MOZGÁS FELNAGYÍTÁS SZERKEZETEKNÉL, IRODALMI ÁTTEKINTÉS, ALKALMAZÁS

MOTION ENLARGEMENT AT STRUCTURES, LITERATURE REVIEW, APPLICATION

Bozzay Péter[*]*, Bodnár Dávid*[**]*, Dr. Jármai Károly*[***]

ABSTRACT

This paper demonstrates how novel motion-enhancing computer vision techniques can use high-speed video cameras to see and quantify the vibrational mode shapes of simple structures. Motion magnification is an algorithm designed to magnify the subtle movements of video footage. The basic principle of the algorithm is to obtain a representation of the video that allows time-frequency bandpass filtering, amplification and reconstruction of the signals corresponding to the motion of objects. This process results in a video in which the apparent motion of objects is amplified within a given frequency band.

1. BEVEZETÉS

A mechanikai rezgések életünk számos területén jelen vannak, bizonyos esetekben hasznosak számunkra, azonban sok esetben csupán valamilyen berendezés üzemeltetésének többé-kevésbé káros melléktermékei, melyek mértéke az elhanyagolhatótól a katasztrofális hatású szintig terjed, gondoljunk például az utcán elhaladó autó hatására megzörrenő ablakra, vagy a Tacoma híd jó ismert esetére.

Rezgésvizsgálatra többféle módszer áll rendelkezésre. A humán érzékelés tapintás, hallás vagy látás alapján legfeljebb a szint és a frekvencia szubjektív detektálására és összehasonlítására alkalmas. A mechanikus valamilyen finommechanikai szerkezetre alapuló, egyszerűbb mérésre és regisztrálásra is alkalmas, mára elavult technológia. Leginkább elterjedt az elektronikus, gyorsulás, elmozdulás vagy deformáció mérésére alkalmas érzékelővel és analóg-digitális-szoftveres mérő-jelfeldolgozó rendszerrel kialakított, nagy pontosságú, akár sokcsatornás mérésre és regisztrálásra is alkalmas technológia. Ezek mellett a digitális képfeldolgozó hardver-szoftver technológia nagy fokú fejlődése lehetővé tette egy újfajta, optikai megoldás létrejöttét, ahol a szabad szemmel (szinte) láthatatlan mozgások, deformációk is megmutathatók, azok szoftveres felnagyításával. Ennek gyakorlati alkalmazási lehetőségeire kerestünk megoldásokat a rendelkezésünkre álló eszközrendszerrel.

Ez a cikk bemutatja, hogy az újszerű mozgásnagyító számítógépes látásmódok hogyan képesek nagysebességű videókamera segítségével egyszerű szerkezetek rezgési módusformáit meglátni. A mozgásnagyítás egy olyan algoritmus, amelyet arra terveztek, hogy a videofelvételek finom mozgásait felerősítse. Az algoritmus alapelve a videó olyan reprezentációjának megszerzése, amely lehetővé teszi a tárgyak mozgásának megfelelő jelek időfrekvenciás sávszűrését, felerősítését és rekonstruálását. Ez a folyamat olyan videót eredményez, amelyben a tárgyak látszólagos mozgása egy adott frekvenciasávon belül felerősödik.

2. A FÁZISALAPÚ MOZGÁSNAGYÍTÓ ALGORITMUS

A fázisalapú mozgásnagyító algoritmus a videójeleket egy komplex értékű, irányítható piramisszűrő bank segítségével helyi térbeli amplitúdóra és fázisra bontja. A helyi térbeli fázisjeleket időbeli Fourier-dekompozícióval szinuszfüggvények sorozatává bontja, amelyek harmonikus mozgást képviselnek. A fázisjelek időbeli sávszűrésen, erősítésen és rekombináción mennek keresztül, hogy mozgásnagyított videót állítsanak elő. Az eredmény az, hogy a videó egy meghatározott időbeli frekvenciatartományon belül a mozgás nagyítását mutatja.

