A Cserépfedés Automatizált Építésének Problémaelemzése

BME Építészmérnöki Kar, Építéstechnológia és Építésmenedzsment Tanszék

2022. TDK konferencia

Szerző: Barna Bettina

Konzulens: Dr. Vidovszky István János

Tartalomjegyzék

Absztrakt 2

1 Bevezetés 3

2 Módszertan 7

2.1 Modellkörnyezet 7

2.2 Raktár elrendezési módok 8

2.3 Rakásképek 9

2.4 A program felépítése 10

3 Eredmények 13

4 Összegzés 14

5 Kitekintés 15

6 Források 18

6.1 Szöveges tartalom 18

6.2 Ábrák 18

7 Kódrészlet 19

Absztrakt

Az elmúlt évtizedek során a robotok a legtöbb iparág kulcsfontosságú szereplőivé váltak, de építéstechnológiai alkalmazásuk máig kísérleti jellegű. Belátható azonban, hogy e területen is létezik jónéhány olyan munkafolyamat, melyek automatizálása nagymértékben növelhetné a hatékonyságot illetve a biztonságot. Minden magasban végzett tevékenység kiemelt baleseti kockázattal jár, amit a körülményesnek tartott munkavédelmi előírások gyakori be nem tartása csak tovább fokoz. További gond az egyre súlyosbodó szakemberhiány, melynek velejárójaként néha képzetlen személyekkel is végeztetnek ilyen jellegű munkát.

A kutatás témája egy tetőhéjalás automatizálási kísérletsorozat. Hasonló feladatot feldolgozó szakirodalom hiányában jelen dolgozat a témát a legáltalánosabb peremfeltételek mentén vizsgálja. A feladat egy makett méretű egyszerű nyeregtető lefedése illeszkedő léptékű síkcserepekkel. Mivel a kiegészítő elemek (pl. gerinc-és szegélycserepek) automatizált elhelyezése csak az alapelemekre vonatkozó átfogó elemzések után, a fejlődési folyamat egy további szakaszában időszerű, ezért ezek itt nem képezik vizsgálat tárgyát.

Az automatizáláshoz DobotMagician sínes asztali robotkart, valamint általa támogatott programozási környezet alkalmazunk. A kutatás célja minél több, a valós építés során bizonyosan felmerülő, laboratóriumi körülmények közt is elfogadhatóan modellezhető kérdés vizsgálata, mint például az elemek tárolási és megfogási módja, vagy a ferde tetősíkhoz történő zökkenőmentes illesztés.

1 Bevezetés

Az ipari robotok kutatása az 1950-es, 60-as években indult be jelentős mértékben, és a számítástechnika fejlődése által, a 80-as, 90-es évekre jutott el a termelésben való kiterjedt alkalmazás szintjére. Első megjelenésük a különösen repetitív és jól sorba rendezhető technológiai fázisokból felépülő, nagyszámú típusterméket előállító, környezeti hatásoktól megfelelően függetleníthető iparágakhoz köthető. Ezek gyártási folyamatai mára olyan mértékben automatizálttá váltak, hogy akár teljesen önállóan, vagy csak alig néhány, a termelés mennyiségéhez képest elenyésző számú személy bevonásával képesek működni. Az automatizálás célja a minél magasabb minőség és hatékonyság elérése, valamint az emberi munkaerő helyettesítése monoton vagy veszélyes feladatokban.

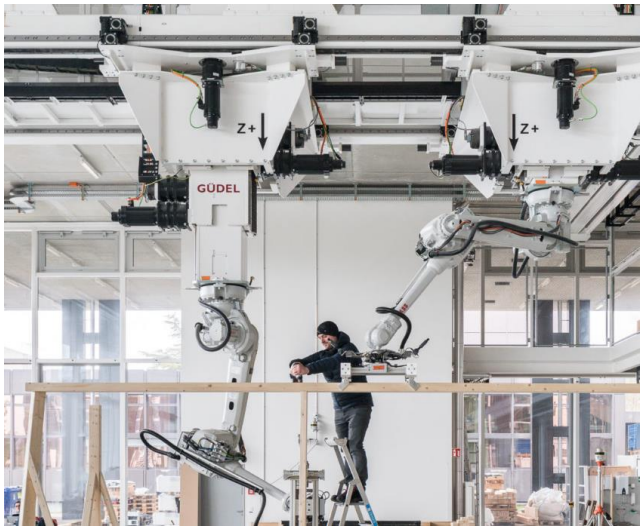
Jelenleg a legnagyobb arányban robotokat használó iparág az autógyártás, de számuk az elektronikai-, fém-és gépiparban is rendkívül magas. Leggyakrabban végzett feladataik az anyagmozgatás-és rendezés, a fémmegmunkálás és a különböző összeszerelési folyamatok, melyek kifejezetten alkalmasak a gépi interpretációra. [1] Azon iparágakban, ahol a munkafolyamatok kiszámíthatatlanabb környezetben zajlanak, vagy az előállítandó termékek gyakran egyediek, illetve nagyon nagy számú alkotóelemből állnak össze, az automatizálás folyamata több új nehézségbe ütközik – mindazonáltal valószínűsíthető, hogy bizonyos idő elteltével ezeken a területeken is bekövetkezhet egy hasonló irányú fejlődés, amely a környezettel intenzívebb kommunikációt folytató, kollaboratív jelleg, valamint a könnyebb és rugalmasabb, szélesebb felhasználói kört célzó programozhatóságra épül.

