

VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
PROGRAMŲ SISTEMŲ KATEDRA

Bakalauro baigiamasis darbas

Autonominis ketursraigčio skrydžio valdymas
Autonomous Control of Quadcopter Flight

Atliko: 4 kurso 1 grupės studentas
Rytis Karpuška
(parašas)

Darbo vadovas:
Irus Grinis, lekt.
(parašas)

Recenzentas:
Vytautas Valaitis, lekt.
(parašas)

Vilnius 2015

Santrauka

Šio darbo metu buvo sukurti ir išbandyti algoritmai ketursraigčio autonominiam skrydžiui palaikyti, bandymus atliekant su realiu ketursraigčiu. Ši užduotis buvo išskaidyta į tris dalis: kampinės pozicijos įvertinimas, kampinės pozicijos valdymas bei abosliučios pozicijos valdymas. Kampinė pozicija yra skaičiuojama pasitelkiant kvaternionus, šitaip efektyviai atliekant skaičiavimus įterptinės sistemos aplinkoje bei išvengiant užrakinimo (ang.: *gimbal lock*) problemos. Kampinės pozicijos valdymas atliekamas naudojant du PID nuosekliai sujungtus valdiklius. O absoliuti pozicija valdoma atviro-ciklo (ang.: *open-loop*) valdiklio, kuriam yra saugoma tam tikrų kampinių pozicijų lentelė ir jos duomenys yra persiunčiami kampinės pozicijos valdikliams tam tikru laiko momentu.

Raktiniai žodžiai. Ketursraigtis, inercinis matavimo įrenginys, kvaternionai, kampinė pozicija, PID, uždaro-ciklo valdiklis, atviro-ciklo valdiklis, autonominis

Summary

During this thesis algorithms for autonomous quadcopter control has been developed and tested in a real environment. Autonomus control of quadcopter has been divided into three parts: angular position estimation, angular position control and absolute position control. Angular position estimation is being performed using quaternions in order to exprress 3D rotations efficiently and to avoid gimbal lock problem. Angular position control is achieved using two PID controllers connected in series. And absoulte position control is achieved by using open-loop controller where a table of certain angular positions are saved and at aproprate times sent to the angular position controller algorithms.

Keywords. Quadcopter, inertial measurement unit, quaternion, attitude, PID, closed-loop controller, open-loop controller, autonomous

Turinys

Santrauka	1
Summary	2
Įvadas	4
1. Ketursraigčio techninė įranga	7
1.1. Rėmas, varikliai ir propeleriai	7
1.2. Valdymo elektronika	8
2. Matematinis skrydžio modelis	9
2.1. Lokalioji ir globalioji koordinačių sistemos	9
2.2. Keliamoji jėga	10
2.3. Sukamoji jėga	11
2.4. Bendras judėjimo modelis	13
3. Kampinės padėties skaičiavimas	14
3.1. Kvaternionai	14
3.2. Sensoriai	17
3.3. Kampinės padėties skaičiavimas pagal giroskopą	18
3.4. Kampinės padėties skaičiavimas pagal akselerometrą	19
3.5. Galutinis kampinės padėties radimas	20
4. Kampinės padėties valdymo algoritmas	22
4.1. PID valdymo algoritmas	22
4.2. PID pritaikymas ketursraigčio valdymui	22
4.3. Pasisukimo pagal lokaliają Z ašį valdymas	25
5. Skrydžio autonomiškumas	27
5.1. Atviro-ciklo valdymas	27
5.2. Kampinės pozicijos tikslų lentelė	27
5.3. Atviro-ciklo valdymo trūkumai	27
6. Programinė įranga	29
6.1. Bendroji architektūra	29
6.2. Kompiuteriui skirtas klientas	29
6.3. Retransmitorius	29
6.4. Ketursraigčio pagrindinis valdiklis	29
Išvados	33
Literatūros sąrašas	35

Įvadas

Ketursraigčiai per keletą paskutinių metų pradėjo stipriai populiarėti, ypač akademinėje ir mėgėjų aplinkoje. Tačiau jie yra ir gali būti taikomi labai plačiose srityse, pradedant nuo pirmo-asmens vaizdo (ang.: *first-person-view*) įrenginių mėgėjų skraidymui, įvairių renginių ar įvykių filmavimu ir baigiant vaistų pristatymu sunkiai prieinamiems ligoniams ar gelbėjimo ir paieškos tipo operacijose. Tobulėjančios akumuliatorių bei elektrinių variklių technologijos leidžia išlaikyti ketursraigčių ore vis ilgiau ir užkrauti vis sunkesnius krovinius. Tačiau ketursraigčių valdymas yra per daug sudėtingas, kad pilotas galėtų vadyti ketursraigčio skrydį tiesiogiai, todėl elektroninių sistemų pagalba yra būtina. Ši sistema priima paprastesnes žmogui komandas iš piloto, pavyzdžiui „skristi pirmyn“, „skristi šonan“ ir t.t. ir paskui paverčia jas elektriniais variklių signalais. Šio darbo tikslas yra sukurti, suprogramuoti ir išbandyti tokią ketursraigčio skrydžio valdymo sistemą. Tam buvo surinktas vidutinio dydžio ketursraigtis, ir paruošta elektroninė įranga.



1 pav. Ketursraigtis.

Testavimo aplinka. Ketursraigčio valdymo algoritmų kūrimas yra labai ap-sunkinamas, dėl sudėtingų testavimo sąlygų. Neištėstavus pakeitimų kode negalima leisti ketursraigčiui laisvai skristi, nes klaidos atveju, ketursraigtis gali atsitrenkti į kliūtį, sulaužyti savo dalis, ar blogiausiu atveju, dėl daugiau kaip 10,000 sūkių per minutę besisukančių propelerių – stipriai sužeisti aplink esančius žmones ar gyvūnus.

Tačiau duomenų surinkimas apie algoritmų darbą negali įvykti, kol šie algoritmai nebuvo įdarbinti realioje aplinkoje, o dėl sudėtingos ketursraigčių dinamikos bei aerodinamikos – realistiškos simuliacijos atlikimas yra per daug sudėtinga užduotis. Susiduriama su „vištos ir kiaušinio“ problema, algoritmai negali būti tobulinami, kol jie nebuvo paleisti darbui, bet algoritmų negalima paleisti darbui, kol jie neištobulinti.

Šiai problemai spręsti buvo sumontuotas specialus stendas, apribojantis ketursraigčio judėjimo laisvę. Stendas susideda iš pakankamai sunkaus stovo, kurio ketursraigtis negalėtų nuversti ar pakelti, netgi generuodamas maksimalią traukos jėgą. Toliau yra mechanizmas užfiksuojuantis ketursraigčių stovą taip, kad ketursraigtis galėtų sukis tiksliai apie vieną ašį. Konkrečiu atveju buvo ištiesta plieninė styga, išilgai kurios buvo tvirtinamas ketursraigtis X arba Y ašimi.



2 pav. Ketursraigčio stendas.

Šis stendas išsprendė testavimo sudėtingumo problemą ir leido algoritmų testavimo metu prijungti įvairius komunikacijos ar galios laidus prie ketursraigčio, tad supaprastėjo duomenų kaupimas, derinimas, taip pat galima buvo atlikti testavimus su išoriniu maitinimo šaltiniu vietoj baterijų, o tai leido sutaupyti laiko, dėl ilgo baterijos krovimosi laiko.

Literatūros apžvalga. Paskutiniu metu ketursraigčių valdymo sistemos irgi susilaukė palyginus nemažai mokslininkų ir tyrėjų dėmesio.

[ABCO15] pateikia kampinės padėties skaičiavimus pagal inercinius ir magnetinius sensorius, paaiškina sensorių savybes, ir pasiūlo oilerio kampais grystą kampinės pozicijos įvertinimo būdą. Taip pat sumodeliuoja kalmano filtrą giroskopo dreifavimo problemai spręsti.

Straipsnyje [KJ04] autoriai pateikia sensorių modelį, bei kalibravimo procedūras, nors jų IMU nebuvo taikomas ketursraigčiuose. Taip pat pasiūlomas kampinės padėties įvertinimo algoritmas naudojant kalmano filtrą su statiniu stiprinimo koeficientu.

Papildomasis (ang.: *complementary*) filtras panaudojamas [ECM⁺08] skaičiuojant kampinę poziciją fiksuoto sparno lėktuvui. Atlikti tyrimai parodo akselerometro triukšmo lygius, bei filtravimo būdus.

