion akan menyajikan unjuk kerja modul `pyScrew4Mobility` untuk menentukan mobilitas empat buah manipulator paralel, yaitu 3-PRRR atau tripteron, 3-PR(Pa)R, 4-PRRU, dan 6-UPS. Sedangkan, bagian akhir akan memaparkan kesimpulan dari penelitian.

2. METODOLOGI

2.1. Proses Penentuan Mobilitas Manipulator Paralel

Langkah-langkah yang digunakan untuk menentukan mobilitas suatu manipulator paralel dilustrasikan di dalam Gambar 1. Langkah pertama hingga langkah keempat mengadaptasi hasil penelitian Zhao, dkk [5]. Langkah pertama dilakukan dengan mengkonstruksi seluruh sistem *motion screw* atau sistem *twist* dari setiap kaki (*limb*) manipulator yang kemudian disebut dengan *limb twist system* (disingkat LTS). Untuk mengkonstruksi LTS pada setiap kaki (*limb*) diharuskan untuk menghitung keseluruhan twist satuan (*unit twist* atau *unit motion screw*) dari setiap sambungan (*joint*) pada masing-masing kaki manipulator.

Berdasarkan teori *screw* bahwa suatu *screw* didefinisikan secara matematis oleh

$$\$ = \rho\hat{\$} = \rho\begin{pmatrix}\mathbf{s} & \mathbf{s}_0\end{pmatrix} = \rho\begin{pmatrix}\mathbf{s} & \mathbf{r}\times\mathbf{s} + h\mathbf{s}\end{pmatrix}, \qquad (1)$$

dengan ρ merupakan magnitudo *screw* yang bernilai 1 untuk *screw* satuan (*unit screw*), kecepatan sudut untuk *twists*, atau gaya untuk *wrenches*. $\hat{\$}$ merupakan *screw* satuan, **s** adalah vektor satuan yang menyatakan arah dari sumbu *screw*, dan **r** merupakan vektor posisi yang mendefinisikan posisi **s** di dalam suatu kerangka acuan tetap. Terakhir, *h* menyatakan *pitch* dari *screw*.

Pitch suatu *screw*, *h*, dapat memiliki nilai dalam rentang 0 hingga ∞. *Screw* dengan *pitch* bernilai 0 (0-*pitch screw*) merupakan representasi dari gerak rotasi atau momen gaya/kopel gaya. Sementara itu, *screw* dengan *pitch* bernilai ∞ (∞-*pitch screw*) merepresentasikan gerak translasi atau gaya. Dengan demikian, persamaan (1) dapat direpresentasikan menjadi,

$$\$ = \rho\hat{\$} = \begin{cases}\rho\begin{pmatrix}\mathbf{s} & \mathbf{r}\times\mathbf{s}\end{pmatrix}; & \text{if } h = 0,\\ \rho\begin{pmatrix}\mathbf{0} & \mathbf{s}\end{pmatrix}; & \text{if } h = \infty.\end{cases} \qquad (2)$$

Sambungan revolut (R) dan sambungan prismatik (P) masing-masingnya merupakan *twist* dengan pitch bernilai 0 dan ∞. Sambungan mekanis lainnya seperti sambungan universal (U), silindris (C), dan sferis (S) dapat dinyatakan sebagai kombinasi dari beberapa sambungan R dan/atau P. Misalnya, sambungan U, C dan S dapat direpresentasikan masing-masingnya sebagai dua buah sambungan R yang saling bersilangan tegak lurus, satu sambungan R dan satu sambungan P di satu sumbu, dan tiga buah sambungan R yang saling bersilangan tegak lurus.

Untuk mengkonstruksi LTS pada setiap kaki (*limb*) dibutuhkan tiga parameter utama, yaitu **s**, **r**, dan *h*. Kebutuhan tiga parameter utama ini di dalam penelitian ini disebut dengan langkah ke nol (*the zeroth step*) yang tentunya bergantung pada konstruksi geometris setiap manipulator paralel yang menjadi kajian. Baik **s** dan **r** harus didefinisikan pada sistem kerangka acuan tetap yang biasanya ditempatkan pada *base* manipulator paralel, sebagaimana yang dinyatakan oleh [6] sebagai langkah kedua. Dalam penelitian ini, **s** dan **r** pada setiap kaki (*limb*) ditransformasikan secara langsung ke kerangka acuan tetap sebelum

mengkonstruksi setiap *twist* satuan pada setiap sambungan di masing-masing kaki (*limb*) manipulator paralel. Dalam hal implementasi langkah ini akan dideskripsikan pada sub bagian 2.2. Dengan demikian, LTS pada setiap kaki (*limb*) akan membentuk *twist m-system*, dengan *m* merupakan jumlah *screw* yang dimiliki oleh kaki (*limb*) tertentu. Meskipun *m* ini dapat bernilai lebih dari 6, tetapi jumlah gerakan yang unik pada setiap kaki (*limb*) tidak akan pernah lebih dari 6. Akhirnya, *m* dapat dinyatakan sebagai jumlah gerakan unik yang dimiliki oleh kaki (*limb*) manipulator paralel.

Langkah kedua dari prosedur pada Gambar 1 adalah penentuan sistem *constraint screw* atau sistem *wrench* yang mengkonstrain gerakan pada setiap kaki (*limb*). Sistem *wrench* pada setiap kaki dinamakan dengan *limb wrench system* atau disingkat dengan LWS. LWS pada setiap kaki manipulator (*limb*) dapat ditentukan dengan menemukan sistem *screw* yang resiprokal terhadap LTS-nya. Misal, LWS pada *limb* pertama ditentukan dari sistem *screw* yang resiprokal dari LTS pada *limb* pertama. Hal ini akan mengakibatkan bahwa sistem *screw* yang resiprokal ini akan membentuk sistem *wrench* sejumlah 6 – *m* sebagai LWS pada suatu kaki manipulator. Penentuan produk resiprokal dari dua *screw* tunggal secara matematis dapat dihitung dengan

$$\$ \circ \$^r = \rho\rho^r \left(\mathbf{s} \cdot \mathbf{s}_0^r + \mathbf{s}^r \cdot \mathbf{s}_0 \right), \qquad (3)$$

dengan *superscript r* menyatakan *screw* resiprokal.

Setelah keseluruhan sistem *wrench* pada setiap kaki manipulator berhasil ditemukan, selanjutnya sistem tersebut dapat digunakan untuk menentukan sistem *wrench* pada *platform* manipulator yang merupakan langkah ketiga. Sistem *wrench* pada *platform* manipulator yang kemudian disebut dengan *platform wrench system* (PWS) merupakan sistem *wrench* yang unik dari setiap sistem *wrench* pada kaki manipulator (LWS).

Akhirnya, langkah keempat dapat dilakukan dengan menentukan sistem *screw* resiprokal dari PWS. Sistem *screw* resiprokal dari PWS adalah berupa sistem *twist* yang bekerja pada *platform* manipulator dan kemudian disebut dengan *platform twist system* (PTS). Jika *k* merupakan jumlah *wrenches* atau *constraint screw* pada PWS, maka jumlah *twists* atau *motion screw* pada PTS adalah sebesar 6 – *k*. Melalui PTS yang telah diperoleh ini dapat ditarik informasi terkait jumlah (nominal) DOF, jenis DOF dan arah gerakan dari DOF yang dimiliki oleh suatu manipulator paralel.