A robotok építőipari alkalmazásának gondolata már az 1980-as években felmerült, és elméleti alapjai is megfogalmazásra kerültek. Napjainkban, az építőiparhoz köthető szakágakban is egyre általánosabbá váló 3D alapú tervezés, az erre épülő, elterjedőben lévő BIM szemlélet (az épület, illetve az építés és üzemeltetés folyamatának digitális adatbázisként való kezelése), a kísérletező parametrikus épületformálás eszköztárának vagy a 3D nyomtatás lehetőségeinek

bővülése megfelelő alapot szolgáltathat egy újabb technológiai lépcső megközelítéséhez. [2] Jelenleg a leginkább jellemmező, az építőiparhoz köthető magas szinten automatizált folyamat az építési termékek (pl. falazó, burkoló és tetőfedő elemek, nyílászárók, előregyártott tartószerkezetek) előállítása és csomagolása, amit általában specicális eszközökkel felszerelt robotkarok végeznek. Ilyen, automata gépsorokon készült gyártmányok már szinte minden épületben találhatók, de beépítésüket a helyszínre szállítást követően kézzel, vagy kézi vezérlésű gépek segítségével végzik. Létező, de nem a hazai piacon kisebb mértékeben elterjedt az előregyártott épület, melynek elemei (egész falak, pillérek, nagyméretű panelek, stb. - jól vágható és rétegezhető anyagokból) üzemi körülmények között készülnek. Ebből többnyire emberi munkaerő állítja össze az épületet, de már akár robotok közreműködésével összeszerelt modellek is kaphatóak (pl. AUAR). [3]

1. *ábra: AUAR – Automated Architecture Ltd.*

Kisebb, készen is szállítható méretű házak esetében kihasználható az az előny, hogy az összeszerelés is üzemi körülmények között történhet, kiküszöbölve ezzel az időjárás és az építési helyszín esetleges akadályozó tényezőit. A munkához használható több fix robotkar, illetve speciális geometriájú szerkezetek esetében praktikus megoldás a sínen vagy függesztő kereten mozgó változat (pl. DFAB HOUSE, ETH Zürich). [4]



2. ábra: DEFAB HOUSE project, ETH Zürich

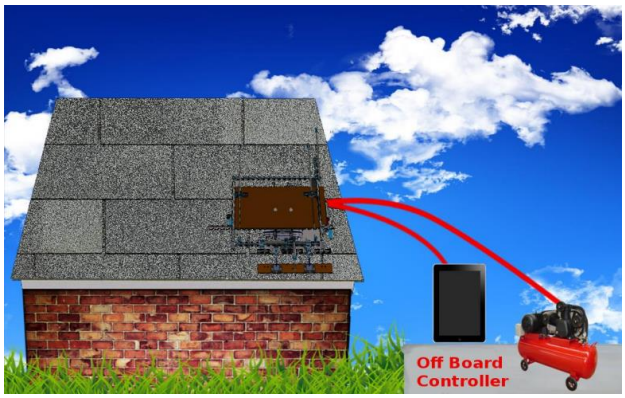
Innovatív jelleggel létrejöttek olyan automatizált megoldások, melyek egy adott építési folyamatot magas hatékonysággal és minőséggel, a helyszínen képesek megvalósítani (pl. Hadrian X, mely közvetlenül CAD rajzok alapján, speciális téglákból épít falat). [5]

3. ábra: HADRIAN X, FBR Ltd.

További érdekes, még inkább kísérleti állapotú elképzelés a drónokkal történő építés. [6] Nehézsége, hogy a repülő robotok nem képesek nagy terhek mozgatására, ezért valószínűleg a jövőben is maximum kiegészítő (pl. rögzítő, részletképző, takarító) funkció betöltésére használható.

4. *ábra: Tetőfedő drónok szegező pisztollyal, University of Michigan*

A tetőfedés témakörében a főként a szigetelőlemez fektetés automatizálása kapcsán jelentek meg új ötletek. A legalapvetőbb ilyen eszközök a kereskedelmi forgalomban is kapható, jelen kontextusban értelmezve „félautomata” forró levegős hegesztők, melyek a szigetelő rétegek egymáshoz simításával lehetővé teszik, hogy mindössze egy fő, segítség nélkül is végezhesse a munkát. [7] Magasabb szintű tudással rendelkezők, ám kísérleti állapotúak azok a robotok, melyek képesek egyszerű, felépítménymentes, nem túl meredek tetősíkon a feljuttatásukat követően önállóan haladva és navigálva szigetelő lemezeket szállítani és a megfelelő helyre rögzíteni, valamint a nem egész elemek méretét kiszámítani és azokat levágni. A robot munkavégzését itt is szükséges egy hozzáértő személynek felügyelnie, ezt azonban nagyrészt a földön tartózkodva, kényelmes és biztonságos távolságból teheti (pl. RiPUS). [8]



5. *ábra: Automated Shingling Robot, Team RiPUS*

A projekt fő motivációi között említésre kerül a magasban végzett munkák kiemelkedő baleseti kockázata, a szakemberhiány, valamint a nagy, egybefüggő tetőfelületek esetében jelentkező monoton jellegű munka. Valamennyi megállapítás helytálló a tető befejező rétege, azaz a héjazat készítésének munkafolyamatára nézve is, ezért a jövőben ennek automatizálása is egy hasonlóan érvényes kérdés lehet - jelen dolgozat ennek kapcsán igyekszik néhány alapvető, lényeges szempontot megvizsgálni.

2 Módszertan

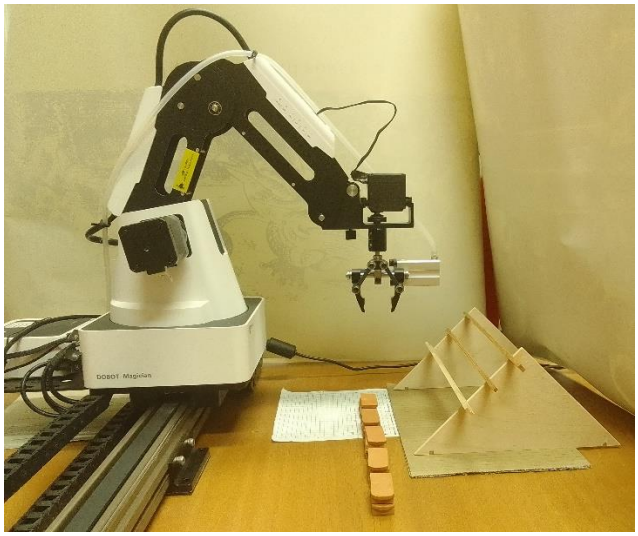
A kísérlet célja adott területű tetőrész automatizált lefedésének vizsgálata különböző raktár elrendezési módok és fedésképek mellett. Az elsődleges összehasonlítási alap az idő, de emellett egyéb paraméterek is érdekesek lehetnek, mint például pontosság, hibaszázalék, stb.