Teorinė ketursraigčio valdymo sistema analizuojama [Gib10], kartu su fizikinių ketursraigčio modeliu, bei atliekama simuliacija. [Gib10] buvo remtasi išvedant fizikinį modelį šiame darbe.

[MS11] pateikia valdymo sistemą sistemą valdomą FPGA valdiklio.

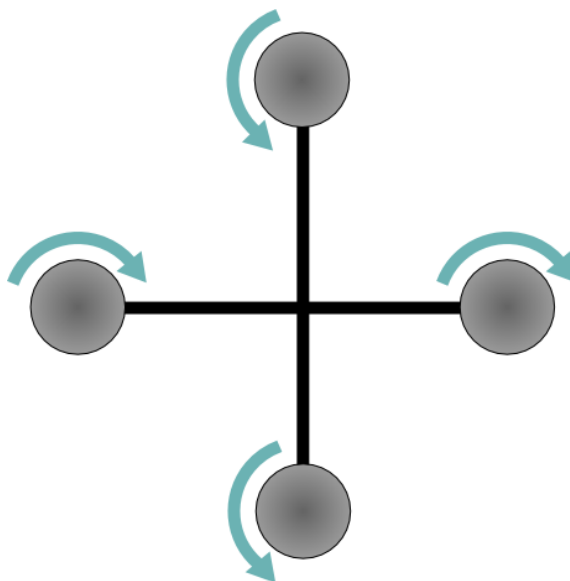
Nors FPGA ir gali pasiūlyti greitesnį duomenų apdorojimą, tačiau [MS11] kontroleris dirbo 100Hz dažniu, tuo tarpu, kai šiame darbe atnaujinimai vykdomi 500Hz dažniu, nors ir naudojamas ne FPGA, bet ARM procesorius. Taip pat [MS11] naudoja oilerio kampus kampinės pozicijos skaičiavimui, todėl kampinė pozicija potencialiai gali užsirakinti (ang.: „gimbal lock“). Skrydžio kampinės padėties valdymo algoritmui buvo naudojamas P valdiklis ir kampinės padėties greičio kompensavimas, tačiau išvadose rašoma, kad ore ketursraigtis negalėjo išsilaikyti. Tačiau [MS11] atliko labai naudingus ESC tiesiškumo, traukos bei sukimo momento priklausomybių empirinius testus.

1. Ketursraigčio techninė įranga

Įprasto ketursraigčio techninė struktūra yra palyginus lengvai suprojektuojama bei pagaminama, tačiau to negalima pasakyti apie programinę įrangą, bei algoritmus valdančius skrydį.

1.1. Rėmas, varikliai ir propeleriai

Ketursraigtis susideda iš „X“ formos rėmo, kurio galuose yra po elektrinį bešepetėlinį variklį su propeleriu. Priešingai nei įprastuose sraigtasparniuose, šių propelerių atakos kampas nėra reguliuojamas, o tai leidžia stipriai supaprastinti skraidyklės techninę struktūrą ir atsisakyti sudėtingų mechaninių dalių. Varikliai skirstomi į dvi grupes iš kurių viena sukasi pagal laikrodžio rodyklę, kita – prieš laikrodžio rodyklę. Šių variklių sukimosi greitis yra reguliuojamas siekiant išgauti tinkamą sukamąją bei keliamąją jėgas.



3 pav. Ketursraigčio rėmo ir variklių išdėstymas. Jų sukimosi kryptys.

Šio darbo tikslams pasiekti buvo nupirktas „HobbyKing x525“ 600 mm skersmens rėmas, pagamintas iš aliuminio ir stiklo pluošto. Taip pat elektriniai bešepetėliniai 160 W galios varikliai „Turnigy D2822“ ir 8 colių ilgio, bei 4 laipsnių atakos kampo propeleriai.

1.2. Valdymo elektronika

Stabilaus skrydžio išlaikymas ketursraigtyje yra per sudėtinga užduotis žmogui (pilotui) todėl pasitelkiama pagalbinė elektronika palengvinanti ketursraigčio valdymą.

Valdymo elektronikai išskiriami tokie uždaviniai:

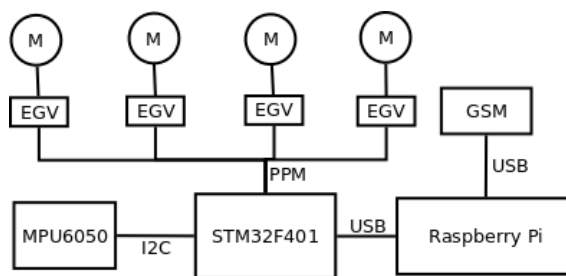
- skrydžio stabilizavimas bei kontrolė;
- ryšio su pilotu arba valdančia sistema palaikymas;
- galios signalų, reikalingų varikliams, generavimas.

Skrydžio stabilizavimas bei kontrolė. Skrydžio stabilizavimui bei kontrolei atlikti naudojami sensoriai pagal kurių duomenis yra paskaičiuojami kokių korekcinų veiksmų reikia imtis norint įgyvendinti piloto ar valdančios sistemos komandas. Išskiriami svarbiausi parametrai yra tikslumas bei greitis.

Atsižvelgiant į reikalavimus šioms parametrų, buvo parinktas kompanijos „STMicroelectronics“ procesorius „STM32F401“ bei kompanijos „InvenSense“ sensorius „MPU6050“

Ryšio su pilotu arba valdančia sistema palaikymas. Ryšio palaikymas parametrizuojamas pagal duomenų persiuntimo greitį ir uždelimą, bei veikimo ribas. Ketursraigčio atveju persiunčiami duomenys yra tik valdymo signalai iš piloto arba valdančios sistemos, todėl buvo pasirinktas GSM ryšys. Taip pat „Raspberry pi“ kompiuteris su „Linux“ operacine sistema atlikdavo viską kas reikalinga GSM ryšiui palaikyti.

Galios signalų generavimas. Bešepetėliniai varikliai reikalauja trijų galios signalų jų sukimuisi palaikyti. Šiam tiklui buvo nupirkti 18 A elektroniniai greičio valdikliai galintys suvaldyti apie 200 W galios, tad puikiai tinkantys 160 W galios varikliams.

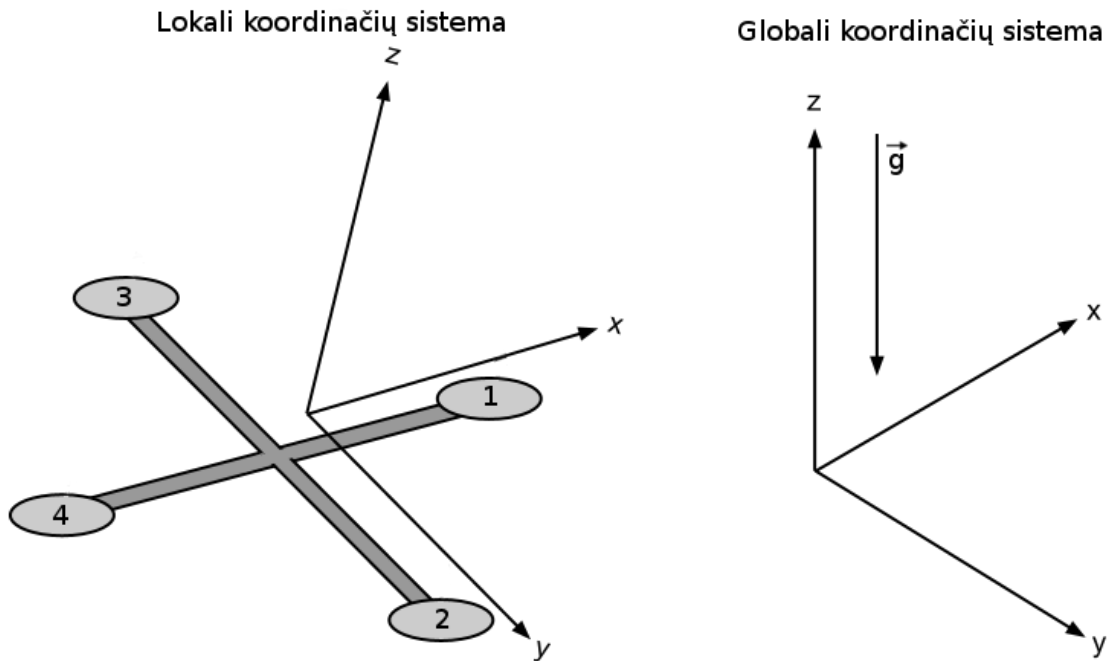


4 pav. Ketursraigčio elektronikos sudedamųjų dalių schema.