2.2. Realisasi Proses Penentuan Mobilitas Manipulator Paralel Menggunakan Sympy

Untuk merealisasikan kelima langkah yang dinyatakan oleh Gambar 1 dilakukan dengan menggunakan bahasa pemrograman python versi 3 dan pustaka sympy sebagai *open-source computer algebra system*, [11], [12]. Hanya langkah pertama hingga keempat yang dipersiapkan menjadi sebuah modul python yang kemudian dinamakan dengan modul `pyScrew4Mobility`. Sementara itu, langkah kenol merupakan tahapan yang bergantung pada konfigurasi geometri manipulator itu sendiri, sehingga bersifat didefinsikan sendiri oleh pengguna (*user defined*). Modul `pyScrew4Mobility` dikembangkan dengan menggunakan python versi 3.7 and SymPy versi 1.14 pada lingkungan PyCharm Professional IDE 2019.3 (lisensi academis penulis). Modul ini dikembangkan untuk untuk tiga *class* yaitu *class* `Screw`, *class* `ScrewSystem`, dan *class* `ManipulatorMobility`. Untuk itu, diagram *unified modeling language* (UML) ketiga class ini disediakan dalam artikel ini.

Setelah modul `pyScrew4Mobility` dikembangkan, selanjutnya perlu dicek performansinya dalam menentukan mobilitas manipulator paralel. Untuk itu, dalam penelitian ini digunakan empat manipulator paralel untuk maksud tersebut, yaitu 3-PRRR [14], [15]; 3-PR(Pa)R [16]; 4-PRRU [17]; dan 6-UPS [18], [19]. Mobilitas keempat manipulator paralel ini dihitung dengan membuatkan beberapa baris kode program python di dalam lingkungan kerja Jupyter Lab atau Jupyter Notebook [20]. Jupyter Notebook atau

Jupyter Lab merupakan sebuah lingkungan kerja dokumen saintifik interaktif (dikenal dengan *notebook*), berupa sebuah aplikasi berbasis peramban (*browser-based*) yang dapat menempatkan visualisasi statik atau dinamik, tabel, kode html, markdown, latex, javascript, video, dan gambar. Saat ini, Jupyter Notebook atau Jupyter Lab tidak hanya mendukung kernel untuk python tetapi untuk bahasa pemrograman lainnya seperti R, Julia, dll.

Langkah ke nol bergantung pada konfigurasi geometri manipulator. Pengguna harus mendefinisikan sendiri sebuah fungsi yang mendeskripsikan geometri manipulator paralel yang menjadi kajian seperti yang ditunjukkan oleh *cell* **`In[q]`** dalam Gambar 2. Langkah ini memerlukan modul `sympy.physics.mechanics` yang merupakan modul bawaan (*built-in*) pustaka sympy untuk mekanika benda kaku seperti *class* `ReferenceFrame`. *Class* ini dapat digunakan untuk mengkonstruksi setiap kerangka acuan bergerak yang ditempatkan pada setiap sambungan melalui penggunaan metode `.orientnew()`. Selanjutnya, matriks kosinus arah dapat diperoleh dengan menggunakan metode `.dcm()`, yang sangat bermanfaat ketika menentukan posisi *screw* dan arah sumbu *screw* terhadap kerangka acuan tetap.

Tugas utama pada langkah ke nol ini adalah untuk memperoleh tiga parameter yaitu $\mathbf{s}_{ij}$, $\mathbf{r}_{ij}$, dan h_{ij} untuk setiap sambungan pada masing-masing kaki manipulator, dengan $i = 1, \ldots, n$ dan $j = 1, \ldots, m$. n dan m masing-masingnya merupakan jumlah kaki manipulator dan jumlah. $\mathbf{s}_{ij}$ dan $\mathbf{r}_{ij}$ adalah vektor kolom berukuran 3×1 yang telah didefinisikan kerangka acuan tetap. Penulis menggunakan kondisi berikut dalam merepresentasikan $\mathbf{s}_{ij}$ dan $\mathbf{r}_{ij}$ pada kerangka acuan tetap yang selalu ditempatkan pada *base*. Jika salah satu sumbu kerangka acuan tetap, misal sumbu *X*, adalah paralel terhadap sumbu sambungan ke-*ij*, maka sumbu *x* pada kerangka acuan bergerak akan paralel terhadap sumbu *X*. Konsekuensinya adalah sumbu *y* pada kerangka acuan bergerak akan segaris dengan batang serta mengikuti urutan *XYZ*, *YZX*, dan *ZXY*.

Parameter terakhir adalah *pitch screw*, h_{ij}, yang dapat bernilai 0 atau ∞, ∞ dapat didefiniskan dengan `sympy.oo`, untuk masing-masing sambungan R dan P. Ketiga parameter ini, $\mathbf{s}_{ij}$, $\mathbf{r}_{ij}$, dan h_{ij}, untuk setiap sambungan dinyatakan ke dalam tipe data `list`. Selanjutnya, keseluruhan sambungan pada masing-masing kaki manipulator juga ditempatkan kembali ke dalam sebuah `list`. Akhirnya, untuk masing-masing kaki manipulator kembali dinyatakan ke dalam tipe data `list` sebagai luaran dari langkah ke nol ini.

```
In[q]:
# Langkah kenol berbeda untuk setiap manipulator, sesuai dengan konfigurasi geometri
def PM_xxx():
    # Transformasi sij dan rij dilakukan pada baris-baris ini dan juga hij.
    return #tiga parameter: sij, rij, hij

In[q+1]:
Mobility_PM_xxx = ManipulatorMobility(PM_xxx())

In[q+2]:
Mobility_PM_xxx.Mobility

Out[q+2]:
{'Number': 4, 'Type': '3T1R', 'Direction': ['Tx', 'Ty', 'Tz', 'Rz']}

In[q+3]:
print("Limb Twist System")
print("=================")
for ii, item in enumerate(Mobility_PM_xxx.LimbTwistSystem):
    print("Limb: ", ii+1)
    display(item.screw_system_list)
print("\n\nLimb Wrench System")
print("==================")
for ii, item in enumerate(Mobility_PM_xxx.LimbWrenchSystem):
    print("Limb: ", ii+1)
    display(item.screw_system_list)
print("\n\nPlatform Wrench System")
print("======================")
display(Mobility_PM_xxx.PlatformWrenchSystem.screw_system_list)
print("\n\nPlatform Twist System")
print("=====================")
display(Mobility_PM_xxx.PlatformTwistSystem.screw_system_list)

Out[q+3]:
…menampilkan LTSs, LWSs, PWS, dan PTS manipulator yang dikaji…
```

Gambar 2. Realisasi langkah ke nol hingga langkah

Mobilitas manipulator selanjutnya ditentukan dengan menggunakan *class* `ManipulatorMobility`. *Class* ini membutuhkan argumen masukan secara langsung dari luaran langkah kenol dan kemudian dinyatakan ke sebuah objek (*object*) sebagai instansiasi dari *class* bersangkutan, lihat *cell* **`In[q+1]`** pada Gambar 2. Penerapan properti `.Mobility` pada objek yang diinstansiasi dari *class* `ManipulatorMobility`. dapat digunakan untuk mengekstraksi informasi terkait mobilitas manipulator paralel yang dikaji. Mobilitas yang dihasilkan ini dinyatakan dalam jumlah, jenis, dan arah gerakan yang ditempatkan ke dalam suatu tipe data `dict` (*dictionary*), perhatikan *cell* **`In[q+2]`** dalam Gambar 2. Selanjutnya, jika ingin ditampilkan seluruh sistem *screw* yang dimiliki oleh manipulator paralel yang dikaji, yaitu LTS; LWS; PWS; dan PTS, maka *cell* **`In[q+3]`** dalam Gambar 2 dapat digunakan untuk tujuan tersebut.