2.1 Modellkörnyezet

A mérések helyszíne egy laborasztalon készített elrendezés, mely építési területet modellez. Az elemek mozgatásához használt DobotMagician robotnak 4 a szabadságfokú kar, illetve a rögzített sínjén történő csúszási lehetőség biztosít az alapanyagraktár és a tető teljes területét lefedő mozgásteret. A csúsztató sín nem csak az elérhető terület kibővítése miatt hasznos, hanem ún. prizmatikus kapcsolatot is jelent, mely egy kitüntetett irányban a sok szabadságfokú karoknál stabilabb és pontosabb mozgását tesz lehetővé. Ennek az iránynak a tető ereszével párhuzamos irányt célszerű választani, mivel valamennyi munkafázis közül a felhelyezendő elemek egymás mellé való illesztése igényli a legnagyobb precizitást. A tetőfedő elemek égetett kerámia hódfarkú síkcserepek. Általános tetőfelület fedés vizsgálat okán jelenleg kizárólag egész alapelemeket használunk, de a fedésképkialakításakor a fél elemek helyeit következetesen üresen hagyjuk. A kar végén egy automatikusan vezérelhető ún. gripper end-effektor (befogó fej) található, ennek segítségével tudja a robot felemelni a cserepeket. A cserepek fogadószerkezete egy sematikusan megépített, 45°os hajlásszögű egyszerű nyeregtető-forma, mely tartalmazza a feladat elvégzéséhez szükséges cseréplécéket. Továbbra

is az általános tetőfelület elvét követve, a cserepek egyszerű beakasztással, kiegészítő rögzítés nélkül kerülnek a lécekre. (lásd 6. ábra) Megjegyzendő, hogy a modell helyszín a körülményekből adódóan nem tud néhány, valós szituációra minden bizonnyal befolyást gyakorló tényezőt figyelembe venni:

- időjárási viszonyok, melyek általában minden szabadtéri munkafolyamatot érintenek, az automatizáció fokától függetlenül
- a gripper cseréphez viszonyított mérete aránytalanul nagy, emiatt pontatlanabbak az illesztések
- a robot kialakítása (pl. mozgástér, end-effektor forgási tartomány) jelentősen korlátozza az építéshely lehetséges berendezési alternatíváit, speciálisan hasonló jellegű feladatra tervezett szerkezettel ezeknek szélesebb köre lenne tanulmányozható

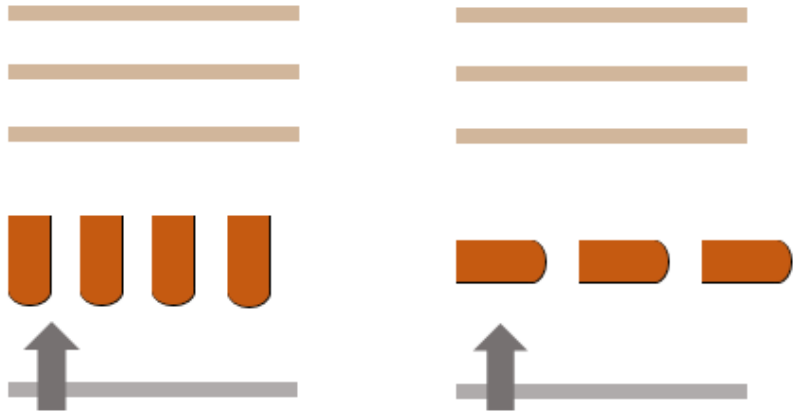


6. ábra: A modellezett építési helyszín, párhuzamos raktár-elrendezéssel

2.2 Raktár elrendezési módok

A raktár szituációk megválasztásakor szempont volt egy tetőre merőleges, illetve egy tetővel párhuzamos elemállást bemutató eset kialakítása (lásd 7. ábra). Ezen belül vizsgálat tárgyát képezi az egyes elemek rakatoszlopon belüli elhelyezési módja, mivel a nem megfelelő irányban álló cserepek elhelyezési helyzetbe történő forgatása többlet feladatot és időt jelent. Könnyen belátható, hogy a beakasztó fül miatti aszimetriából kifolyólag magasabb elemszám

esetén a rakatoszlop stabilitásának érdekében a cserepeket célszerű 180°kal elforgatva egymásra halmozni (lásd 8. ábra). E szabály betartása esetén mind a merőleges, mind a párhuzamos elrendezésben szükséges forgatás, előbbiben minden 2. elemnél 180° azonos irányban, utóbbinál elemenként 90° váltakozó irányokban. Mivel a gripperhez kapcsolt szervó motor alaphelyzetétől számítva mindkét irányban csak 150°-ot tudott fordulni, ezért jelen kísérletben a váltott elemhelyzetű raktár csak a párhuzamos kialakítás esetén valósulhatott meg.



7. ábra: Raktár elrendezési módok

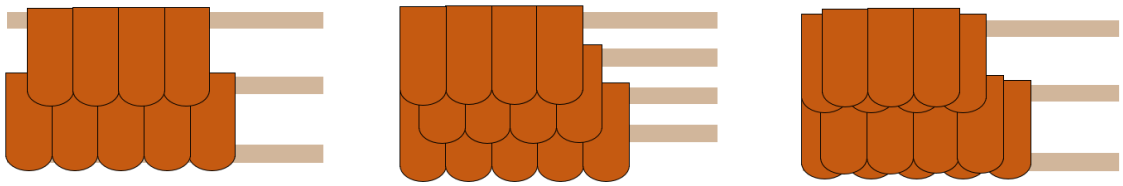
8. ábra: Tetőfedő elemek halmozási lehetőségei

A tető-sín viszony változtatására nem került sor, egyérészt a fentebb már említett pontossági ok miatt, másrészt, mert így a robotkar a tető hossza mentén már csak a karja által nyújtott mozgástartományban működhetett volna, ami elégtelennek és gyakorlati szempontból is értelmetlennek bizonyult.