Képpiramisok egy olyan többskálás ábrázolás, amely rekurzívan szétválasztja a kép magas és alacsony frekvenciájú komponenseit, létrehozva egy olyan frekvenciasáv bankot, amely egyértelműbben azonosítja a képen belül a mögöttes mintázatokat [1,2].

A standard Gauss-piramisban a kép rekurzívan elmosódik és almintavételezik, hogy elkülönítsék a jelkomponensektől [3].

Az összetett irányítható piramis az egyik módszer a jel elhalványulásának és a piramis rekonstrukciós hibáinak csökkentésére [4].

A CSP hasonló felépítésű, mint a Gauss-piramis, de az egyes piramisszintek létrehozása előtt a térbeli

* *tanszéki mérnök, Miskolci Egyetem Energetikai és Vegyipari Gépészeti Intézet*
** *mérnök, Emerson Automation FCP Kft. Eger, PhD hallgató, Miskolci Egyetem, Energetikai és Vegyipari Gépészeti Intézet*
*** *egyetemi tanár, Miskolci Egyetem Energetikai és Vegyipari Gépészeti Intézet*

tartományt át kell váltani a frekvenciára [5]. A komplex irányítható piramis alternatívája a Riesz-piramis. A Laplace-piramisra (a Gauss-piramis és az eredeti kép közötti különbség) vízszintesen és függőlegesen is közelítő Riesz-transzformáció alkalmazásával a frekvenciatartalom kvaternionos ábrázolása jön létre. A Wadhwa [6] által bemutatott Riesz-piramis jobb inverzióval rendelkezik, mint a komplex irányítható piramis, és jelentősen csökkenti a számítási időt, cserébe a piramisszintek felépítésének pontosságában bekövetkező kis hibákért.

A videóban rezgő tárgyak modális frekvenciáinak kinyeréséhez először a Liu által kifejlesztett C++ optikai áramlási kód MATLAB fájlját használják [7]. A spektrális teljesítmény sűrűség (PSD) kiszámítására és megjelenítésére szolgáló programmal a kapott frekvenciaválaszokat a MATLAB diszkrét Fourier-transzformációs függvényeinek felhasználásával írták meg.

Az eljárás átfogó magyarázatát és levezetését a [8,9] hivatkozások tartalmazzák. Egy rezgő szerkezet esetében egy megfelelő frekvenciasáv kiválasztásával olyan videót kapunk, amely a működési alakváltozást a közelítő rezonanciafrekvenciák egyikén ábrázolja, feltéve, hogy a módusok megfelelő távolságra vannak egymástól. A szorosan összekapcsolt módusok esetében a nagyobb amplitúdót mutató módus valószínűleg vizuálisan dominál [10].

A [11] hivatkozás részletesen elemzi az üzemi kitérési alakzatok és a klasszikus módusalakzatok közötti különbségeket és értelmezéseket. Egy szerkezet működési rezgésmódjainak ez a minőségi vizualizációja alapvető fontosságú a geometriailag összetett szerkezetek esetében.

Az Euler-alapú lineáris közelítési módszer egy tér-időbeli folyamat révén javítja a finom színeket és az észrevehetetlen mozgásokat, hatékonyan feltárva mind a finom térbeli jeleket, mind az időbeli színváltozásokat a videóban. Ez a módszer egyszerű és hatékony, azonban vannak bizonyos korlátai. A komplex irányítható piramist használó fázisalapú videófeldolgozás a kis mozgások feldolgozásához különböző léptékekben és orientációban elemzi a helyi fázist az időben, és felerősíti az időbeli fáziskülönbséget a megfelelő sávban. A fázisalapú technika zajmentes eredményeket ad, és kiváló minőségű, fotorealisztikus videókat készít, felerősített mozgással. A gyors fázisalapú videófeldolgozás a Riesz-piramist alkalmazza, elsősorban a fázisalapú videónagyítás sebességére helyezi a hangsúlyt a minőség megőrzése mellett [12]. A Riesz-piramis megkönnyíti a fázisalapú mozgásnagyítás valós idejű végrehajtását. A továbbfejlesztett Euler-videónagyítás a képtorzítás révén felerősíti az időbeli videómozgást, felhasználva az előzetes mozgástérképezést [13]. Az E2VM hatékony mozgásnagyítási technikát vezet be, amely tér-időbeli szűrés és képtorzítás révén minimalizálja a zajt.