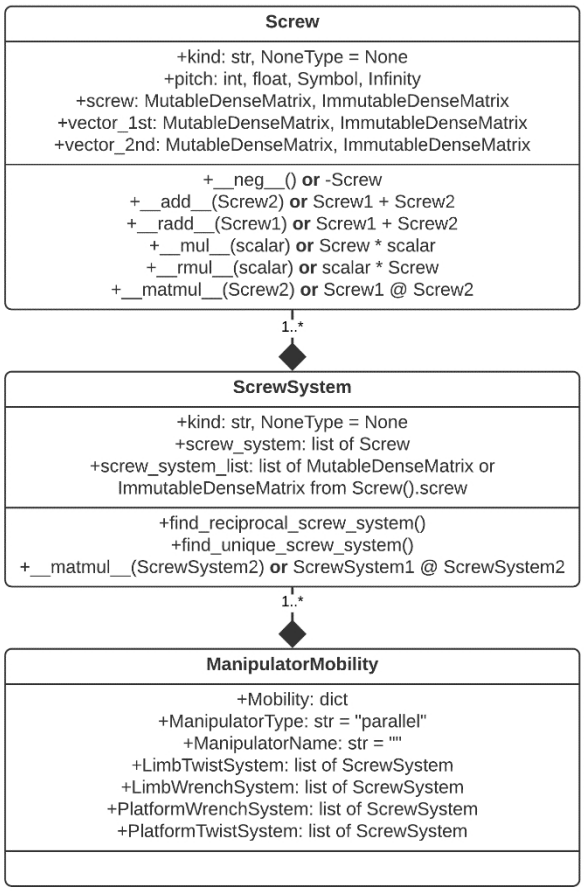
Waktu komputasi juga diukur untuk menentukan seberapa lama proses kalkulasi penentuan mobilitas dari keempat manipulator paralel berlangsung. Untuk itu, sepuluh kali perulangan diterapkan agar diperoleh rataan dan standar deviasi waktu komputasi dari masing-masing manipulator paralel. Hal ini dapat dilakukan dengan menggunakan perintah *magic* IPython (IPython *magic command*) yaitu `%%timeit -n 1 -r 10`. Modul `pyScrew4Mobility` dan *notebook* masing-masingnya dikembangkan dan dieksekusi pada sebuah notebook PC dengan prosesor Intel i5 8250 (*max. clock* 3.40 GHz) arsitektur 64-bit, memori 24 GB, pada sistem operasi LinuxMint 19.1. Anaconda3 2019.03 untuk Linux x86-64 digunakan sebagai lingkungan python untuk komputasi saintifik dalam menjalankan kode yang ditulis pada *notebook*. Anaconda3 ini juga digunakan sebagai lingkungan python untuk mengembangkan modul `pyScrew4Mobility` menggunakan IDE PyCharms 2019.3.

Modul `pyScrew4Mobility` berikut dengan *notebook*-nya ditempatkan di GitHub-nya penulis melalui tautan https://github.com/adriyan/journal_metal_2020. Keduanya dirilis dibawah lisensi 3-*clause* BSD (Berkeley Software Development).

3. HASIL DAN PEMBAHASAN

3.1. Modul `pyScrew4Mobility`

Implementasi teori *screw* untuk menentukan mobilitas atau DOF suatu manipulator paralel (langkah pertama hingga keempat pada Gambar 1) dilakukan dengan mengembangkan modul python secara pemrograman berorientasi objek (*object-oriented programming*). Modul ini dinamai dengan modul `pyScrew4Mobility` dan terdiri atas tiga *class* yaitu `Screw`, `ScrewSystem`, dan `ManipulatorMobility`, serta satu fungsi `pi_matrix`. Diagram *class unified modeling language* (UML) untuk modul `pyScrew4Mobility` diberikan melalu Gambar 3. Sementara itu, penggunaan modul `pyScrew4Mobility` berikut dengan ketiga *class*-nya diberikan secara detil pada masing-masing *docstring*-nya.



Gambar 3. Diagram *class unified modeling language* (UML) modul `pyScrew4Mobility`.

Class pertama yaitu `Screw` memiliki empat metode yang dapat digunakan untuk perhitungan *screw* secara aljabar yaitu negasi, penjumlahan, perkalian dengan skalar, dan produk *screw* yang masing-masingnya dapat dimanfaatkan melalui penggunaan operator `-`, `+`, `*`, dan `@`. Kemudian, *class* ini memiliki lima atribut/properti yaitu `.kind`, `.pitch`, `.screw`, `.vector_1st`, dan `.vector_2nd`. Properti `.kind` digunakan untuk merepresentasikan jenis *screw*, apakah *twist* atau *wrench*. Atribut `.pitch` merupakan nilai *pitch screw*. Properti `.screw` merupakan representasi *screw* sebagai vektor kolom 6 × 1, atau dapat dipisah jadi properti `.vector_1st` dan `.vector_2nd`. Dua yang terakhir ini masing-masingnya adalah tiga baris pertama dan tiga baris terakhir `.screw`. Instantiasi suatu objek dari *class* `Screw` membutuhkan (**s**, **r**, *h*) satu sambungan dalam tipe data `list` atau suatu vektor kolom ukuran 6 × 1 dengan tipe data `sympy.matrix`, dengan **s** dan **r** adalah vektor kolom 3 × 1, dan *h* merupakan suatu konstanta.

Selanjutnya, *class* `ScrewSystem` sebagai class kedua dapat diinstantiasi dengan menggunakan *list of list* (**s**, **r**, *h*) yang bersumber dari beberapa sambungan atau *list* dari suatu vektor kolom 6 × 1. Untuk manipulator paralel penggunaan *class* `ScrewSystem` adalah untuk setiap kaki manipulator. *Class* ini memiliki tiga atribut/properti yaitu `.kind`, `.screw_system`, dan `.screw_system_list`. Properti `.kind` diperuntukkan sebagai representasi dari jenis sistem *screw*, apakah sistem *twist* (*twists*) atau sistem *wrench* (*wrenches*). Atribut berikutnya yaitu `.screw_system`, dan `.screw_system_list` masing-masingnya merupakan *list* dari objek *screw* dan *list screw* yang dinyatakan dalam vektor kolom ukuran 6 × 1. Metode pada *class* ini adalah produk resiprokal dua sistem *screw* yang dapat dilakukan melalui penggunaan operator `@` yaitu `.find_reciprocal_screw_system()`, dan `.find_unique_screw_system()`.

Metode `.find_reciprocal_screw_system()` diterapkan untuk menemukan *screw* resiprokal dari

sistem *screw* yang telah diketahui dan dibuat melalui *class* `ScrewSystem`. Kemudian, metode `.find_unique_screw_system()` dapat digunakan untuk menemukan sistem *screw* unik dari sistem *screw* yang telah ada. Kasus ini dapat terjadi ketika permasalahan jumlah sambungan yang lebih dari 6 pada satu kaki manipulator (*limb*). Kasus lainnya berupa implementasi penentuan PWS dari gabungan setiap LWS.

Class ketiga dalam modul the `pyScrew4Mobility` yaitu *class* `ManipulatorMobility`. *Class* ini digunakan untuk menghitung mobilitas manipulator secara langsung yang merupakan implementasi langkah pertama hingga langkah keempat (Gambar 1). *Class* ketiga ini telah menerapkan kedua kelas sebelumnya (*class* `Screw` dan *class* `ScrewSystem`) sebagai inti untuk penentuan mobilitas manipulator paralel menggunakan teori *screw*. Dengan demikian, hanya *class* ketiga ini yang digunakan dalam menentukan mobilitas paralel manipulator dengan masukan berupa langkah ke nol yang telah didefinisikan oleh user sesuai dengan konfigurasi geometri manipulator yang dikaji.