2. Matematinis skrydžio modelis

2.1. Lokalioji ir globalioji koordinatinių sistemų

Pravartu apibrėžti lokaliają ir globaliąją koordinatinių sistemas, kuriose bus nagrinėjama ketursraigčio dinamika. Lokaliojoje koordinatinių sistemoje X ir Y ašis yra sulygiuota su ketursraigčio rėmo strypais, kur X rodo pirmojo variklio kryptimi, Y - antrojo. Globalioji sistema yra susieta su žemės gravitaciniu lauku, ir toje sistemoje gravitacinio lauko vektorius nukreiptas priešinga Z ašiai kryptimi.



5 pav. Lokalioji ir globalioji koordinatinių sistemų palyginimas.

Konvertavimas tarp šių koordinatinių sistemų bus vykdomas kvaternionų pagalba (žr.: 3.1):

$$q_l = q_p * q_g * q_p^{-1} \quad (1)$$

ir į priešingą pusę:

$$q_g = q_p^{-1} * q_l * q_p \quad (2)$$

Čia q_g – tašką arba vektorių globaliojoje koordinatinių sistemoje atvaizduojantis

kvaternionas, q_l – tašką arba vektorių lokolioje koordinačių sistemoje atvaizduojantis kvaternionas, q_p – kampinės pozicijos kvaternionas (žr.: 3.1).

2.2. Keliamoji jėga

Ketursaitis sukuria keliamąją jėgą priversdamas judėti orą žemyn. Kadangi naudojami nekintamo atakos kampo propeleriai, keliamoji jėga reguliuojama valdant propelerių sukimosi greitį. pagal [Gib10, MS11] keliamoji jėga gali būti apskaičiuota:

$$T = C * \omega^2 \quad (3)$$

Kur T yra keliamoji jėga niutonais vienam varikliui, C yra konstanta priklausanti tik nuo propelerio nekintamų savybių, o ω yra variklių sukimosi greitis.

Taip pat galime apskaičiuoti bendrąją keliamąją jėgą visiems varikliams.

$$T = \begin{bmatrix} C_1 * \omega_1^2 - C_4 * \omega_4^2 \\ C_2 * \omega_2^2 - C_3 * \omega_3^2 \\ \sum_{i=1}^4 C_i * \omega_i^2 \end{bmatrix} \quad (4)$$

Šiuo atveju T yra keliamosios galios vektorių lokolioje koordinačių sistemoje. Verta pastebėti, kad esant nevienodiems ω_1, ω_4 arba ω_2, ω_3 sukuriamą jėgą sukanti ketursaitį apie jo lokalią Y arba X ašį atitinkamai.

O skaičiuojant globaliojoje koordinačių sistemoje:

$$q_{tl} = \begin{bmatrix} 0 \\ T_x \\ T_y \\ T_z \end{bmatrix} \quad (5)$$

$$q_{tg} = q_p^{-1} * q_{tl} * q_p \quad (6)$$

$$T_g = \begin{bmatrix} q_{tgx} \\ q_{tgy} \\ q_{tgz} \end{bmatrix} \quad (7)$$

Čia q_{tl} yra kvaternionas, reprezentuojantis keliamosios jėgos vektorių lokolioje koordinačių sistemoje, q_p kvaternionas reprezentuojantis pasukimą nuo globaliosios iki lokalsios koordinačių sistemos (žr.: 3), q_{tg} – keliamosios jėgos vektorių atvaizduojantis kvaternionas globaliojoje koordinačių sistemoje, T_g – keliamosios jė-

gos vektorius globaliojoje koordinačių sistemoje.

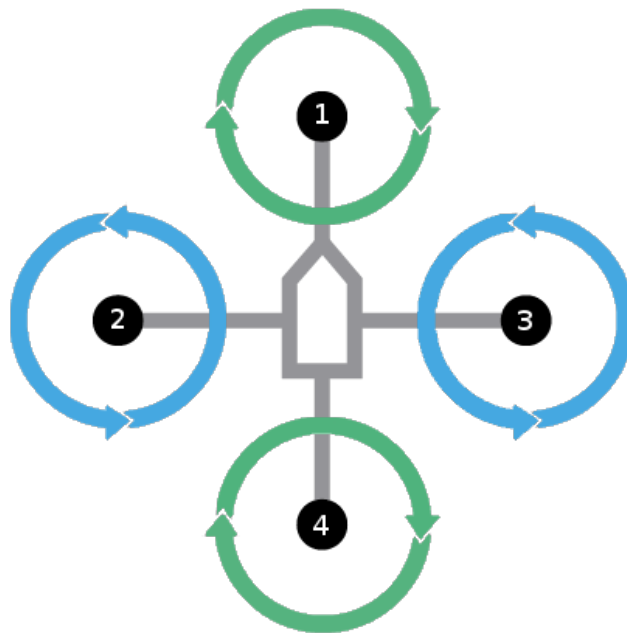
2.3. Sukamoji jėga

Besisukant propeleriams oro trintis į mentes sukelia papildomą jėgą, kuri veikia ketursraigčio rėmą. Kiekvienam iš variklių ši jėga yra nukreipta priešinga kryptimi nei sukasi variklis. Norint išvengti nevaldomo ketursraigčio sukimosi, parenkami du propeleriai skirti suktis pagal laikrodžio rodyklę ir du propeleriai skirti suktis prieš laikrodžio rodyklę, varikliai sujungiami taip, kad jų sukimosi kryptis atitiktų propelerį ir oro srautas būtų nukreiptas žemyn. Konkrečiu atveju varikliai 1 ir 4 sukasi prieš laikrodžio rodyklę, o varikliai 2 ir 3 – pagal. Šitaip vienos krypties jėga kompensuoja kitą ir tampa įmanomas skrydis be sukimosi apie Z ašį lokalioje koordinačių sistemoje.

Šią jėgą modeliuosime tiesiškai priklausančią nuo sukimosi greičio:

$$F_{tq} = -\omega * C \quad (8)$$

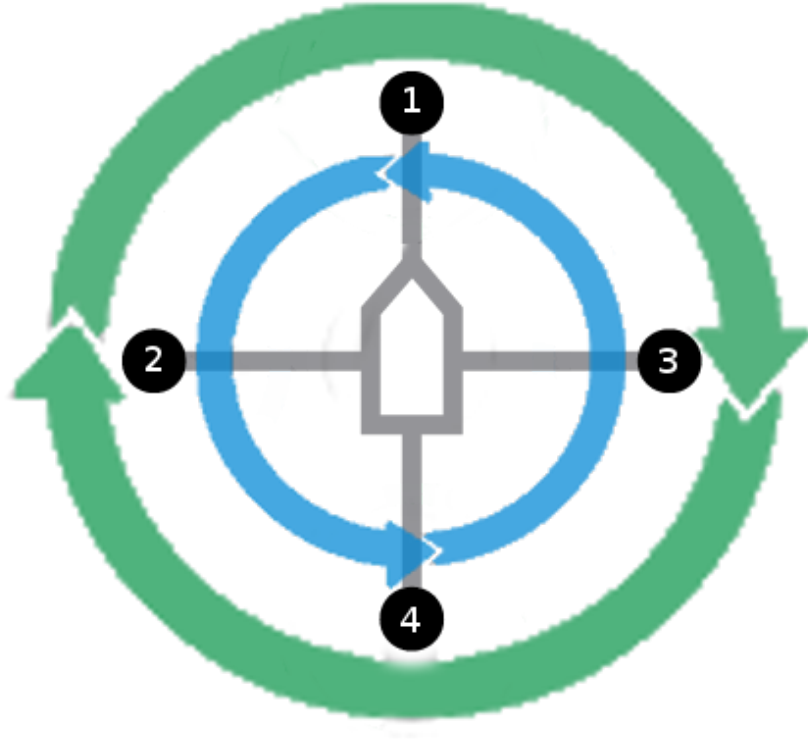
Čia C yra konstanta priklausanti tik nuo propelerio savybių. Sukamoji jėga yra sukeliamą oro pasipriešinimo, todėl ji yra priešinga judėjimo kryptčiai ω .



6 pav. Jėgos sukeliamos dėl oro trinties į propelerius.

Verta pastebėti, kad varikliai skaidomi grupėmis pagal sukimosi kryptį. Tos

pačios sukimosi krypties variklis sukuria tos pačios krypties sukamąją jėgą, tik centras kitoje vietoje. Tačiau rėmo atžvilgiu šias grupes galima nagrinėti kaip vieną jėgą. Tokiu atveju turime dvi jėgas priešingų krypčių (žr.: pav. 7).