2.3 Rakásképek

A kísérlet során az elemeket két, síkcserepek esetében közismert, közel azonos léctávolságot igénylő rakásképet alkalmaztunk: egyszeres fedést és lovag (más néven korona) fedést. Az

egyszeres fedés rendkívül anyagtakarékos, alacsony fokú vízzárása miatt főként egyszerűbb épületeken használatos. A lovagfedés hasonlóan ritka lécezés mellett, egymásra rakott dupla sorainak köszönhetően már jó vízzárást biztosít. Az egymásra fekvő cserepek miatt ugyan kevésbé ellenálló viharos szélben, de különleges esztétikájú mintázata miatt a történeti korok kedvelt megoldása volt, mely rekonstrukciós szempontból ma is igen jelentős. napjainkban legáltalánosabban elterjedt, legjobban teljesítő, ám legnagyobb anyagigényű kétszeres fedés a lécezés sűrítése által jön létre az egyszeres fedés analógiájára. Ezt jelenleg nem vizsgáltuk, viszont az építésmód azonos logikája miatt mindenképp megemlítendő. (lásd 9. ábra)



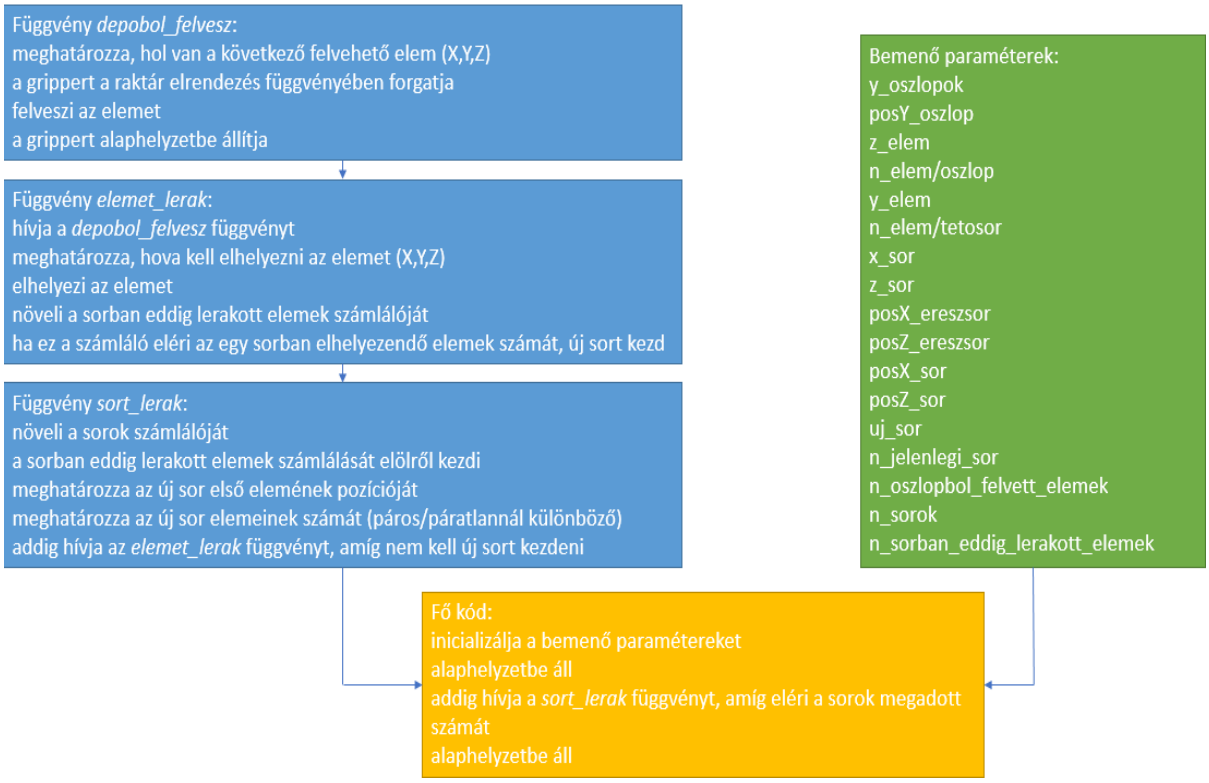
9. ábra: Síkcserepekre jellemző rakásképek: egyszeres, kétszeres, és lovagfedés

2.4 A program felépítése

A robotot vezérlő szoftver Python nyelvet támogató programozási környezetet, továbbá számos, gyárilag beépített függvényt biztosít, melyeken keresztül a robot alapvető mozgásfunkciói irányíthatók. Mivel a célfeladat viszonylag egyszerű és rövid, a hozzá tartozó kód is kellően specifikus, azaz mentes néhány olyan általánosító szerkesztési elvtől, amelyek egy nagyobb, komplexebb, biztonságosabban és rugalmasabban fejleszthető, illetve jobban ellenőrizhető tetőfedő algoritmus esetében már lényegesek lennének, jelen esetben viszont felesleges bonyolítást jelentettek volna. Már ezen a szinten is fontos volt azonban a munkafolyamat 3 alapvető részének külön logikai egységként való kezelése:

1. egy adott elem raktárból történő felvétele
2. egy adott, már felvett elem elhelyezése a tetőn
3. egy adott sor kirakása a tetőn

Ezeket az egységeket külön függvényekbe szerveztük, és egy, a kirakandó tetősorok darabszáma szerint lefutó ciklusban helyeztük el őket. Valamennyi bemenő paraméterüket külső változókban kezeltük, melyeket még a fő ciklus előtt egy kezdő értékkel hoztunk létre, és amennyiben szükséges volt, a megfelelő pontokon változtattunk.