Kemudian, untuk memperoleh mobilitas dapat dilakukan dengan menerapkan atribut/properti `.Mobility` pada objek yang telah diinstansiasi dari *class* `ManipulatorMobility`. Atribut ini akan memberikan mobilitas manipulator paralel yang dikaji dalam hal jumlah, jenis, dan arah gerak DOF dan direpresentasikan oleh oleh sebuah tipe data `dict` yang merupakan interpretasi dari PTS manipulator tersebut, lihat *cell* **`In[q+2]`** dalam Gambar 2. Melalui penggunaan objek yang telah diinstantiasi oleh *class* `ManipulatorMobility` seluruh sistem *screw* yang dimiliki oleh manipulator; yaitu LTSs, LWSs, PWS, and PTS; dapat dengan mudah diperoleh dengan mengikuti potongan kode yang diberikan pada *cell* **`In[q+3]`** dalam Gambar 2.

Akhirnya, `pi_matrix()` merupakan satu-satunya fungsi di dalam modul `pyScrew4Mobility`. Fungsi ini digunakan untuk mendefinisikan sebuah matrik berukuran 6 × 6 yang dinyatakan dengan

$$\Pi = \begin{bmatrix} \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} \\ \mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} \end{bmatrix} \tag{4}$$

dengan

$$\mathbf{0}_{3\times3} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}; \text{ dan } \mathbf{I}_{3\times3} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{5}$$

Fungsi `pi_matrix()` merupakan sebuah fungsi bantu yang dibutuhkan oleh *class* `Screw` dan `ScrewSystem` dalam proses penghitungan produk resiprokal dan menemukan *screw* resiprokal di dalam sistem *screw*.

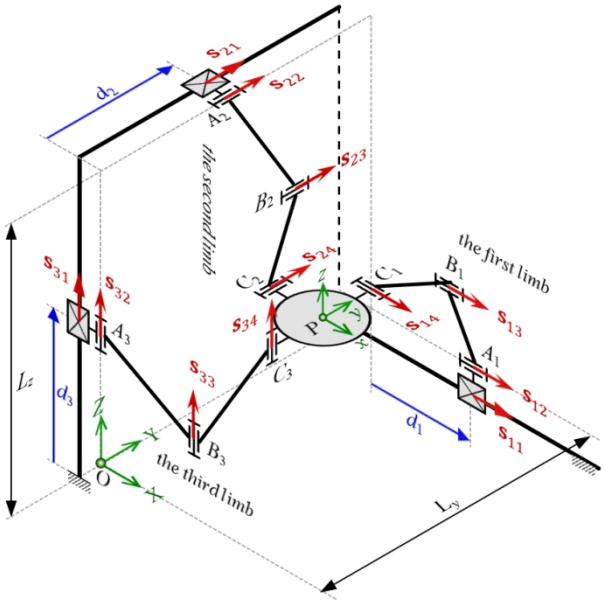
3.2. Studi Kasus: Penentuan Mobility Empat Manipulator Paralel

3.2.1. Manipulator Paralel 3-PRRR

Manipulator paralel 3-PRRR dikonstruksi oleh tiga kaki (*limb*) dengan masing-masing kaki disusun oleh rantai kinematik PRRR. Sumbu dari setiap sambungannya adalah paralel satu sama lainnya seperti yang diperlihatkan oleh Gambar 4. Manipulator ini memiliki 9 batang apung pada setiap kaki, satu batang apung sebagai *platform*, dan satu batang tetap yang menjadi *base*. Selanjutnya, manipulator ini terdiri atas 11 batang, 12 sambungan pada setiap kakinya, dan setiap sambungan hanya mengizinkan satu gerakan (satu DOF). Dengan demikian, mobilitas yang dimiliki oleh manipulator ini jika dihitung dengan mengunakan formulasi GK adalah 0 (nol). Akan tetapi, mobilitas manipulator ini adalah 3 (tiga) yang merupakan jenis translasi murni di ruang (*pure translation in the space*) [14], [15].

Untuk manipulator ini dapat dengan mudah diset vektor satuan yang menunjukkan arah sumbu di setiap sambungannya, $\mathbf{s}_{ij}$, untuk i = 1, 2, 3 dan j = 1, 2, 3, 4, dalam kerangka acuan tetap *O-XYZ*. Vektor satuan yang menyatakan arah setiap sumbu pada kaki pertama, kedua, dan ketiga masing-masingnya adalah $\mathbf{s}_{1j} = (1 \;\; 0 \;\; 0)^T$, $\mathbf{s}_{2j} = (0 \;\; 1 \;\; 0)^T$, and $\mathbf{s}_{3j} = (0 \;\; 0 \;\; 1)^T$. Sementara itu, vektor posisi sambungan $\mathbf{r}_{i1}$ dan $\mathbf{r}_{i2}$ adalah sama dengan vektor

$\overrightarrow{OA_i}$. Kemudian, vektor posisi sambungan $\mathbf{r}_{i3}$ dan $\mathbf{r}_{i4}$ diberikan masing-masingnya oleh $\overrightarrow{OB_i} = \overrightarrow{OA_i} + \overrightarrow{A_iB_i}$ dan $\overrightarrow{OC_i} = \overrightarrow{OA_i} + \overrightarrow{A_iB_i} + \overrightarrow{B_iC_i}$. Panjang batang $\overline{A_iB_i}$ dan batang $\overline{B_iC_i}$ masing-masingnya adalah *a* dan *b*.



Gambar 4. Manipulator paralel 3-PRRR atau Tripteron [14].

Berdasarkan *notebook* yang diilustrasikan oleh *cell* **`In[q]`** dalam Gambar 2, ketiga parameter untuk mengkonstruksi LTS manipulator 3-PRRR ini dibuatkan sebuah fungsinya yang diberi nama dengan `PM_3PRRR()`. Langkah selanjutnya dilakukan dengan menginstansiasi sebuah objek dari *class* of `ManipulatorMobility` yang bernama `Mobility_PM_3PRRR`. Mobilitas dan seleuruh sistem *screw*-nya dapat diperoleh melalui penerapan langkah seperti yang dinyatakan oleh *cell* **`In[q+2]`** dan **`In[q+3]`** dalam Gambar 2.