Euler-féle mozgásnagyításnál a képfeldolgozás képes a kisebb mozgások felerősítésére, annak ellenére, hogy nem alkalmazza a mozgáskövetést, mint a Lagrange-módszereknél [5]. Ebben a szakaszban történik az időbeli feldolgozáson keresztül a mozgásnagyítás, az optikai áramláselemzésekben jellemzően alkalmazott elsőrendű Taylor-soros kiterjesztéseken alapuló elemzést használva [7].

Az időbeli feldolgozás és a mozgásnagyítás közötti kapcsolat egy transzlációs mozgást tapasztaló egydimenziós jel vizsgálatán keresztül szemléltethető. Ez az elemzés közvetlenül alkalmazható a két dimenzióban lokális transzlációs mozgásra [14].

Zajérzékenység vonatkozásában a következő állapítható meg. Az érdeklődésre számot tartó jel amplitúdóváltozása gyakran jelentősen kisebb marad, mint a videóban jelenlévő eredendő zaj. Ezekben az esetekben a pixelértékek közvetlen feljavítása nem hozza felszínre a kívánt jelet. A térbeli szűrés a finom jelek feljavítására szolgál. Ha azonban az alkalmazott térbeli szűrő nem elég nagy, az érdekes jel elfedve marad [15].

3. ESZKÖZÖK

A rendszer három fő komponensből áll. Kritikus eleme a kamera, amelynek képminősége alapvetően befolyásolja a feldolgozás végeredményét. A képérzékelő adott lapkamérettel és képfelbontással rendelkezik, egy adott nyílásszögű objektív erre vetíti a vizsgált objektum képét, az analóg jelerősítő lánc óhatatlanul rendelkezik valamekkora saját zajjal, továbbá a digitalizálási folyamatból (mintavételezés, kvantálás, képtömörítés) és a feldolgozó algoritmus implementációjából (szemantika, számok bitszélessége) is adódik kis mértékben változó képtartalom, mely nem valós mozgásként szintén feldolgozásra kerül.

Fontos szempont továbbá, hogy lehetőség szerint a vizsgálat szempontjából fontos objektumrész minél jobban töltse ki a képmezőt, melyet a távolság és az objektív nyílásszöge határoz meg. Például az általunk tesztelt Chronos CR21 gyorskamera képszenzorának mérete 29,5 x 16,6 mm, felbontása 1920 x 1080 pixel, ez az alkalmazott 50 mm-es objektívvel 1 m tárgytávolságnál 0,182 mm/pixel felbontásnak felel meg (amely a tárgytávolsággal egyenes arányban változik), ez azt jelenti, hogy ettől kisebb elmozdulást a rendszer nem képes érzékelni ebből a távolságból. Emellett végeztünk teszteket beépített, USB és ethernet webkamerákkal, illetve mobiltelefon kamerájával, vegyes minőségű eredménnyel főként képzaj szempontjából. A minél hatásosabb zajcsökkentés érdekében a lehető legerősebb, még túlvezérlést (kép beégést) nem okozó, villogásmentes fényforrás alkalmazása is nagyon fontos. Második fő komponens a feldolgozást végző számítógép.