10. ábra: Alapvető program struktúra

Az egy rakatoszlopban felhalmozott és az egy tetősorban lerakott elemek számát, valamint a kirakandó sorok számának paraméteres megadási lehetőségét szintén meghagytuk, ám az időmérés során természetesen azonosra állítottuk: 4 elem/rakatoszlop, 5 elem/tetősor (a kezdő sorra vonatkozóan), 2 tetősor (egy tetősornak az egy lécre akasztott cserepek sora tekintendő). A raktárban felhalmozott elemek esetleges többlete az életszerűséget követve irreleváns volt, a munkafolyamat a fent meghatározott 2 szabályosan kirakott tetősor befejezésével véget ért.

A kódban alkalmazott változók alapvetően három típusba sorolhatók, a jó azonosíthatóság kedvéért típusukra elnevezésük kezdő tagja is utal:

n_: egész szám, „darab” jellegű mennyiség (pl. oszlopban lévő elemek száma, a lerakott sorok számlálója)

x, y, z_: véges tizedestört, „távolság” jellegű mennyiség (pl. elemmagasság, oszlopok távolsága)

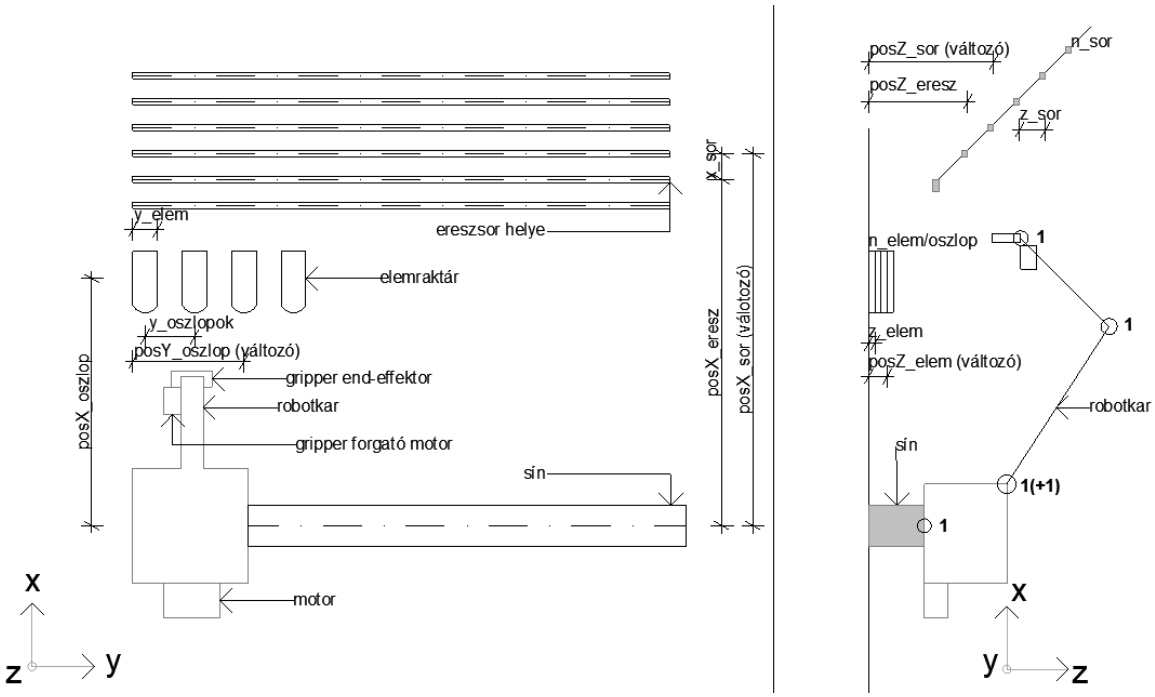
posX, posY, posZ_: véges tizedestört, „pozíció”, azaz origótól mért távolság jellegű mennyiség (pl. kezdősor koordinátái, aktuális sor koordinátái)

Az modelkörnyezetet leíró paramétereket a 11. ábra szemlélteti. Az elnevezésekben szereplő x, y, z betűk az ábrán látható tengelyek irányában értelmezett mennyiségeket jelölik. A kótasorokon szereplő elnevezések a használt változók nevei. Nem minden változó „valódi”, vannak olyanok, melyek a program futásának szempontjából igazából konstansok. Ezek az elrendezés méreteinek esetleges változási lehetőségét fenntartva kerültek paraméterként a kódba, mivel számos műveletnek részei, de értéküket mindig megőrzik (pl. az elem magassága, vagy az ereszsor helyzete). A valódi változók neve mögött az ábrán a „(változó)” megjegyzés szerepel (pl. egy adott oszlop pozíciója).

Vannak olyan valódi változók is, melyek számláló jellegűek, geometriai tartalmuk nem értelmezhető. Ezért nem jelennek meg az ábrán, de a program struktúrája szempontjából kulcsfontosságúak: ezek a sorban eddig lerakott elemeket, az oszlopból eddig felvett elemeket, valamint az eddig lerakott sorokat számolják. Mindenkori értékük rendre feltételként használt az új sor kezdésénél, az új oszlop kezdésénél és az elemek forgatásánál, illetve a feladat befejezésénél. Az új sor kezdéséhez egy logikai változó is szükséges. Ennek alapértéke mindaddig hamis, amíg a sorban lerakott elemek száma el nem éri az adott sorban lerakandó elemek számát. Ekkor az elem lerakásáért felelős függvényen belül igazzá válik, a program kilép az adott sort rakó ciklusból és az azt tartalmazó függvényből. Ha még nincs meg az összes

kirakandó sor, akkor új sort rakó függvény kezdődik, mely futása kezdetén beállítja az új sor paramétereit, és a logikai változót is visszakapcsolja.