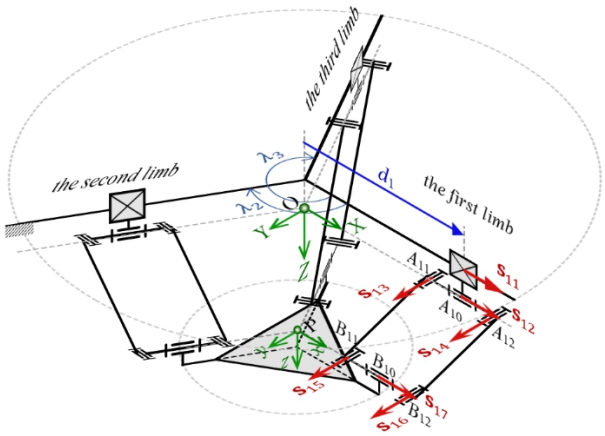
Selanjutnya, manipulator dapat dideskripsikan dengan menggunakan hasil yang diperoleh sebelumnya. Setiap kaki manipulator membentuk LTS 4-sistem yang memberikan secara langsung LWS 2-sistem. Kombinasi setiap LWS untuk ketiga kaki manipulator akan menghasilkan PWS 6-sistem. Bagaimanapun juga, PWS hanya membentuk tiga *wrench* unik. Dengan demikian, PTS-nya adalah *twist* 3-sistem yang dinyatakan secara matematis oleh

$$\begin{aligned}
\hat{\$}_1^{\text{PTS}} &= (0 \quad 0 \quad 0 \mid 1 \quad 0 \quad 0)^T, \\
\hat{\$}_2^{\text{PTS}} &= (0 \quad 0 \quad 0 \mid 0 \quad 1 \quad 0)^T, \\
\hat{\$}_3^{\text{PTS}} &= (0 \quad 0 \quad 0 \mid 0 \quad 0 \quad 1)^T.
\end{aligned} \tag{6}$$

Persamaan (4) ini merupakan hasil perhitungan yang dilakukan pada *notebook* dan dapat diakses melalui GitHub penulis. Kemudian, mengacu pada persamaan (4) dapat diketahui bahwa manipulator 3-PRRR memiliki mobilitas sebesar 3 yang sesuai dengan jumlah *twist*-nya. Setiap *twist* secara berurut mengindikasikan arah gerakan yaitu translasi dalam arah *X*, *Y*, dan *Z* yang secara langsung diindikasikan dari nilai *pitch*-nya yaitu ∞. Akhirnya, berdasarkan arah gerakan ini dapat disimpulkan bahwa manipulator 3-PRRR termasuk ke golongan manipulator paralel translasi murni di ruang (*pure translational* DOF) atau manipulator paralel 3T.

3.2.2. Manipulator Paralel 3-PR(Pa)R

Manipulator paralel 3-PR(Pa)R atau STAR [16] dikonstruksi oleh tiga kaki dengan masing-masing kakinya disusun oleh tujuh buah sambungan yaitu satu sambungan prismatik (P) dan enam sambungan revolut (R). Sabungan ketiga hingga keenam disusun secara paralel sehingga membentuk strukur paralelogram (Pa) atau suatu mekanisme yang dibentuk oleh empat batang biner yang dihubungkan dengan empat sambungan revolut dengan keempat sumbu sambungan saling sejajar satu sama lain. Sementara itu, sumbu dari sambungan pertama, kedua, dan ketujuh saling sejajar satu sama lainnya. Ketiga kaki manipulator disusun secara simetris dengan konfigurasi geometri seperti yang ditunjukkan oleh Gambar 5.



Gambar 5. Manipulator paralel 3-PR(Pa)R atau STAR [16].

Mengacu pada konfigurasi geometri manipulator 3-PR(Pa)R ini diketahui bahwa ada 21 sambungan dengan setiap satu sambungan hanya mengizinkan satu DOF. Kemudian, manipulator ini memiliki 17 batang yang terdiri dari 15 batang apung di ketiga kaki, satu *platform*, dan satu *base*. Penerapan formulasi GK berdasarkan informasi ini menghasil mobilitas sebesar -9. Bagaimanapun juga, manipulator paralel 3-PR(Pa)R memiliki mobilitas sebesar 3 dengan jenis translasi murni di ruang. Hal ini akan dibuktikan melalui penerapan teori *screw*, tentunya dengan penggunaan modul yang telah dikembangkan.

Vektor satuan pada kaki pertama yaitu $\mathbf{s}_{11}$, $\mathbf{s}_{12}$, dan $\mathbf{s}_{17}$ adalah sejajar dengan sumbu *X*, sedangkan $\mathbf{s}_{13}$, $\mathbf{s}_{14}$, $\mathbf{s}_{15}$, dan $\mathbf{s}_{16}$ saling tegak lurus dengan bidang *XZ*. Paralelogram $A_1A_2B_2B_1$ memiliki dimensi panjang $\overline{A_{11}B_{11}} = \overline{A_{12}B_{12}} = b$ dan $\overline{A_{11}A_{12}} = \overline{B_{11}B_{12}} = a$. Titik A_{10} berlokasi di titik tengah antara titik A_{11} dan A_{12}, hal ini sama juga dengan representasi titik B_{10}.

Selanjutnya, vektor posisi $\mathbf{r}_{1j}$ dapat dibangun dengan mengacu pada geometri manipulator sebagaimana yang ditunjukkan dalam Gambar 5. Misalnya dapat ditentukan bahwa $\mathbf{r}_{11} = \mathbf{r}_{12} = \overrightarrow{OA_{10}}$, $\mathbf{r}_{13} = \mathbf{r}_{10} + \overrightarrow{A_{10}A_{11}}$, $\mathbf{r}_{14} = \mathbf{r}_{10} + \overrightarrow{A_{10}A_{12}}$, $\mathbf{r}_{15} = \mathbf{r}_{13} + \overrightarrow{A_{11}B_{11}}$, $\mathbf{r}_{16} = \mathbf{r}_{14} + \overrightarrow{A_{12}B_{12}}$, dan $\mathbf{r}_{17} = \mathbf{r}_{15} + \overrightarrow{B_{11}B_{10}} = \mathbf{r}_{16} + \overrightarrow{B_{12}B_{10}}$. Setelah mengkonfigurasi kaki manipulator yang pertama, vektor $\mathbf{s}_{ij}$ dan $\mathbf{r}_{ij}$ untuk kaki kedua dan ketiga dapat dengan mudah diperoleh dengan menerapkan matriks rotasi $\mathbf{R}_{Z,\lambda_i}$ terhadap kaki pertama. Besarnya sudut λ_2 and λ_3 dapat dengan mudah diketahui dari susunan kaki yang simetris yaitu masing-masingnya sebesar 120° and 240°.

Dengan mengikuti langkah yang telah diilustrasikan dalam Gambar 2 dapat diterapkan untuk manipulator paralel 3-PR(Pa)R. Fungsi `PM_3PRPaR()` dibuatkan di dalam *notebook* yang ditujukan untuk mendeskripsikan konfigurasi geometri manipulator paralel 3-PR(Pa)R. Kemudian, fungsi ini akan menghasilkan tiga parameter yang dibutuhkan nantinya untuk penentuan mobilitas. Sebuah objek bernama `Mobility_PM_3PRPaR` digunakan untuk menginstantisi *class* `ManipulatorMobility`. Dengan demikian, mobilitas dan seluruh sistem *screw* manipulator paralel 3-PR(Pa)R dapat dengan mudah ditentukan melalui penerapan urutan kerja yang telah ditunjukkan pada *cell* **`In[q+2]`** dan **`In[q+3]`** dalam Gambar 2.