A fenti algoritmusok képfelbontástól függően nagy képpont mátrixokkal végeznek összetett műveleteket (jobb végeredmény – nagyobb képfelbontás – nagyobb mátrixok), így mind számítási teljesítményben, mind RAM kapacitásban magasak lehetnek az igények, ennek mértéke a képfelbontástól, a szoftveres fejlesztői platformtól és az algoritmus konkrét implementációjától is függ, illetve függ az adott processzor felépítésétől is. Bizonyos implementációk képesek nem csak CPU, hanem GPU (jellemzően nVidia) alapú számításokra is, amely jelentősen meggyorsíthatja az adatfeldolgozást. Az operációs rendszer a tesztelt projektek egy részénél lehet Windows és Linux is, másoknál csak egyik vagy másik, mi az adott célra sokkal rugalmasabban használható Linuxot választottuk.

Harmadik fő komponens az a szoftver, amely az adott algoritmust működteti a számítógépen. Ennek egyik fő összetevője a magát az algoritmust megvalósító programkód, a másik azon függvénykönyvtárak összessége, amelyek külső forrásból rendelkezésre állnak a felhasználásra. Utóbbiak ebben az esetben jellemzően képfeldolgozás (gépi látás), jelfeldolgozás és mátrix műveletek területén általánosan használható függvénykönyvtárak, mint pl. az OpenCV, Tensorflow, NumPy, SciPy, PIL, stb, illete az nVidia GPU-n futtatáshoz a CUDA és a grafikus kezelőfelülethez pl. a Qt.

Léteznek erre a feladatra termékszintű, piaci megoldások is. Ezek között megtalálható a fentiek szerinti (célra optimalizált) kamera + számítógép + szoftver alapú is, illetve olyan speciális kamera is, amely önmagában, egy beépített, FPGA alapú célszámítógépben hardveresen implementálja az algoritmust, így a CPU/GPU alapú megoldásokhoz képest lényegesen gyorsabb feldolgozásra képes, melyet egy kiváló minőségű és felbontású képfelvevő egységgel kombinálva rendkívül realisztikus eredményt lehet elérni. Cserébe ezek ára rendkívül magas. pl. a piacvezető RDI Technologies termékei 30e EUR ár körül indulnak (1. ábra). Ez adta a motivációt, hogy nézzük meg a fellelhető nyílt forráskódú megoldások sajátosságát, képességét, rendszerigényét.

1. ábra. RDI kamera Forrás: rditechnologies.com

4. PROJEKTEK

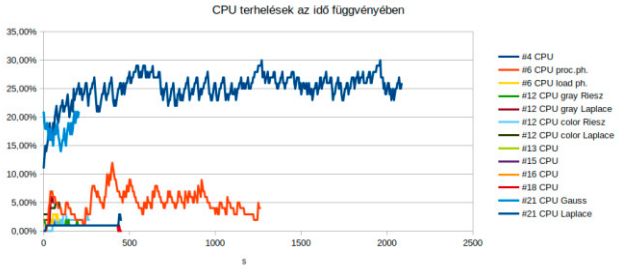
Az Internetről összesen 24 olyan projektet gyűjtöttünk össze, amelyek futtatását általunk is reprodukálhatónak ítéltük meg. Ezek négyféle programozási nyelven készültek, a megadott hivatkozási sorszámokkal:
- 4 db Matlab (1, 17, 19, 22)
- 6 db C++ (12, 13, 14, 15, 16, 21)
- 13 db Python (2, 3, 4, 5, 7, 8, 9, 10, 11, 18, 20, 23, 24)
- 1 db ImageJ (6)
- Készült továbbá egy saját, egyszerűbb algoritmussal működő kísérleti megoldás is Python-ban.

A Matlab verziókat a költséges licencek kötöttsége és a Linuxos telepítési nehézségek miatt félretettük, a C++ esetében a QtCreator + gcc, Python-hoz a PyCharm + Python 3 fejlesztőkörnyezetet, míg ImageJ-hez a Fiji verzió + EVM plugin-t használtuk.