A nyíllal ellátott feliratok tájékoztató jelleggel megjelölt fizikai elemei a munkakörnyezetnek, melyek meghatározó szempontként jelentkeztek a kód kialakításakor. A jobb oldali metszeten a robot csuklói melletti számok az adott kapcsolat szabadságfokait jelölik. A (+1) a kar y irányú, sínes helyettesítés miatt kihasználatlan mozgási lehetőségét szimbolizálja.



11. ábra: A modellkörnyezet fő elemei és paraméterei

3 Eredmények

Az adott raktár elrendezési módok mellett, az adott rakásképek elkészítéséhez szükséges idők, a kód futási ideje alapján mérve (perc/másodperc/századmásodperc):

	Merőleges elrendezés, forgatás nélkül		Párhuzamos elrendezés, forgatással	
Egyszeres fedés	02:23:87	Átlag: 02:23:92	02:59:83	Átlag: 02:59:87
	02:23:97		03:01:12	
	02:23:91		02:59:67	
Lovagfedés	04:58:51	Átlag: 04:58:58	06:20:12	Átlag: 06:20:00
	04:58:63		06:19:81	
	04:58:58		06:20:11	

Az századmásodperc különbségek oka, hogy a kód sok helyen tartalmaz mozgó alkatrészeket vezérlő elemeket. Ezeknek a kódrészletek beépített függvények, végrehajtásuk közben visszacsatolásokra várhatnak a robottól, melyek a mechanikus részek miatt ebben az időtartományban eltérőek lehetnek.

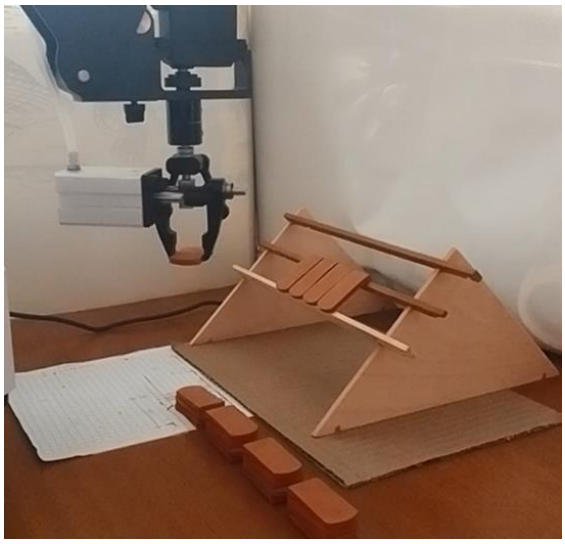
4 Összegzés

Végrehajtási idő tekintetében a legfeltűnőbb különbség természetesen a rakásképek között mutatkozik, hiszen azonos méretű felület elkészítése a lovagfedés esetében dupla cserépmennyiséggel lehetséges. Ez a kétszeres szorzó az átlagidőkben is jól tükröződik.

A második figyelemre méltó különbség forrásai a raktár elrendezési módok. Ezekben az esetekben a rakatoszlopok tengelytávolsága, valamint a forgatás szükségessége is különbözött. Mindkettő a második oszlopban található szituációk idejét rontó tényező, ám ezen a ponton meg kell különböztetni a tengelytávolság különbségéből származó és a forgatásból származó időtöbbletet. A rakatoszlopok vonala mentén történő sínes mozgás a lehető leggyorsabb a robot összes mozgásformája közül, csak oszloponként növekszik és a modellben szereplő kevés oszlop következtében csekély. Ezzel szemben a forgatás kezdetektől fogva minden egyes elemet érint, és külön műveletként ékelődik be a programba. Mivel a robot fizikai mozgást végez, egy külön mozgási művelet elkezdése rendszerint több időt igényel, mint egy meglévő mozgás folytatása. Ezért valószínűsíthető, hogy a kettő közül a forgatási idő adja a különbség szignifikánsabb hányadát.

Figyelembe véve, hogy egy nagyobb elemszámot igénylő, kiterjedtebb tetőfelület esetén több rakatoszlop sor kialakításával a párhuzamos és a merőleges elhelyezés közötti különbség gyakorlatilag ki is egyenlítődik, az adatsor leghasznosabb tanulsága a forgatási idő felismerése, mint nem elhanyagolható, az elemszám emelkedésével halmozódó időnövelő tényező. Fontos

ez azért is, mivel az elemek preferált szállítási és tárolási módjai okán az életszerűséghez továbbra is közelebb áll az a munkafolyamat, amelyben forgatás szerepel.