7 pav. Jėgos sukkeliamos dėl oro trinties į propelerius.

Šiuo atveju vidinės rodyklės rodo 2 ir 3 variklių sukamąsias jėgas, o išorinės – 1 ir 4 (žr.: pav. 7). Verta pastebėti, kad rodyklių storis nėra susijęs su jėgos dydžiu.

Sukamosios jėgos neturi vieningos krypties, kuri nepriklausytų nuo stebėjimo pozicijos, todėl ji bus išreikšta ne vektorine forma, taigi nagrinėsime sukamosios jėgos dydį ties varikliais. Taip pat, sakysime, kad jėga yra teigiama kai sukamasi apie lokalią Z ašį pagal laikrodžio rodyklę. Tada gauname, kad sukamoji jėga:

$$F_{tq1,4} = C * (\omega_1 + \omega_4) \quad (9)$$

$$F_{tq2,3} = -C * (\omega_2 + \omega_3) \quad (10)$$

Ir bendra jėga:

$$F_{tq} = F_{tq1,4} - F_{tq2,3} = C * (\omega_1 - \omega_2 - \omega_3 + \omega_4) \quad (11)$$

Šias jėgų kryptis atitinkamai galime pasukti priešingu kvaternionu q_p , ir gausime jėgą globaliojoje koordinačių sistemoje. Verta pastebėti, kad šių jėgų kryptys visada bus lygiagračios su propelerių plokštuma.

2.4. Bendras judėjimo modelis

Ketursraigčio skrydžio judėjimui apibrėžti reikia įsivesti gravitacijos kryptį.

$$g_g = \begin{bmatrix} 0 \\ 0 \\ -9.8 \end{bmatrix} \quad (12)$$

g_g yra apibrėžtas globalioje koordinačių sistemoje.

Tuomet bendra jėga veikianti ketursraigį:

$$F_g = T_g - m * g_g \quad (13)$$

Čia m yra ketursraigčio bendra masė.

Toliau pagal antrąjį Niutono dėsnį:

$$a_g = \frac{F_g}{m} \quad (14)$$

kur a yra pagreičio vektorius globalioje koordinačių sistemoje.

Toliau randamas greičio bei pozicijos priklausomybė nuo nykstamai mažo laiko pokyčio Δt :

$$v_g = a_g * \Delta t \quad (15)$$

$$x_g = a_g * \Delta t^2 \quad (16)$$

Tais pačiais principais randama ir kampinė padėtis.

3. Kampinės padėties skaičiavimas

Ketursraigčio valdymo algoritmai yra tiesiogiai priklausomi nuo kampinės padėties, bet nėra sensorių leidžiančių tai išmatuoti tiesiogiai, todėl šiame skyrelyje apibūdinami sukurti principai, pagal kuriuos yra skaičiuojama ketursraigčio kampinė pozicija globaliosios koordinatų sistemos atžvilgiu.

Ketursraigčių kampinės padėties skaičiavimui buvo parinkta matematinė skaičiavimo sistema vadinama kvaternionais. Šioje sistemoje trimačiai posūkiai gali būti atvaizduoti labai efektyviai ir patogiai.

3.1. Kvaternionai

Kvaternionai matematikoje yra skaičių sistema, kuri išplėčia įprastus kompleksinius skaičius. Kvaternionas yra vektorius:

$$q = \begin{bmatrix} w \\ i \\ j \\ k \end{bmatrix} \quad (17)$$

kur w yra sveikoji dalis, o i, j, k , yra menamosios.

Menamosios dalys tarpusavyje siejamos pagal:

$$i^2 = j^2 = k^2 = ijk = -1 \quad (18)$$

Dauginimui kiekvieno elemento su kiekvienu naudojamas toks sąryšis:

*	1	i	j	k
1	1	i	j	k
i	i	-1	k	-j
j	j	-k	-1	i
k	k	j	-i	-1

Įprastas kvaternionas užrašomas kaip šių elementų suma:

$$q = w + xi + yj + zk \quad (19)$$

kur w, x, y, z yra realioji skaičiai, o i, j, k yra menamosios kvaterniono dalys.

Šiame darbe kvaternionai bus nagrinėjami naudojant vektorine forma:

$$q = \begin{bmatrix} w \\ xi \\ yj \\ zk \end{bmatrix} \quad (20)$$

Šio vektoriaus atskiras komponentes bus žymimos q_w, q_x, q_y, q_z .

Daugyba. Kvaternionų daugyba atliekama pagal Hamiltono taisykles:

TODO sužiūrėti newlinenus

$$q_1 * q_2 = [w_1, x_1 i_1, y_1 j_1, z_1 k_1] * [w_1, x_1 i_1, y_1 j_1, z_1 k_1] = q_1 * q_2 = \begin{bmatrix} w_1 w_2 - x_1 x_2 - y_1 y_2 - z_1 z_2 \\ (w_1 x_2 + x_1 w_2 + y_1 z_2 - z_1 y_2) i \\ (w_1 y_1 - x_1 z_2 + y_1 w_1 + z_1 x_1) j \\ (w_1 z_1 + x_1 y_2 - y_1 x_1 + z_1 w_1) k \end{bmatrix} \quad (21)$$

Invertavimas. Invertuotas kvaternionas yra kvaternionas atvaizduojantis pasukimą į priešingą pusę ir žymimas:

$$q^{-1} \quad (22)$$

Invertuoti kvaternioną galima dvejopai, arba invertuojant realiąją dalį

$$q^{-1} = \begin{bmatrix} -q_w \\ q_x \\ q_y \\ q_z \end{bmatrix} \quad (23)$$

arba invertuojant visus tris menamuosius daugiklius

$$q^{-1} = \begin{bmatrix} q_w \\ -q_x \\ -q_y \\ -q_z \end{bmatrix} \quad (24)$$

Šis metodas ir bus naudojamas toliau.

Taško erdvėje atvaizdavimas kvaternionu. Taškas erdvėje pagal jo X , Y , Z koordinates kvaternionu atvaizduojamas taip:

$$q = \begin{bmatrix} 0 \\ Xi \\ Yj \\ Zk \end{bmatrix} \quad (25)$$

Erdvinio pasukimo atvaizdavimas kvaternionu. Žinant kampą ϕ ir ašies kryptį (X, Y, Z) , aplink kurią sukuama, kvaternioną atvaizduojamas taip:

$$q = \begin{bmatrix} \cos(\frac{\phi}{2}) \\ X \sin(\frac{\phi}{2})i \\ Y \sin(\frac{\phi}{2})j \\ Z \sin(\frac{\phi}{2})k \end{bmatrix} \quad (26)$$

Po formulėje 26 kvaterniono suformavimo jis turi būti normalizuotas.

Tuomet taško erdvėje pozicija po pasukimo kvaternionu q randama taip:

$$q_{tp} = q * q_t * q^{-1} \quad (27)$$

Čia q_{tp} yra tašką atvaizduojantis kvaternionas po pasukimo, q_t yra tašką atvaizduojantis kvaternions prieš pasukimą.

Funkcija *Lerp*. Šios funkcijos pavadinimas yra trumpinys tiesinei interpoliacijai (ang.: *Linear Interpolation*). Trimačių pasukimo kontekste, *Lerp* yra aproksimacija *Slerp* (ang.: *Spherical Interpolation*).

Darant prielaidą, kad kvaternionas išreiškia pasukimą taip kaip nurodoma pastraipoje „Erdvinio pasukimo atvaizdavimas kvaternionu“, tuomet funkcija *Slerp* išreiškiama taip:

$$Slerp(q_0, q_1, t) = (q_1 q_0^{-1})^t q_0 \quad (28)$$

Slerp funkcija randa kvaternioną atvaizduojantį pasukimą proporcingą pateiktam skaliarui su ribomis nuo pirmo pateikto kvaterniono iki antro. Pavyzdžiui jeigu turime kvaternioną q_0 sukanti 90° dešinėn, ir kvaternioną q_1 sukantį 90° kairėn, tai gautume tokius rezultatus priklausomai nuo skaliaro t

skaliaras	pasukimo kryptis	pasukimo kampas
0.0	dešinèn	90°
0.25	dešinèn	45°
0.5		0°
0.75	kairèn	45°
1.0	kairèn	90°

Lerp funkcija apibrëžiama taip:

$$Lerp(q_0, q_1, t) = \begin{bmatrix} q_{0_w} + (q_{1_w} - q_{0_w})t \\ (q_{0_x} + (q_{1_x} - q_{0_x})t)i \\ (q_{0_y} + (q_{1_y} - q_{0_y})t)j \\ (q_{0_z} + (q_{1_z} - q_{0_z})t)k \end{bmatrix} \quad (29)$$

Slerp funkcija suteikia vienodà judëjimo greitį nuo kvaterniono q_0 prie q_1 priklausomai nuo skaliaro, tačiau reikalauja bent trijų kvaternionų daugybų ir këlimo laipsniu. Tuo tarpu funkcija *Lerp*, nors ir nesuteikia vienodo judëjimo greičio, bet ji šį judëjimà aproksimuoja pakankamai tiksliai ketursraigčio kampinės pozicijos skaiçavimo tikslams. Taip pat lerp galima apskaiçiuoti tik su 4 skaliarinës daugybos operacijom, bei keleta atimçių ir sudëçių.