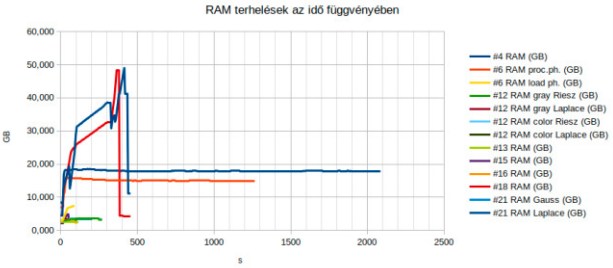
Hardver oldalról a projekteket az alábbi gépeken teszteltük:
- Sun Microsystems SunFire X4600 szerver (AMD CPU-k, 8x4 mag, 256 GB RAM)
- Dell T3600 Workstation (Intel Xeon CPU, 8 mag, 32 GB RAM)

Az nVidia videovezérlő sajnos egyik esetben sem CUDA kompatibilis, így a GPU nem használható a számításokhoz.

A szerveres tesztek során látható volt, hogy egyik projekt sem használja ki a nagyszámú CPU magot (2. ábra), a használt mag(ok) és a RAM terhelése (3. ábra), illetve a feldolgozási idő pedig számottevő szórást mutat. Ezek regisztrálására egy saját shell scriptet használtunk, amely 5s-onként rögzítette ezeket az értékeket az alábbi diagramok szerint.



2. ábra CPU terhelések az egyes programoknál



3. ábra RAM terhelések az egyes programoknál

Bizonyos projektek csak rögzített videofájl feldolgozására alkalmasak, néhányuk kamera videostream közvetlen kezelésére is.

A forráskódok fordítása során a legtöbb esetben kisebb-nagyobb nehézségekbe ütköztünk, főként a külső függvénykönyvtárak és a fejlesztőrendszerek időközben bekövetkezett verzióváltásai és egyéb kompatibilitási problémák miatt, így sok esetben kisebb-nagyobb módosításokat kellett végezni a forráskódokban, néhány projekt esetén sajnos sikertelenül.

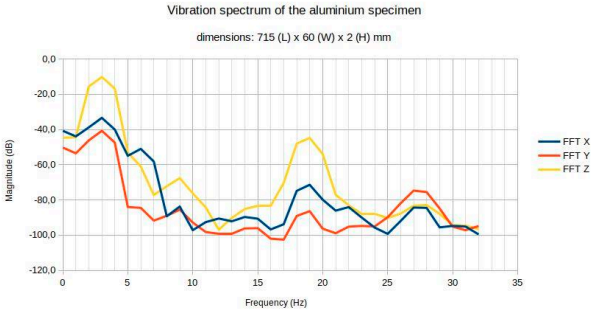
Az összehasonlíthatóság érdekében minden esetben ugyanazt a videofájlt és ugyanazokat a feldolgozási paramétereket használtuk. A próbatest egy egyik végén rögzített lemez, amelyet egy éppen látható mértékű lengést okozó gerjesztőimpulzus ért és ezt a mozgást rögzítettük kamerával (4. ábra).

4. ábra A teszt videofile

5. ábra A teszt videofile feldolgozva

Egy véletlen folytán a jelölt terület külön jelentőséget kapott a tesztelésnél. Gyakorlatilag csak a feldolgozott videón látszott, hogy a felvétel közben ezen a területen egy hangya haladt át (5. ábra). Mint később kiderült, ez a láthatóság projekt- és paraméterfüggő, így ez is a minősítés egyik tényezője lett.

Mivel az algoritmusok frekvencia sávszűrést is alkalmaznak a nemkívánatos változások figyelmen kívül hagyása érdekében, így fontos a vizsgálandó mozgás frekvenciatartományának ismerete, melyet a képen látható gyorsulásérzékelővel és egy NI CompactDAQ műszerrel mértünk meg. A diagramon látható, hogy az első sajátfrekvenciája 3 Hz, ami 30 fps sebességű felvétel esetén jól feldolgozható (6. ábra).