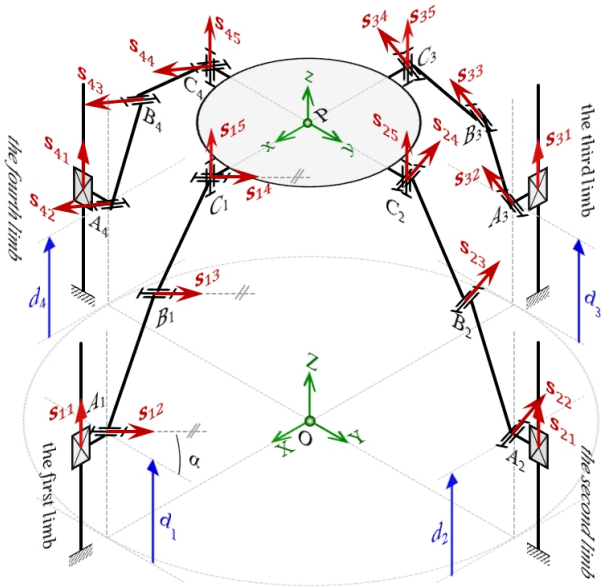
Selanjutnya, deskripsi terkait keseluruhan sistem *screw* manipulator paralel 3-PR(Pa)R dapat diperoleh dari komputasi yang telah dilakukan. Mengacu pada hasil yang diperoleh ini diketahui bahwa LTS untuk masing-masing kaki membentuk *twist* 7-sistem. Melalui pengamatan dapat diketahui bahwa LTS masing-masing kaki hanya membentuk *twist* 5-sistem, hal ini disebabkan oleh penggunaan konstruksi rantai kinematik paralelogram (RPaR) yang ekivalen dengan penggunaan rantai kinematik UU. Akan tetapi, melalui notebook LTS manipulator ini tetap *twist* 7-sistem. Kemudian, LWS setiap kaki membentuk *wrench* 1-sistem dan penggabungannya akan menghasilkan PWS sebagai *wrench* 3-sistem. Dengan demikian, PTS manipulator paralel ini tentunya akan membentuk *twist* 3-sistem yang dinyatakan secara matematis dengan

$$\begin{aligned}
\hat{\$}_1^{\text{PTS}} &= \begin{pmatrix} 0 & 0 & 0 & | & 1 & 0 & 0 \end{pmatrix}^T, \\
\hat{\$}_2^{\text{PTS}} &= \begin{pmatrix} 0 & 0 & 0 & | & 0 & 1 & 0 \end{pmatrix}^T, \\
\hat{\$}_3^{\text{PTS}} &= \begin{pmatrix} 0 & 0 & 0 & | & 0 & 0 & 1 \end{pmatrix}^T.
\end{aligned} \tag{7}$$

Dengan mengacu pada persamaa (5) secara jelas dapat diketahui bahwa manipulator ini memiliki mobilitas sebesar tiga. Gerakan *platform* manipulator ini adalah gerak translasi murni di ruang yang dikenali dari nilai *pitch twist*-nya yaitu ∞. Jadi, manipulator paralel 3-PR(Pa)R merupakan tipe manipulator translasi murni di ruang atau manipulator paralel 3T.

3.2.3. Manipulator Paralel 4-PRRU

Dengan mengacu pada konfigurasi geometris manipulator paralel 4-PRRU pada Gambar 6 diketahui bahwa manipulator ini disusun oleh 4 kaki yang masing-masingnya terdapat 4 sambungan. Sambungan terakhir yang berupa sambungan universal (U) dapat direpresentasikan sebagai dua buah sambungan revolut (R) yang saling tegak lurus. Agar manipulator ini memiliki mobilitas 4 DOF, manipulator ini harus disusun sedemikian rupa berdasarkan kondisi yang dinyatakan oleh Kong dan Gosselin, yaitu: (i) sumbu sambungan P dan sumbu sambungan R pertama tidak boleh saling tegak lurus satu sama lain, (ii) sumbu sambungan R terakhir pada setiap kaki manipulator harus sejajar satu sama lainnya, dan (iii) sumbu sambungan R kedua; ketiga; dan keempat harus saling sejajar pada kaki yang sama [17].



Gambar 6. Manipulator paralel 4-PRRU [17].

Penggunaan rantai kinematik PRRU mengindikasikan bahwa terdapat 5 gerakan yang diizinkan pada setiap kaki manipulator (*limb*). Kemudian, manipulator disusun oleh 14 batang dengan rincian 3 batang di masing-masing kaki, satu *base*, dan satu *platform*. Untuk susunan ini dapat diketahui bahwa mobilitasnya hanya 2 (dua) melalui penghitungan dengan formulasi GK. Akan tetapi, manipulator ini memiliki mobilitas sejumlah 4 yang terdiri dari 3 DOF berupa translasi dan 1 DOF rotasi atau dikenal juga dengan gerakan Schönflies.

Melalui konfigurasi geometri manipulator pada Gambar 6, vektor satuan $\mathbf{s}_{i1}$ dan $\mathbf{s}_{i5}$ adalah paralel terhadap sumbu *Z*. Sementara, vektor satuan $\mathbf{s}_{i2}$, $\mathbf{s}_{i3}$, dan $\mathbf{s}_{i4}$ adalah paralel satu sama lain dengan $\mathbf{s}_{i2}$ memeliki orientasi sudut α terhadap bidang *XY*. Selanjutnya, vektor posisi untuk kaki pertama dinyatakan sebagai berikut, yaitu $\mathbf{r}_{11} = \mathbf{r}_{12} = \overrightarrow{OA_1}$, $\mathbf{r}_{13} = \mathbf{r}_{11} + \overrightarrow{A_1B_1}$, dan $\mathbf{r}_{14} = \mathbf{r}_{15} = \mathbf{r}_{13} + \overrightarrow{B_1C_1}$. Dengan demikian, untuk kaki kedua, ketiga, dan keempat dapat diperoleh melalui kondisi kesimetrisan manipulator ini. Untuk itu, matrik rotasi $\mathbf{R}_{Z,\lambda_i}$ diterapkan disini dengan besar λ_i untuk i = 2, 3, dan 4 masing-masingnya adalah 90°, 180°, dan 270°.

Pada tahap berikutnya, berdasarkan informasi yang telah diilustrasikan dapat didefinisikan sebuh fungsi untuk langkah kenol penentuan mobilitas manipulator paralel 4-PRRU ini yaitu `PM_4PRRU()` melalui *notebook*. Fungsi ini akan mengembalikan ketiga parameter yang akan dibutuhkan sebagi input nantinya dalam penentuan mobilitas melalui penerapan *class* `ManipulatorMobility`. Untuk itu, sebuah objek dengan nama `Mobility_PM_4PRRU` diinstansiasi dari penggunaan *class* `ManipulatorMobility`.

Selanjutnya, mengacu pada hasil yang telah diperoleh dapat dikethui bahwa LTS setiap kaki manipulator membentuk *twist* 5-sistem yang tentunya berkorelasi dengan LWS berupa *wrench* 1-sistem. Melalui penggabungan LWS dari seluruh kaki akan membentuk *wrench* 4-sistem di *platform*. Namun, hanya terdapat dua *wrench* unik (*wrench* 2-sistem) yang mengekang *platform* sebagai

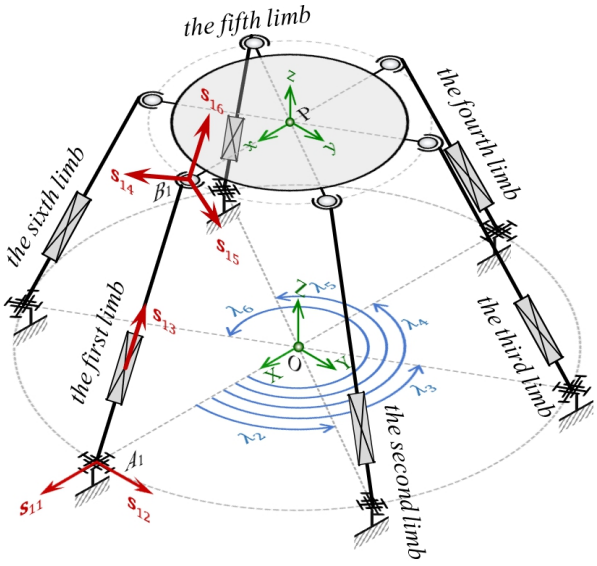
PWSnya. Dengan demikian, *platform* manipulator akan membentuk *twist* 4-sistem yang kemudian disebut dengan PTS dan diekspresikan secara matematis sebagai

$$\begin{aligned}
\hat{\$}_1^{\text{PTS}} &= \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}^T, \\
\hat{\$}_2^{\text{PTS}} &= \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}^T, \\
\hat{\$}_3^{\text{PTS}} &= \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}^T, \\
\hat{\$}_4^{\text{PTS}} &= \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}^T.
\end{aligned} \tag{8}$$

Berdasarkan PTS yang ditunjukkan oleh persamaan (6) dapat diketahui bahwa manipulator ini memiliki mobilitas sebesar 4. Selanjutnya, dengan meninjau setiap PTS dapat diperoleh informasi terkait arah gerak dari manipulator ini yaitu translasi dalam arah sumbu *X*, *Y*, dan *Z* serta satu rotasi terhadap sumbu *Z*. Dengan demikian, manipulator ini merupakan manipulator paralel bertipe 3T1R.