3.2. Sensoriai

Ketursraigčio kampinë pozicija skaiçiuojama pagal du sensorius: giroskopà ir akselerometrà.

Giroskopas. Giroskopas yra sensorius matuojantis ketursraigčio kampinį greitį (visomis trimis ašimis)

$$\omega_{gyro} = \begin{bmatrix} \omega_{gyro_x} \\ \omega_{gyro_y} \\ \omega_{gyro_z} \end{bmatrix} \quad (30)$$

Verta pastebëti, kad visi matavimai atliekami lokoliojoje koordinaçių sistemoje, o mažos paklaidos dël netikslumų montuojant giroskopà yra ignoruojamos.

Akselerometras. Akselerometras yra sensorius matuojantis pagreiçius visomis trimis ašimis lokoliojoje koordinaçių sistemoje. Verta pastebëti, kad žemës gravitacija sensorių veikia lygiai taip pat, kaip akseleracija priešinga kryptimi. Kadan-

gi žemės graviacijos kryptis yra visada nukreipta žemyn, todėl jos kryptis gali būti naudojama kampinės padėties apskaičiavimui.

$$a_{acc} = \begin{bmatrix} a_{acc_x} + g_x \\ a_{acc_y} + g_y \\ a_{acc_z} + g_z \end{bmatrix} \quad (31)$$

g_x, g_y, g_z , yra gravitacijos vektoriaus komponentės lokaloje koordinačių sistemoje.

3.3. Kampinės padėties skaičiavimas pagal giroskopą

Kaip jau aptarta ankščiau, giroskopas matuoja kampinį greitį, todėl norėdami rasti poziciją turime jį integruoti.

$$x_t = \int_0^t \omega(\tau) d\tau \quad (32)$$

tačiau realus giroskopas atlieka matavimus diskretiškai, todėl daroma prielaida, kad išmatuotas kampinis greitis ω_t yra konstanta viso laiko momento t metu, tada:

$$x_t = x_{t-1} + \omega_t * \Delta t \quad (33)$$

Kur x_t yra kampinė pozicija laiko momentu t , o ω_t yra kampinis greitis laiko momentu t .

Tą patį išreiškus kvaternionais:

$$q_{pgt} = q_{p_{t-1}} * q_{gt} \quad (34)$$

Čia q_{pgt} kvaternionas laiko momentu t , reprezentuojantis pasukimą nuo globalios iki lokalsios koordinačių sistemos ir paskaičiuotas pagal giroskopą, o q_{gt} kvaternionas reprezentuojantis pasisukimą laiko momentu t . $q_{p_{t-1}}$ yra galutinės kampinės pozicijos kvaternionas pasukantis globaliąją koordinačių sistemą iki lokalsios laiko momentu $t - 1$ (žr.: 3.5).

Kvaternionas q_{gt} yra sudaromas pagal principus apibūdintus skyrelyje 3.1,

kuomet sukimo kampas ϕ yra giroskopo išmatuoto vektoriaus ilgis:

$$q_{g_t} = \begin{bmatrix} \cos\left(\frac{\sqrt{\omega_x^2 + \omega_y^2 + \omega_z^2}}{2}\right) \\ \frac{\omega_x}{\sqrt{\omega_x^2 + \omega_y^2 + \omega_z^2}} \sin\left(\frac{\sqrt{\omega_x^2 + \omega_y^2 + \omega_z^2}}{2}\right) \\ \frac{\omega_y}{\sqrt{\omega_x^2 + \omega_y^2 + \omega_z^2}} \sin\left(\frac{\sqrt{\omega_x^2 + \omega_y^2 + \omega_z^2}}{2}\right) \\ \frac{\omega_z}{\sqrt{\omega_x^2 + \omega_y^2 + \omega_z^2}} \sin\left(\frac{\sqrt{\omega_x^2 + \omega_y^2 + \omega_z^2}}{2}\right) \end{bmatrix} \quad (35)$$

Čia $\omega_x, \omega_y, \omega_z$, yra giroskopo atlikti matavimai. 35 Sudarytas kvaternionas q_{g_t} jau yra normalizuotas.

3.4. Kampinės padėties skaičiavimas pagal akselerometrą

Akselerometro matavimai, nesant išorinių akseleracijų, leidžia tiesiogiai apskaičiuoti kampinę sensoriaus poziciją, tačiau šis matavimas turi palyginus dideles paklaidas ir triukšmo lygį. Taip pat esant išorinėms akseleracijoms paklaidų pasiskirstymas gali būti nekoreliuotas su gravitacijos vektoriaus reikšme. Todėl filtravimas yra būtinas ir jis įgyveninamas apjungiant kampines pozicijas apskaičiuotas pagal girokopą ir pagal akselerometrą (žr.: 3.5)

Patogumo dėlei bus skaičiuojama ne absoliuti kampinė pozicija, o pasukimas, kuris pasuka dabartinę poziciją q_p iki akselerometro išmatuotos pozicijos. Tam tikslui pradžioje surandame vektorių lokoliojoje koordinačių sistemoje, kuris globalioje koordinačių sistemoje yra nukreiptas tiesiai aukštyn.

$$q_{up_g} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ k \end{bmatrix} \quad (36)$$

pasukame:

$$q_{up_l} = q_p * q_{up_g} * q_p^{-1} \quad (37)$$

Išskiriame vektorių į viršų

$$v_{up} = \begin{bmatrix} q_{upl_x} \\ q_{upl_y} \\ q_{upl_z} \end{bmatrix} \quad (38)$$

Toliau sudarome kvaternioną pavaizduojantį pasukimą nuo dabartinės apskaičiuotos kampinės pozicijos iki akcelerometro išmatuotos pozicijos.

$$q_{acc_{diff}} = \begin{bmatrix} \sqrt{||v_{up}||^2 ||a_{up}||^2 (a_{up} \cdot v_{up})} \\ a_{up_y} v_{up_z} - a_{up_z} v_{up_y} \\ a_{up_z} v_{up_x} - a_{up_x} v_{up_z} \\ a_{up_x} v_{up_y} - a_{up_y} v_{up_x} \end{bmatrix} \quad (39)$$

Čia \cdot reiškia skaliarinę sandaugą, a_{up} yra akcelerometro išmatuotas vektorius. O $||v||$ žymi vektoriaus v ilgį.

3.5. Galutinis kampinės padėties radimas

Pagal giroskopą apskaičiuota pozicija yra palyginus tiksli, bet ilgainiui gali krypti nuo tikrosios pozicijos, t.y. dreifuoti. Tuo tarpu pagal akcelerometrą paskaičiuota pozicija yra palyginus triukšminga, bet negali dreifuoti, todėl skaičiuodami galutinę poziciją imame rezultatus, kuriuos duoda giroskopas ir pritaikome papildomąjį (ang.: *complementary*) filtrą.

$$q_{pgt} = q_{p_{t-1}} * q_{g_t} \quad (40)$$

Tuomet sudaromas kvaternionas reprezentuojantis 0° pasukimą.

$$q_0 = \begin{bmatrix} 1.0 \\ 0.0 \\ 0.0 \\ 0.0 \end{bmatrix} \quad (41)$$

Toliau apskaičiuojamas svorio koeficientas akcelerometro išmatuotai kampinei pozicijai filtruoti taip, kad šis koeficientas esant $||a|| = 1.0g$ būtų lygus sistemos parametrais C_{max} ir atitinkamai mažėtų iki parametro C_{cutoff} .

$$W_{acc} = C_{max} - |||a|| - 1.0| \left(\frac{C_{max}}{C_{cutoff} - 1} \right) \quad (42)$$

Šis svorio koeficientas W_{acc} yra apribojimas žemutine riba 0.

$$W = \begin{cases} W_{acc}, & \text{Jeigu } W_{acc} \geq 0 \\ 0, & \text{Kitu atveju} \end{cases} \quad (43)$$

Toliau yra apskaičiuojamas kvaternionas, kuriuo pasukus q_{pgt} bus atliktas kompensavimas pagal akselerometro duomenis

$$q_{diff} = Lerp(q_0, q_{acc_{diff}}, W) \quad (44)$$

Ir pasukame q_{pgt}

$$q_p = q_{pgt} * q_{diff} \quad (45)$$

q_p yra galutinis pozicijos kvaternionas, naudojamas ketursraigčio skrydžio valdymo algoritmuose.