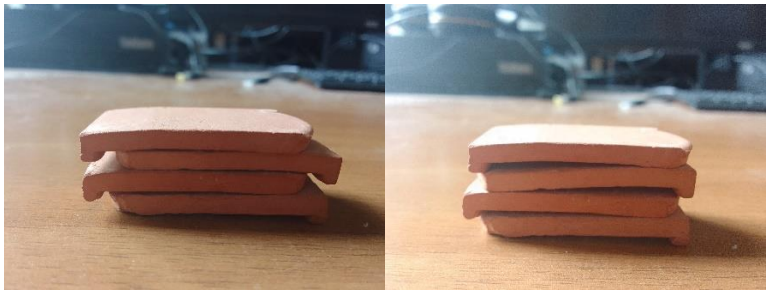


12. ábra: Elem forgatása elhelyezési pozícióba

5 Kitekintés

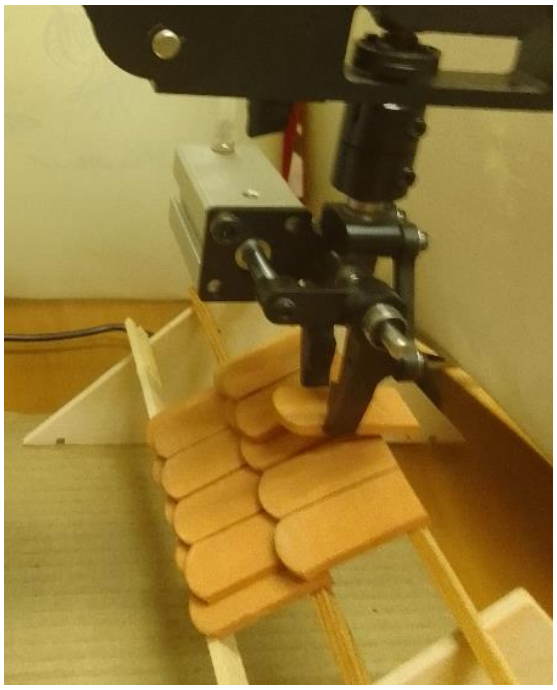
Az előbbiekben bemutatott munkafolyamaton a jövőben számos tekintetben lehetne optimalizálást végezni.

Jelentős fejlesztés lenne például valamilyen szenzor alkalmazása, melynek segítségével a robot önállóan is azonosíthatná az elemeket és a tetőléceket. A leírt vizsgálatban minden egyes műveleti elem helye egy kalibrációs úton kimért kezdőértéktől függött. A kezdőértékben foglalt pontatlanság az ismétlésszám növekedésével halmozódhatott, illetve az esetlegesen előforduló hibás, vagy pontatlanabbul raktározott elemek sem voltak megfelelően kezelhetők. A 13. ábrán megfigyelhető például, hogy az oszlopmagasság nem minden esetben egyenlő az alkotóelemek vastagságának és számának szorzatával. Ezt a problémát jelen kísérlet során gondos válogatással, raklapösszeállítással és több elem méréséből számított átlagokkal kezeltük, de a valódi építőiparban az ilyen körülményes módszerek természetesen kerülendők, egy adott építési technológia elvi szintjén legalábbis bizonyosan.



12. ábra: Elem halmozási pontatlanságok

További oka az elemek pontatlan elhelyezésének a felvett, vízszintes síkú elem és a ferde tetősík közötti eltérés, mely a tetősík dőlésszögével értelemszerűen fokozódik. Egy tetősíkkal párhuzamosan érkező elem valószínűsíthetően jobb esélyekkel rendelkezne a tetőn maradásra (lásd 13. ábra). Ezt pl. 6 szabadságfokú robotkarral lehetne elérni, amely nemcsak vízszintes, hanem függőleges síkban is képes a megfogott elem forgatására. 4 szabadságfokú kar esetén megkerülhető lenne a probléma valamilyen külső elemátfordító szerkezet, vagy egy speciálisan erre a célra kialakított end-effektor hozzáadásával.



13. ábra: Az érkező elem és a tetősík szögének viszonya

Az elhelyezési idő kapcsán mindenképpen megfontolandó az elemek raktárból egyesével történő felvételének kiküszöbölése. Kisméretű lapos elemek esetén a munkafolyamat

nagymértékben gyorsítható több elem egyidejű felvételével, és egy olyan robotkar segítségével, amely képes több, ideiglenesen tárolt elem egyenként való elhelyezésére.

A kutatásnak közeljövőben megvalósítható folytatása lehetne egy több raktársort használó elrendezés megvizsgálása. Ez jelenleg sajnos több, a robot kialakításával kapcsolatos akadályba ütközött. A kar félgömb alakú, felfelé szűkülő műveleti tere egyáltalán nem volt praktikus a tető felfele haladva egyre távolodó lécei szempontjából. A merőleges elrendezésben egyáltalán nem érte volna el egy kétsoros raktár esetén a második sort, a tető közelebb húzását lehetővé tevő párhuzamos elrendezésnél pedig a gripperre szerelt szervó motor helyigénye jelentkezett a forgatási műveleteknél (lásd 14. ábra). A feltételeket ehhez a fejlődési lépéshez egy nagyobb hatótávolság karral, illetve egy optimalizált méretű/formájú gripper forgató motorral felszerelt robot teremthetné meg. A gripper forgató motor eltérő konfigurációjával egyébként a 180°-os forgatás, azaz a merőleges elrendezés forgatással kiegészített változata is vizsgálhatóvá válna, amely szintén egy hiánypótló adatot szolgáltatna a kapott eredmények teljességéhez.

14. ábra: A gripper forgató motor helyigénye

6 Források

6.1 Szöveges tartalom

[1] Grau, A., Indri, M., Bello, L. L., & Sauter, T. (2020). Robots in industry: The past, present, and future of a growing collaboration with humans. *IEEE Industrial Electronics Magazine*, *15*(1), 50-61.

[2] Balaguer, C., Gambao, E., Barrientos, A., Puente, E. A., & Aracil, R. (1996, June). Site assembly in construction industry by means of a large range advanced robot. In *Proc. 13th Int. Symp. Automat. Robotics in Construction (ISARC'96)* (pp. 65-72).

[3] automatedarchitecture.io: AUAR (letöltve: 2022.09.01.)

[4] dfabhouse.ch: DEFAB HOUSE (letöltve: 2022.09.01.)

[5] www.fbr.com.au: HADRIAN X (letöltve: 2022.09.01.)

[6] Romano, M. M., Chen, Y., Kuevor, P., Marshall, O., & Atkins, E. (2021). Nailed It: Autonomous Roofing with a Nailgun-Equipped Octocopter. In *AIAA AVIATION 2021 FORUM* (p. 3211)

[7] www.leister.com: LEISTER (letöltve: 2022.09.01.)