3.2.4. Manipulator Paralel 6-UPS

Manipulator paralel 6-UPS [18], [19] merupakan manipulator paralel simmetris dengan enam kaki (*limb*). Manipulator ini disusun oleh 3 sambungan di masing-masing kakinya dan memiliki 14 batang yang terdiri atas 2 batang di setiap kaki, satu *base*, dan satu *platform* seperti yang ditunjukkan oleh Gambar 7. Dengan mengacu pada seluruh sambungan di setiap kaki manipulator menunjukkan bahwa semua sambungan mengizinkan adanya 6 gerakan. Selanjutnya, melalui informasi yang telah dipaparkan ini dapat ditentukan mobilitas manipulator 6-UPS dengan menggunakan formulasi GK yaitu sebesar 6.



Gambar 7. Manipulator paralel 6-UPS [18], [19].

Penentuan mobilitas ini dengan menggunakan teori *screw* dapat diawali dari konfigurasi pada kaki pertama melalui vektor satuan $\mathbf{s}_{1j}$ dan vektor posisi $\mathbf{r}_{1j}$. Vektor satuan $\mathbf{s}_{11}$ dan $\mathbf{s}_{12}$ adalah paralel masing-masingnya terhadap sumbu *X* dan *Y*. Kemudian, vektor satuan $\mathbf{s}_{13}$ adalah sejajar dengan gerakan sambungan prismatik. Akhirnya, vektor satuan $\mathbf{s}_{16}$ ditempatkan secara paralel dengan $\mathbf{s}_{13}$, sedangkan $\mathbf{s}_{14}$; $\mathbf{s}_{15}$; dan $\mathbf{s}_{16}$ harus saling tegak lurus satu sama lainnya. Untuk vektor posisi $\mathbf{r}_{1j}$ dapat didefinisikan sebagai berikut yaitu $\mathbf{r}_{11} = \mathbf{r}_{12} = \overrightarrow{OA_1}$, $\mathbf{r}_{14} = \mathbf{r}_{15} = \mathbf{r}_{16} = \mathbf{r}_{11} + \overrightarrow{A_1B_1}$, dan $\mathbf{r}_{13}$ tidak direpresentasikan, karena tidak diperlukan akibat sambungan prismatik yang memiliki *pitch* ∞.

Kemudian, vektor satuan $\mathbf{s}_{ij}$ dan vektor posisi $\mathbf{r}_{ij}$ untuk kaki lainnya dapat ditentukan dengan menerapkan matriks rotasi $\mathbf{R}_{Z,\lambda_i}$ terhadap kaki pertama, dengan λ_i, untuk i = 2; 3; …; 6 secara berurutan adalah 60°; 120°; 180°; 240°; dan 300°. Di dalam *notebook*-nya, langkah kenol ini dinyatakan ke dalam sebuah fungsi bernama `PM_6UPS()`. Fungsi ini kemudian dipanggil ketika menginstansiasi sebuah objek yang dinamai dengan `Mobility_PM_6UPS` melalui penggunaan *class* `ManipulatorMobility`. Sebagai luarannya dapat diperoleh mobilitas dan semua sistem *screw* pada manipulator 6-UPS ini seperti pola umum yang telah dinyatakan dalam *cell* **`In[q+2]`** dan **`In[q+3]`** pada Gambar 2.

Dengan mengacu pada hasil yang diperoleh diketahui bahwa LTS di setiap kaki membentuk *twist* 6-sistem. *Twist* 6-sistem di setiap kaki ini merupakan *twist* unik, sehingga konsekuensinya LWS di setiap kaki merupakan *wrench* 0-sistem. Dengan menggabungkan keseluruhan *wrench* pada setiap kaki manipulator tentunya *wrench* yang dterbentuk sebagai PWS juga merupakan *wrench* 0-sistem. Kondisi ini berarti bahwa *platform* tidak mengalami konstrain oleh gaya ataupun momen. Berdasarkan PWS 0-sistem ini maka dapat ditentukan PTS-nya yang berupa *twist* 6-sistem dan secara matematis dinyatakan dengan

$$
\begin{aligned}
\hat{\$}_1^{\mathrm{PTS}} &= \begin{pmatrix} 0 & 0 & 0 & | & 1 & 0 & 0 \end{pmatrix}^T, \\
\hat{\$}_2^{\mathrm{PTS}} &= \begin{pmatrix} 0 & 0 & 0 & | & 0 & 1 & 0 \end{pmatrix}^T, \\
\hat{\$}_3^{\mathrm{PTS}} &= \begin{pmatrix} 0 & 0 & 0 & | & 0 & 0 & 1 \end{pmatrix}^T, \\
\hat{\$}_4^{\mathrm{PTS}} &= \begin{pmatrix} 1 & 0 & 0 & | & 0 & 0 & 0 \end{pmatrix}^T, \\
\hat{\$}_5^{\mathrm{PTS}} &= \begin{pmatrix} 0 & 1 & 0 & | & 0 & 0 & 0 \end{pmatrix}^T, \\
\hat{\$}_6^{\mathrm{PTS}} &= \begin{pmatrix} 0 & 0 & 1 & | & 0 & 0 & 0 \end{pmatrix}^T.
\end{aligned}
\tag{9}
$$

Persamaan (7) mengindikasikan bahwa mobilitas manipulator adalah 6 (enam). Dengan demikian, tentu saja *platform* dapat bergerak dalam 6 gerakan di ruang yaitu 3 translasi dan 3 rotasi. Selanjutnya manipulator paralel 6-UPS ini merupakan sebuah manipulator paralel bertipe 3T3R.

3.3. Waktu Komputasi dalam Penentuan Mobilitas Keempat Manipulator Paralel

Komputasi untuk setiap manipulator yang dibahas dalam artikel ini berbeda dalam hal besarnya komputasi aljabar secara simbolik pada setiap langkahnya. Waktu komputasi setiap manipulator paralel tersebut diukur untuk 10 perulangan seperti yang telah dideskripsikan pada bagian metodologi. Selanjutnya, waktu komputasi ini dapat direpresentasikan dalam nilai rata-rata (s) ± nilai standar deviasinya (ms) seperti yang diperlihatkan dalam Tabel 1.

Tabel 1. Waktu komputasi untuk proses penghitungan dalam penentuan mobilitas keempat manipulator paralel.