4. Kampinės padėties valdymo algoritmas

Ketursraigčio kampinės padėties valdymui buvo parinktas PID algoritmas.

4.1. PID valdymo algoritmas

PID valymo algoritmas yra uždaro ciklo valdymo algoritmas. PID operuoja skaičiuodamas skirtumą tarp matuojamo kintamojo ir pateikto tikslo (ang.: *set-point*). Toliau PID algoritmas reguliuoja proceso kintamąjį „stengdamasis“ sumažinti skirtumą tarp matuojamo kintamojo ir pateikto tikslo. PID yra parametrizuojamas trimis konstantomis:

- P – proporcinis koeficientas;
- I – integralinis koeficientas;
- D – diferencialinis koeficientas.

Bendroji valdymo lygtis PID algoritmui:

$$PID(t) = K_p e(t) + K_i \int_0^t e(t) dt + K_d \frac{d}{dt} e(t) \quad (46)$$

Čia $PID(t)$ yra proceso kintamasis, $e(t)$ – matuojamo ir tikslo kintamųjų skirtumas. K_p , K_i , K_d – PID algoritmo parametrai.

Verta pastebėti, kad skaitmeninio procesoriaus aplinkoje PID algoritmas dirba diskretiškai, todėl integravimas ir diferencijavimas aproksimuojami tinkamai suma bei skirtumu

$$\int_0^t x_\tau d\tau \approx x_t + x_{t-1} + x_{t-2} + \dots + x_1 + x_0 \quad (47)$$

$$\frac{d}{dt} x_t \approx x_t - x_{t-1} \quad (48)$$

4.2. PID pritaikymas ketursraigčio valdymui

PID algoritmai pasiekia geriausių rezultatų dirbdami tiesinėse sistemose, todėl pirma bus išvedama kampinės pozicijos priklausomybė nuo proceso kintamojo, arba konkrečiu atveju – propelerių greičio.

$$F = \omega^2 * C \quad (49)$$

Šiuo atveju ω yra tai, kas yra valdoma PID algoritmo. Tam kad sistema būtų tiesiška – ištraukiama šaknis iš PID išeities. Iš to yra formuojamas signalas greičio valdikliams, kurių atsakas yra tiesiškas.

$$u = \sqrt{PID(t)} \quad (50)$$

tuomet

$$F = \sqrt{u}^2 * C \quad (51)$$

Kadangi variklio valdiklių valdymo signalai niekada nebūna neigiami:

$$F = u * C \quad (52)$$

Iš antrojo niutono desnio:

$$a = \frac{F}{m} \quad (53)$$

Taigi:

$$a = \frac{u * C}{m} \quad (54)$$

Kadangi masė m yra nekintantis dydis skrydžio metu, konstantą C pervadiname į:

$$C \leftarrow \frac{C}{m} \quad (55)$$

Įvedame laiko priklausomybę:

$$\omega = at \quad (56)$$

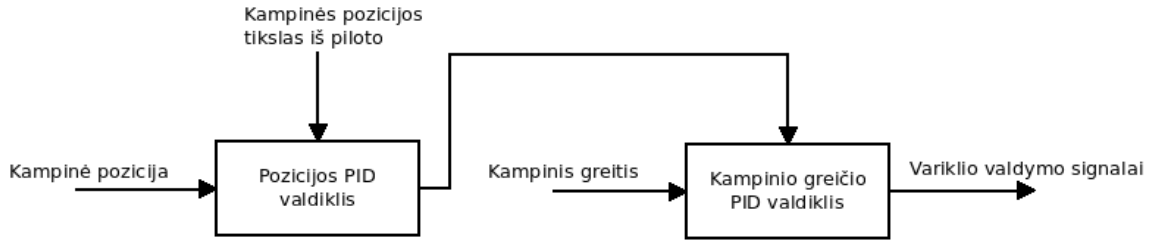
$$\omega = utC \quad (57)$$

Šioje vietoje turimas tiesinis modelis, kuriam taikomas PID algoritmas ketursraigčio kampinio greičio valdymui. Tačiau tikslas yra valdyti kampinę poziciją, o ne greitį, todėl reikia išsivesti kampinės pozicijos priklausomybę nuo laiko:

$$x = \omega * t \quad (58)$$

Pozicija nuo greičio priklauso tiesiškai, todėl čia naudojamas antrasis PID algoritmas nuosekliai jungiamas su kampinį greitį valdančiu PID algoritmu (žr.: pav.

8).



8 pav. Nuoseklus PID jungimas greičiui ir kampiniai pozicijai valdyti. PID valdikliai vaizduojami su rodykle iš kairės reiškiančią matuojamą proceso kintamąjį, iš viršaus tikslo kintamąjį, dešinėje – PID išeitis.

PID paklaidos skaičiavimas. Ketursraigčio kampinės padėties valdymui naudojami du valdikliai apibūdinti ankstesniame skyrelyje. Vienas valdo sukimąsi apie lokalią X ašį, valdydamas apie Y ašį išdėliotus variklius, kitas valdo sukimąsi apie Y ašį valdydamas apie X ašį išdėliotus variklius. Matuojamas proceso kintamasis yra kampas nuo horizontalios padėties. O PID paklaida (ang.: *error*) yra skirtumas tarp matuojamo kintamojo (kampas nuo horizontalios pozicijos) ir piloto (arba autopiloto) nustatytos reikšmės.

PID paklaidą skaičiuojama pagal q_p kvaternioną (žr.: 3.5)

Iš pradžių apskaičiuojamas aukštis, kuriame yra atitinkamas X arba Y variklis. Tam sudaromas kvaternionas žymintis variklio poziciją lokaliaje koordinatinių sistemoje.

$$q_{eng_x} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \quad (59)$$

Šis kvaternionas pasukamas pozicijos kvaternionu:

$$q_{eng_{x_h}} = q_p * q_{eng_x} * q_p^{-1} \quad (60)$$

Ir jo pozicija projektuojama į Z ašį.

$$h_x = q_{eng_{x_{hz}}} \quad (61)$$

Tuomet apskaičiuojamas kampas:

$$\alpha = \arcsin(h_x) \quad (62)$$

α jau gali būti naudojamas kaip matuojamas proceso kintamasis pozicijos PID valdikliui, taip pat $\frac{d}{dt}\alpha$ naudojamas greičio PID valdikliui, tik diskretizuota versija:

$$\frac{d}{dt}\alpha \hat{=} \alpha_t - \alpha_{t-1} \quad (63)$$

Verta pastebėti, kad šitoks skaičiavimas teisingai paklaidas skaičiuoja tik iki tol, kol jos neviršija 90° , bet ketursraigis negali išsilaikyti ore palinkęs tokiais stipriais kampais, todėl atvejai su tokiais kampais nėra nagrinėjami.

PID valdiklio išeitis yra naudojama modifikuoti dabartiniam galios tikslui (ang.: *setpoint*), jeigu tai yra PID valdiklis valdantis pasisukimą pagal X ašį, tuomet antram varikliui galia yra didinama tiek, kokia yra PID išeitis, o trečiam atitinkamai mažinama. Analogiškai veikia ir Y ašis.

4.3. Pasisukimo pagal lokaliąją Z ašį valdymas

Ketursraigčių pasisukimas apie Z ašį remiasi tuo, kad iš keturių propelerių du sukasi pagal ir du prieš laikrodžio rodyklę. Oro pasipriešinimas sukelia jėgą verčiančią ketursraigį suktis priešingą kryptimi negu propeleriai. Kol bendras jėgų didumas propeleriams besisukantiems pagal laikrodžio rodyklę yra panašus į jėgų didumą propeleriams besisukantiems prieš laikrodžio rodyklę, tol šios jėgos viena kitą kompensuoja ir ketursraigis nesisuka apie savo lokaliąją Z ašį. Pakeitus šį balansą galima priversti ketursraigį suktis viena ar kita kryptimi, tačiau šia ašimi ketursraigčiai yra kur kas mažiau manevringesni negu X ar Y todėl valdymo algoritmas yra kur kas paprastesnis.