6. ábra A mozgás frekvenciatartománya

A frekvencia sávszűrés egyúttal azt is jelenti, hogy a mintavételi törvény ebben az esetben is releváns, a felvétel (az adott kamera) sebessége korlátozza a vizsgálandó mozgás frekvenciáját. Egy általános célú kamera 25...60 fps sebességgel működik, ami így közvetlenül nem teszi lehetővé a magasabb frekvenciatartományú vizsgálatokat. Ilyen esetben hasznos egy gyorskamera alkalmazása (esetünkben ez a Chronos CR21 lehet), amely alkalmas akár több ezer fps felvételre, ugyanakkor ennek egy átlagos kamera sebességére transzformált videofájl készítésére is, ebben az esetben a feldolgozás során az $f_{feldolgozási} = f_{valós} * (FPS_{videofájl} / FPS_{rögzítés})$ frekvenciát kell paraméterként használni. Például egy 50 Hz-es rezgőadagoló esetében 600 fps rögzítés és 30 fps videofájl mellett a feldolgozáshoz beállítandó sajátfrekvencia 50 * (30 / 600) = 2,5 Hz.

A projektek feldolgozási minőségének összehasonlítására egy részben objektív, részben szubjektív minősítési rendszert alakítottunk ki, mely figyelembe veszi a rendszerterhelést, a futásidőt, a képzajt, a mozgásnagyítás vizualitását, a hangya láthatóságát, illetve a kezelhetőséget, paraméterezhetőséget, futtathatóságot is. Az eredményeket a működésre bírt projektek esetében az 1. táblázat foglalja össze:

1. táblázat A vizsgált projektek

Projekt #	Nyelv	Futásidő (s)	RAM terhelés max (GB)	CPU terhelés max	Videó szubjektív osztályzat (0...5)	Összesített értékelés (0...5)	Tapasztalatok
2 és 11 [16]	Python	N.A.	N.A.	N.A.	0	0	Grafikus ablak, a két projekt lényegében ugyanaz, kamerát igényel, real-time, elvileg GPU alapú, de CPU-val is működik, viszont azzal nem az elvárt jellegű eredményt nyújtja, így a terhelés teszt nem volt indokolt.
4 [17]	Python	420	49,0	3%	4 (H4)	3	Grafikus kezelőfelület, szerveren működik, PC-n kevés a 32 GB RAM.
6 [18]	ImageJ	>1000	15,9	12%	3 (H0)	2	Grafikus kezelőfelület, a bemenő videót képekké kell konvertálni (ehhez készült saját python utility), majd az eredmény képeket videóvá, csak ff. eredményt ad. A hangya mozgása nem látható.
8 [19]	Python	(>5200)	N.A.	N.A.	N.A.	2	Szerveren CPU utasításkészlet inkompatibilitás miatt nem működik, de PC-n igen, viszont rendkívül lassan és nagy terhelés mellett, pl. az MIT alacsony felbontású demo videóját 50% méretben is >5200 s alatt dolgozta fel, így további tesztelés még nem történt.
12 [20]	C++	75	3,3	6%	4 (H4)	5	Grafikus kezelőfelület, sokoldalú, Laplace és Riesz módszerrel is működik, színes és ff, kamerát is kezel, a Laplace módszerrel gyakorlatilag real-time módban. (a 4 sor: Laplace ff, Riesz ff, Laplace színes, Riesz színes) A Riesz módszerrel plasztikusabb kimenet, de a hangya mozgása nem látható.
		270	3,7	2%	5 (H0)	4	
		120	3,4	5%	4 (H4)	5	
		270	3,7	3%	5 (H0)	4	
13 [21]	C++	130	2,6	2%	4 (H4)	5	Grafikus kezelőfelület, az összes közül a legalacsonyabb rendszerterhelés
15 [22]	C++	80	4,9	2%	4 (H4)	4	Grafikus ablakok, az elkészített videót nem menti fájlba, futtatáskor csak mutatja párhuzamosan a bemeneti videóval. Ez real-time működést is feltételezhet, de a tesztelő gépeken ezt nem képes tartani. A 11 s hosszú videó futásideje 80 s.
16 [23]	C++	125	3,0	2%	5 (H0)	4	Parancssoros, a futási üzenet arra utal, hogy elvileg tudna kezelni GPU-t is. Plasztikusabb kimenet, de a hangya mozgása nem látható.
18 és 20 [24]	Python	365	48,3	2%	3 (H0)	2	Parancssoros, az eredeti MIT algoritmus python implementációja, az MIT demo videó és a saját tesztvideó erőforrásigény különbsége kiugróan nagy. A viszonylag zajos kimenet mellett a hangya mozgása sem látható.
21 [25]	C++	185	3,4	21%	2 (H2)	2	Parancssoros, jelentős CPU mag terhelést mutat, ez a paraméterezés függvényében 100%-ig is ment, a program leállását is okozva. (A 2 sor: Gauss és Laplace módszer)
		>2000	18,6	30%	3 (H1)	1	
BP5	Python	N.A.	N.A.	N.A.	2 (H3)	2	Saját fejlesztésű kísérleti program, más, egyszerűbb működési elven, amely inkább viszonylag nagyobb mozgásokhoz alkalmas, de a hangya mozgása látható. További fejlesztést igényel, így a terhelés teszt még nem volt indokolt. („motext” néven az online videók között)