[8] Patel, P., Reddy, D., Saha, S., & Ulm, B. (2014). Team RiPUS: Roboticists Involved in Putting Up Shingles, Carnegie Mellon University

6.2 Ábrák

1. automatedarchitecture.io: AUAR (letöltve: 2022.09.01.)
2. dfabhouse.ch: DEFAB HOUSE (letöltve: 2022.09.01.)
3. www.fbr.com.au: HADRIAN X (letöltve: 2022.09.01.)
4. Romano, M. M., Chen, Y., Kuevor, P., Marshall, O., & Atkins, E. (2021). Nailed It: Autonomous Roofing with a Nailgun-Equipped Octocopter. In *AIAA AVIATION 2021 FORUM* (p. 3211)

5. Patel, P., Reddy, D., Saha, S., & Ulm, B. (2014). Team RiPUS: Roboticists Involved in Putting Up Shingles, Carnegie Mellon Unuversity

6– 14. saját ábrák

7 Kódrészlet

```
# Meroleges elrendezes, egyszeres fedes

import time
import math

def depobol_felvesz():
        global n_oszlopbol_felvett_elemek, n_elem_oszlop, posY_oszlop, y_oszlopok, posZ_elem,
        z_elem
        if n_oszlopbol_felvett_elemek == n_elem_oszlop:
                posY_oszlop = posY_oszlop + y_oszlopok
                n_oszlopbol_felvett_elemek = 0
        current_pose = dType.GetPose(api)
        dType.SetPTPWithLCmdEx(api, 1, current_pose[0], current_pose[1], current_pose[2],
        current_pose[3], posY_oszlop, 1)
        posZ_elem = -110 + (z_elem * (n_elem_oszlop - n_oszlopbol_felvett_elemek) - z_elem /2)
        dType.SetPTPCmdEx(api, 0, 200, 0, posZ_elem, 0, 1)
        dType.dSleep(1000)
        dType.SetEndEffectorGripperEx(api, 1, 1)
        dType.dSleep(1000)
        current_pose = dType.GetPose(api)
        dType.SetPTPCmdEx(api, 2, 200, 0, 0, current_pose[3], 1)
        n_oszlopbol_felvett_elemek = n_oszlopbol_felvett_elemek + 1

def elemet_lerak():
        global n_jelenlegi_sor, n_sorban_eddig_lerakott_elemek, y_elem, posZ_sor, posX_sor,
        n_elem_tetosor, uj_sor
        depobol_felvesz()
        if n_jelenlegi_sor % 2 == 1:
                current_pose = dType.GetPose(api)
                dType.SetPTPWithLCmdEx(api, 1, current_pose[0], current_pose[1],
        current_pose[2], current_pose[3], (n_sorban_eddig_lerakott_elemek * y_elem), 1)
        if n_jelenlegi_sor % 2 == 0:
                current_pose = dType.GetPose(api)
                dType.SetPTPWithLCmdEx(api, 1, current_pose[0], current_pose[1],
        current_pose[2], current_pose[3], (n_sorban_eddig_lerakott_elemek * y_elem + 0.5 *
        y_elem), 1)
        current_pose = dType.GetPose(api)
        dType.SetPTPCmdEx(api, 2, 200, 0, posZ_sor, current_pose[3], 1)
        current_pose = dType.GetPose(api)
        dType.SetPTPCmdEx(api, 2, posX_sor, 0, posZ_sor, current_pose[3], 1)
        dType.dSleep(1000)
        dType.SetEndEffectorGripperEx(api, 1, 0)
        dType.dSleep(1000)
        current_pose = dType.GetPose(api)
        dType.SetPTPCmdEx(api, 2, 200, 0, posZ_sor, current_pose[3], 1)
        current_pose = dType.GetPose(api)
        dType.SetPTPCmdEx(api, 2, 200, 0, 0, current_pose[3], 1)
        n_sorban_eddig_lerakott_elemek = n_sorban_eddig_lerakott_elemek + 1
        if n_sorban_eddig_lerakott_elemek == n_elem_tetosor:
                uj_sor = True

def sort_lerak():
        global uj_sor, n_jelenlegi_sor, n_sorban_eddig_lerakott_elemek, posX_sor, x_sor,
        posZ_sor, z_sor, n_elem_tetosor
        uj_sor = False
        n_jelenlegi_sor = n_jelenlegi_sor + 1
        n_sorban_eddig_lerakott_elemek = 0
        if n_jelenlegi_sor > 1:
                posX_sor = posX_sor + x_sor
                posZ_sor = posZ_sor + z_sor
        if n_jelenlegi_sor % 2 == 0:
                n_elem_tetosor = n_elem_tetosor - 1
        if n_jelenlegi_sor % 2 == 1 and n_jelenlegi_sor != 1:
                n_elem_tetosor = n_elem_tetosor + 1
```

```
        while uj_sor == False:
                elemet_lerak()

start = time.time()
dType.SetDeviceWithL(api, 1, 1)
dType.SetEndEffectorParamsEx(api, 59.7, 0, 0, 1)
y_oszlopok = 40
posY_oszlop = 0
z_elem = 5
n_elem_oszlop = 4
n_oszlopok = 4
y_elem = 18.5
n_elem_tetosor = 5
x_sor = 22
z_sor = 22
posX_ereszsor = 284
posZ_ereszsor = -25
posX_sor = posX_ereszsor
posZ_sor = posZ_ereszsor
uj_sor = False
n_jelenlegi_sor = 0
dType.SetEndEffectorGripperEx(api, 1, 0)
n_oszlopbol_felvett_elemek = 0
n_sorok = 2
current_pose = dType.GetPose(api)
dType.SetPTPCmdEx(api, 2, 200, 0, 0, current_pose[3], 1)
for count in range(int(n_sorok)):
        sort_lerak()
current_pose = dType.GetPose(api)
dType.SetPTPCmdEx(api, 2, 200, 0, 0, current_pose[3], 1)
current_pose = dType.GetPose(api)
dType.SetPTPWithLCmdEx(api, 1, current_pose[0], current_pose[1], current_pose[2],
current_pose[3], 0, 1)

end = time.time()
runtime = end-start
print(runtime)
hours, rem = divmod(runtime, 3600)
minutes, seconds = divmod(rem, 60)
print("{:0>2}:{:0>2}:{:05.2f}".format(int(hours),int(minutes, seconds))
```