Kaip ir PID kontrolerio atveju, čia skaičiuojama pasisukimo pagal Z ašies paklaida.

Sugeneruojamas kvaternionas atitinkantis X variklio poziciją lokaliaje koordinatinių sistemoje.

$$q_{eng_x} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \quad (64)$$

Pasukamas pagal esamą poziciją

$$q_{eng_{x_l}} = q_p * q_{eng_x} * q_p^{-1} \quad (65)$$

Čia imamas X, Y vektorius ir normalizuojamas.

$$x_{nenorm} = \begin{bmatrix} q_{eng_{x_l x}} \\ q_{eng_{x_l y}} \end{bmatrix} \quad (66)$$

$$x = \frac{x_{nenorm}}{||x_{nenorm}||} \quad (67)$$

Ir skaičiuojamas kampas

$$\alpha_z = \begin{cases} arccos(x_x), & \text{jeigu } x_y \geq 0 \\ -arccos(x_x), & \text{kitu atveju} \end{cases} \quad (68)$$

Valdymo signalai yra gaunami:

$$u_z = \alpha_z * K_{p_z} \quad (69)$$

Šis valdymo signalas yra pridedamas prie variklių galios besisukančių pagal laikrodžio rodyklę, ir atimamas iš variklių, besisukančių prieš laikrodžio rodyklę.

5. Skrydžio autonomiškumas

Skrydžio autonomiškumas pasiekiamas atviro ciklo valdikliu, siunčiant valdymo komandas žemesniame valdymo lygmenyje esantiems PID valdikliams.

5.1. Atviro-ciklo valdymas

Atviro ciklo valdiklis tai yra valdiklis, kuris skaičiuoja valdymo parametrus tik pagal dabartinę būseną ir neatlieka valdomo objekto matavimų. Šio tipo valdiklis ketursraigčiui buvo parinktas dėl fizinių sensorių nebuvimo, kurie galėtų matuoti ketursraigčio poziciją.

5.2. Kampinės pozicijos tikslų lentelė

Ketursraigčio valdymas atliekamas valdant bendrąją galią bei kampą pagal X , Y , Z ašis. Šiuos duomenis priima kampinės pozicijos valdikliai, kaip tikslo kintamuosius, ir nustato atitinkamas variklių galias.

Ketursraigčio valdymui sudaroma šių tikslų lentelė ir jie yra siunčiami ketursraigčiui periodiškai.

galia	kampas X	kampas Y	kampas Z	galiojimo laikas
20 %	0°	0°	0°	2s
40 %	0°	5°	0°	1s
40 %	0°	-5°	0°	1s
40 %	-5°	0°	0°	1s
40 %	5°	0°	0°	1s
20 %	0°	0°	0°	2s

1 lentelė. Supaprastintas tikslų lentelės pavyzdys

Nepriklausomai nuo galiojimo laiko, tikslai yra siunčiami 10Hz dažniu, jeigu galiojimo laikas yra ilgesnis – siunčiamas tas pats tikslas, kol galiojimo laikas pasibaigia. Taip ketursraigtis turi galimybę atpažinti ar nenutrūko ryšys su pilotu ar pilotuojančia programa.

5.3. Atviro-ciklo valdymo trūkumai

Vertinant šį algoritmą pagal pozicijos paklaidas (lyginant du skrydžius vieną su kitu), yra labai didelės paklaidos dėl šio algoritmo nesugebėjimo prisitaikyti prie

nenumatytų išorinių ir vidinių veiksnių (pvz.: Vėjo). Taip pat PID valdikliai nėra tikslūs, o tai dar labiau padidina paklaidas.

TODO__DIAGRAM__OF__ANGLE__ERROR

TODO__DISTURBANCE__EXAMPLE

6. Programinė įranga

6.1. Bendroji architektūra

Ketursraigtis susideda iš trijų esminių fizinių dalių:

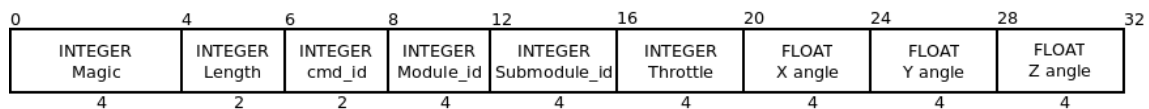
- kompiuterio – formuojančio ir siunčiančio valdymo komandas;
- retransmitoriaus – persiunčiančio valdymo komandas ketursraigčiui;
- pagrindinės valdymo elektronikos – valdančios ketursraigčio skrydį.

Kiekviena iš šių dalių turi savo programinę įrangą.

6.2. Kompiuteriui skirtas klientas

Kompiuteriui skirto kliento buvo paruoštos dvi versijos, viena skirta rankiniam ketursraigčio valdymui vairalazde, kita įgyvendinanti aptartą atviro ciklo valdymo algoritmą (žr.: 5).

Abiejais atvejais tai yra „python“ kalba parašyta programa, 10Hz dažniu nuskaitydama tikslo vertes (ar iš lentelės ar iš vairalazdės) ir suformuodama UDP paketą (žr.: pav. 9)



9 pav. Valdymo paketas siunčiamas valdymo elektronikai. Skaičius apačioje rodo lauko dydį baitais, viršuje lauko poslinkį nuo paketo pradžios

6.3. Retransmitorius

Retransmitorius kartu yra ir fizinis įrenginys. Tai yra Raspberry Pi kompiuteris su Linux operacine sistema. Šis kompiuteris sukonfigūruotas prisijungti prie 3G/4G tinklų ir klausosi UDP paketų siunčiamų jo adresu. Visą gautą informaciją jis tiesiog persiunčia per USB sąsają valdymo elektronikai.

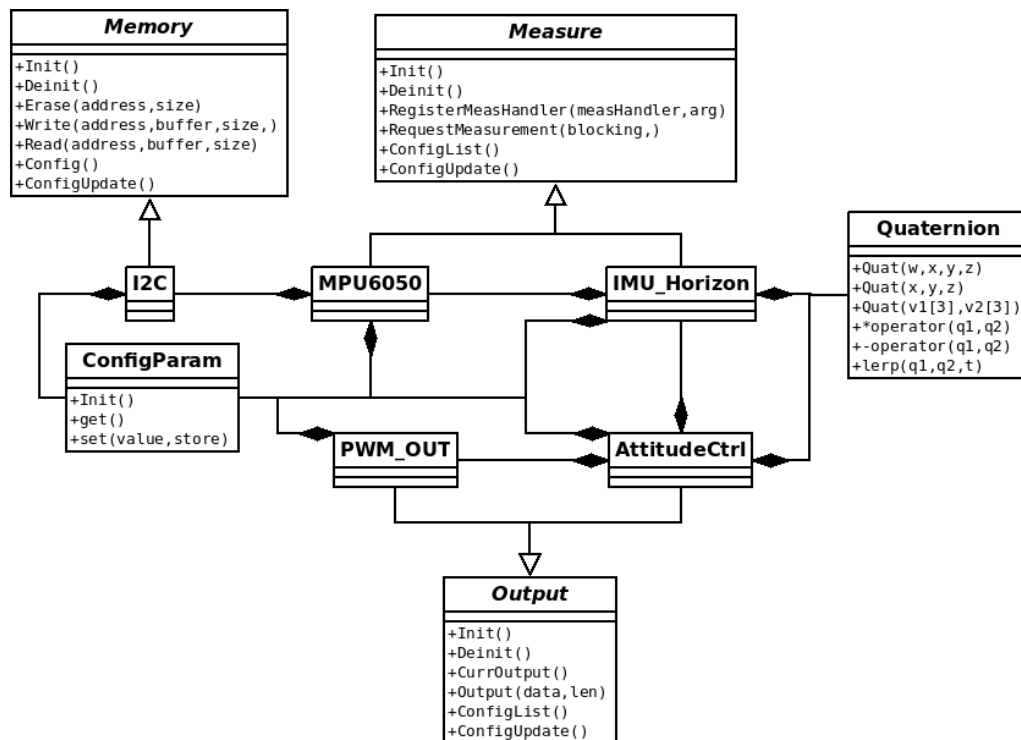
6.4. Ketursraigčio pagrindinis valdiklis

Pagrindinės valdymo elektronika sudaro GY-86 sensorių plokštė, su girokopu ir akselerometru, keturi vienetai greičio valdiklių, ir STM32F401 procesoriumi. Iš

šitų elementų vienintelis STM procesorius reikalauja programavimo, tad jam buvo parašyta programinė įranga įgyvendinanti visus sukurtus algoritmus ir vykdanči skrydžio kontrolę. STM programinė įranga yra išskaidyta į modulius.