Manipulator Paralel	Waktu Komputasi
3-PRRR	1.16 s ± 45.1 ms
3-PR(Pa)R	5.24 s ± 147 ms
4-PRRU	9.01 s ± 329 ms
6-UPS	25.9 s ± 418 ms

Tabel 1 memperlihatkan bahwa waktu yang dibutuhkan dalam proses komputasi berbeda untuk masing-masing manipulator. Waktu komputasi secara langsung berkorelasi pada jumlah sistem *screw* dan kompleksitan geometri manipulator khususnya manipulator yang memiliki cukup banyak fungsi transenden atau trigonometri daalam representasinya. Kekomplesitasan dalam representasi aljabar simbolik setiap LTS tentunya akan mengkonsumsi waktu yang lebih lama dalam penentuan LWS. Di samping itu, penerapan fungsi bawaan pada sympy `nullspace` dalam menentukan *screw* resiprokal juga memakan waktu yang cukup lama.

4. KESIMPULAN

Modul `pyScrew4Mobility` telah dengan sukses diterapkan untuk menentukan mobilitas empat manipulator paralel, yaitu 3-PRRR, 3-PR(Pa)R, 4-PRRU, dan 6-UPS. Meskipun ini merupakan sebuah modul yang cukup kecil, tetapi modul ini dapat menyelesaikan persoalan terkait mobilitas manipulator dengan tipe *over-constrained*, seperti 3-PRRR, 3-PR(Pa)R, dan 4-PRRU. Pengguna modul ini hanya perlu mendefinisikan sebuah fungsi untuk mendeskripsikan geometri dari manipulator paralel yang akan ditentukan mobilitasnya. Luaran dari fungsi ini secara langsung akan menjadi masukan untuk *class* `ManipulatorMobility`. Kemudian, mobilitas manipulator paralel tersebut dapat diketahui dengan menerapakan atribut `.Mobility` berdasarkan objek yang diinstansiasi dari *class* `ManipulatorMobility`. Atribut ini merupakan interpretasi PTS yang selanjutnya dinyatakan dalam jumlah, jenis, dan arah gerak DOF yang dinyatakan dalam sebuah tipe data `dict`. Modul kecil ini juga memberikan sebuah pembuktian

bahwa penerapan teori *screw* untuk penentuan mobilitas dapat diprogram sebagaimana yang telah dinyatakan oleh [6].

Setelah itu, waktu komputasi untuk setiap manipulator paralel juga diukur yang mengindikasikan waktu yang diperlukan untuk proses komputasi dari langkah pertama hingga langkah keempat. Untuk kedepannya, modul `pyScrew4Mobility` akan ditujukan menjadi sebuah pustaka (*library*) atau paket (*package*) python dengan teori *screw* sebagai intinya. Proyeksi untuk pustaka/paket python ini diantaranya adalah dapat (1) menentukan sistem screw transmisi (*transmission screw system*), (2) memperoleh Jacobian, (3) memberikan kondisi singularitas berdasarkan Jacobian dan/atau transmisi gerakan/gaya (*motion/force transmission*), serta (4) analisis kinematika untuk manipulator paralel.

UCAPAN TERIMAKASIH

Penulis mengucapkan terima kasih kepada mitra bestari yang telah menelaah artikel ini secara *blind review*.

DAFTAR PUSTAKA

[1] L.-W. W. Tsai, *Robot Analysis - The Mechanics of Serial and Parallel Manipulators*. Canada: John Wiley & Sons, Inc., 1999.

[2] J. S. Dai, Z. Huang, and H. Lipkin, "Mobility of overconstrained parallel mechanisms," *J. Mech. Des. Trans. ASME*, vol. 128, no. 1, pp. 220–229, 2006, doi: 10.1115/1.1901708.

[3] G. Gogu, *Structural Synthesis of Parallel Robots Part 1: Methodology*. Springer Netherlands, 2008.

[4] Kong, Xianwen and C. M. Gosselin, "Mobility Analysis of Parallel Mechanisms Based on Screw Theory and the Concept of Equivalent Serial Kinematic Chain," in *ASME 2005 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, 2005, pp. 911–920.

[5] B. Li, J. Zhao, B. Li, X. Yang, and H. Yu, "Geometrical Method to Determine the Reciprocal Screws and Applications to Parallel Manipulators," *Robotica*, vol. 27, no. 6, pp. 929–940, 2009, doi: 10.1017/S0263574709005359.

[6] L. Wang, H. Xu, and L. Guan, "Mobility analysis of parallel mechanisms based on screw theory and mechanism topology," *Adv. Mech. Eng.*, vol. 7, no. 11, pp. 1–13, 2015, doi: 10.1177/1687814015610467.

[7] S. Robert and S. Ball, "Sir Robert Stawell Ball and Methodologies of Modern Screw Theory," *Proc. Inst. Mech. Eng. , Part C J. Mech. Eng. Sci.*, 2002, doi: 10.1243/0954406021524828.

[8] X. Kong, "Type Synthesis and Kinematics of General and Analytic Parallel Mechanisms," Université Laval, Québec, 2003.

[9] J. Gallardo-alvarado, R. Rodríguez-castro, M. Caudillo-ramírez, and L. Pérez-gonzález, "An Application of Screw Theory to the Jerk Analysis of a Two-Degrees-of-Freedom Parallel Wrist," *Robotics*, vol. 4, no. 1, pp. 50–62, 2015, doi: 10.3390/robotics4010050.

[10] X. J. Liu, C. Wu, and J. Wang, "A New Approach for Singularity Analysis and Closeness Measurement to Singularities of Parallel Manipulators," *J. Mech. Robot.*, vol. 4, no. 4, pp. 1–10, 2012, doi: 10.1115/1.4007004.

[11] D. Joyner, O. Čertík, A. Meurer, and B. E. Granger, "Open source computer algebra systems: SymPy," *ACM Commun. Comput. Algebr.*, vol. 45, no. 3/4, pp. 225–234, 2012, doi: 10.1145/2110170.2110185.

[12] A. Meurer *et al.*, "SymPy: Symbolic computing in Python," *PeerJ Comput. Sci.*, vol. 3, no. 01, p. e103, 2017, doi: 10.7717/peerj-cs.103.

[13] Adriyan and Sufiyanto, "Kinematic and Singularity Analysis of PRoM-120 – A Parallel Robotic Manipulator with 2-PRU/PRS Kinematic Chains," *Rekayasa Mesin*, vol. 9, no. 3, pp. 201–209, 2018, doi: 10.21776/ub.jrm.2018.009.03.7.

[14] H. S. Kim and L. Tsai, "Design Optimization of a Cartesian Parallel Manipulator," *J. Mech. Des.*, vol. 125, no. March, pp. 43–51, 2003, doi: 10.1115/1.1543977.

[15] Clement Gosselin and Xianwen Kong, "Cartesian parallel manipulators," US Patent 6 729 202, 2004.

[16] J. M. Hervé and F. Sparacino, "Star, a New

Concept in Robotics," in *Third International Workshop on Advances in Robot Kinematics*, 1992, no. July, pp. 176–183.

[17] X. Kong and C. Gosselin, "Parallel Manipulators with Four Degrees of Freedom," US 6,997,669 B2, 2006.

[18] Y. Zhang and Y. Yao, "Kinematic Optimal Design of 6-UPS Parallel Manipulator," in *2006 International Conference on Mechatronics and Automation*, Jun. 2006, pp. 2341–2345, doi: 10.1109/ICMA.2006.257697.

[19] Y. Liu, H. Wu, Y. Yang, S. Zou, X. Zhang, and Y. Wang, "Symmetrical Workspace of 6-UPS Parallel Robot Using Tilt and Torsion Angles," vol. 2018, 2018.

[20] T. Kluyver *et al.*, "Jupyter Notebooks—a publishing format for reproducible computational workflows," in *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, 2016, pp. 87–90, doi: 10.3233/978-1-61499-649-1-87.