- I2C – modulis skirtas palaikyti komunikacijai su sensoriais.
- MPU6050 – sensorių tvarkyklės modulis.
- Quaternion – kvaternionų algebrą įgyveninantis modulis.
- IMU_Horizon – pozicijos kvaternioną skaičiuojantis modulis.
- AttitudeCtrl – kampinės pozicijos valdymo modulis.
- PWM_OUT – modulis generuojantis signalą variklių valdikliams.
- Connection – USB komandų palaikymo modulis.
- ConfigParam – abstraktus konfigūracijos parametru modulis.

Verta pastebėti, kad kodas rašytas C kalba, bet šiek tiek mėgdžiojami kai kurie OOP principai. Todėl toliau pateikiama klasių diagrama atitinkanti programinės įrangos struktūrą, jei ji būtų rašoma objektine programavimo kalba.



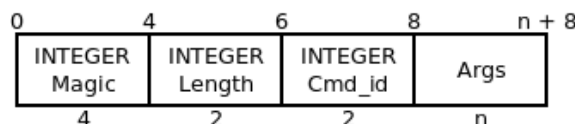
10 pav. Vidinė pagrindinio valdiklio programinė struktūra.

„Connection“ modulis. „Connection“ yra modulis, kuris jungiasi su visomis klasėmis, dėl to atskirai nebuvo pavaizduotas diagramoje. Jis turi tik vieną „Init“ metodą. „Connection“ modulis yra atsakingas už informacijos paėmimą iš USB sąsajos ir jos perdavimą atskiriems moduliams pagal jų ID. Komunikacija per USB sąsają palaikoma pagal serverio-kliento principus. Kompiuteris būdamas klientu siunčia užklausas ir sulaukia atsakymo kiekvienai užklausiai.

„Connection“ modulis palaiko šias komandas:

1. *Reset* – perkrauna įrenginį;
2. *Enable device* – aktyvuoja modulį;
3. *Disable device* – deaktivuoja modulį;
4. *SetConfig* – nustato parametro reikšmę;
5. *ListActive* – pateikia aktyvių modulių sąrašą;
6. *ListSupported* – pateikia palaikomų modulių sąrašą;
7. *ListParam* – pateikia konkretaus aktyvaus modulio parametrų sąrašą;
8. *Output* – iškviečia modulio, įgyvendinančio „Output“ sąsają metodą „Output“;
9. *CurrOutput* – iškviečia modulio, įgyvendinančio „Output“ sąsają metodą „CurrOutput“;
10. *ListChecker* – pateikia klaidų žurnalą;
11. *RequestMeasurement* – iškviečia modulio, įgyvendinančio „Measure“ sąsają metodą „RequestMeasurement“;

Visos komandos paklusta baziniam protokolui, kuriame nurodomas komandos ilgis baitais, komandos identifikatorius be argumentai skirti konkrečiai komandai (žr.: pav. 11).



11 pav. Bazinė „Connection“ modulio užklausos paketo struktūra. Skaičius viršuje rodo poslinkį nuo paketo pradžios, apačioje – lauko ilgį.

Čia laukas „Magic“ yra paketo pradžios identifikatorius ir visada įgauna tokią pačią reikšmę, kuri yra lygi A5B7C957 šešioliktainėje sistemoje. Laukas „Length“ nurodo viso paketo ilgį (įskaitant ir „Magic“ bei „Length“ laukus). Kiekviena komanda turi savo ID, kuris pateikiamas lauke „Cmd_id“, o lauke „Args“ sudedama jau konkrečių kiekvienos komandos reikalaujama informacija.

Išvados

Šiame darbe buvo sukurta fizikiniu modeliu besiremianti ketursraigčio skrydžio valdymo sistema, bei algoritmai kuriais ši sistema vadovaujasi.

Darbe buvo išvestas supaprastintas fizikinis ketursraigčio skrydžio modelis, sukurtas kvaternionais grystas algoritmas leidžiantis rasti ketursraigčio kampinę poziciją pasinaudojant nebrangiais MEMS giroskopo bei akselerometro sensoriais. Toliau pritaikomi uždaro-ciklo PID algoritmai ketursraigčio kampinės pozicijos valdymui, taip pat pateikiamas atviro-ciklo valdymo algoritmas ketursraigčio pozicijos valdymui. Visa tai buvo suprogramuota ir išbandyta realaus ketursraigčio skrydžio metu.

Rezultatai. Šiame darbe buvo sukurti bei išbandyti trys esminiai algoritmai: ketursraigčio kampinės pozicijos skaičiavimo, ketursraigčio kampinės pozicijos valdymo bei ketursraigčio pozicijos valdymo algoritmas. Kampinės pozicijos algoritmas parodė gerus rezultatus, net ir sunkiomis darbo sąlygomis, kuomet sensoriai dirbo stiprios vibracijos sąlygomis. Sensoriui dirbant normaliomis sąlygomis (vibracijų dažnis žemesnis nei ketvirtadalis matavimų dažnio, ir akseleracijos bei sukimo greičiai neviršija sensoriaus matavimo ribų), buvo pasiekta mažesnė nei 1° paklaida, taip pat nebuvo aptikta jokių užrakinimo (ang.: *gimbal lock*) pozicijų, būdingų panašioms algoritmams besiremiantiems oilerio kampais. Tuo tarpu PID algoritmas parodė šiek tiek prastesnius rezultatus. Esant mažoms kampinės pozicijos paklaidoms buvo pastebimos lengvas ketursraigčio kampinės padėties svyravimas. Taip pat nebuvo rastas tinkamas parametrų rinkinys, kuris tenkintų reakcijos greitį bei sistemos stabilumo reikalavimus. Atviro ciklo pozicijos valdymas turėjo stiprias paklaidas, ir tiksliam autonominiam skrydžiui palaikyti – reikėtų ieškoti kito algoritmo.

Tolimesni darbai, patobulinimai. PID algoritmai nesuteikia optimalaus valdymo, ypač kai jų parametrai yra nustatomi rankiniu būdu ir randamas tik „pakankamai-geras“ parametrų derinys. Tuo tarpu tiesinis-kvadratinis reguliatorius (ang.: *linear-quadratic regulator* arba *LQG*), nors ir lieka šiek tiek rankinio algoritmo reguliavimo, teoriškai galėtų pasiekti geresnius rezultatus. Taip pat specialūs fizika paremti modeliai valdymui galėtų būti sukurti agresyviai ketursraigčio valdymui.

Pozicijos valdymas galėtų būti išreikštas uždaro-ciklo valdymo algoritmu, įtaisius tikslius GPS įtaisus, sukūrus modelį bei algoritmus pozicijos valdymui. Tokie patobulinimai pozicijos valdymo tikslumą galėtų pagerinti dešimtis ar net šimtus

kartų. Taip pat oro slėgio sensoriai galėtų būti naudojami automatiniam skrydžio aukščio valdymui. Tam irgi turėtų būti sukurtas modelis, bei valdymo algoritmai.

Literatūros sąrašas

- [ABCO15] Fatemeh Abyarjoo, Armando Barreto, Jonathan Cofino, and Francisco R Ortega. Implementing a sensor fusion algorithm for 3d orientation detection with inertial/magnetic sensors. In *Innovations and Advances in Computing, Informatics, Systems Sciences, Networking and Engineering*. Springer, 2015.
- [ECM⁺08] Mark Euston, Paul Coote, Robert Mahony, Jonghyuk Kim, and Tarek Hamel. A complementary filter for attitude estimation of a fixed-wing uav. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 340–345. IEEE, 2008.
- [Gib10] Andrew Gibiansky. Quadcopter dynamics, simulation, and control. 2010.
- [KJ04] N Shantha Kumar and T Jann. Estimation of attitudes from a low-cost miniaturized inertial platform using kalman filter-based sensor fusion algorithm. *Sadhana*, 29(2):217–235, 2004.
- [MS11] Øyvind Magnussen and Kjell Eivind Skjønhaug. Modeling, design and experimental study for a quadcopter system construction. 2011.