Az egyes projektekkel készült feldolgozott videók az eredeti felvétellel együtt az alábbi linken érhetők el: https://www.youtube.com/playlist?list=PLU6qsdcgtHXsKhNUky6DUzNs4zQoaZcdz

További 12 projekt is nagy valószínűséggel működőképessé tehető, megfelelő MATLAB licenc, CUDA kompatibilis hardver, vagy/és a fejlesztésekor alkalmazott pontos szoftverkörnyezet használatával, melyek jelenleg nem állnak rendelkezésünkre. Ugyanakkor ebben a fázisban ez nem is indokolt, mivel a működő projektek között is vannak olyanok, amelyek megfelelnek a módszer gyakorlati hasznosításának további teszteléséhez.

5. ÖSZEFOGLALÁS

A projekteknek (és magának a Sun szervernek erre a célra) történt használatba vétele során számos nehézségen kellett keresztülmenni, melyek nagy részének bemutatását e cikk keretei közt nem tartottuk szükségesnek. Az eddig elért eredményekből látható, hogy az alkalmazott algoritmuson túl több hardver és szoftver tényező is jelentősen befolyásolja az egyes projektek gyakorlati használhatóságát, akár működőképességét. A megszerzett tapasztalatok alapján a bemutatott projektek közül összességében jelenleg a 12, 13 és 16 számút tartjuk a legalkalmasabbnak további, számos ipari területen előforduló kismértékű mozgások elemzésével történő tesztelésre, majd gyakorlati, oktatási-kutatási célú hasznosítására.

6. KÖSZÖNETNYILVÁNÍTÁS

A bemutatott kutató munka részben a C2307874 számú projekt a Kulturális és Innovációs Minisztérium Nemzeti Kutatási Fejlesztési és Innovációs Alapból nyújtott támogatásával, a KDP-2023 pályázati program finanszírozásában valósult meg.

7. IRODALOM

[1] Burt, P., Adelson, E. 1983. The Laplacian pyramid as a compact image code. IEEE Trans. Comm. 31, 4, 532–540.

[2] Freeman, W. T., Adelson, E. H., Heeger, D. J. 1991. Motion without movement. ACM Comp. Graph. 25, 27–30.

[3] Fuchs, M., Chen, T., Wang, O., Raskar, R., Seidel, H.-P., Lensch, H. P. 2010. Real- time temporal shaping of highspeed video streams. Computers & Graphics 34, 5, 575–584.

[4] Horn, B., Schunck, B. 1981. Determining optical flow. Artificial intelligence 17, 1-3, 185–203.

[5] Liu, C., Torralba, A., Freeman, W. T., Durand, F., Adelson, E. H. 2005. Motion magnification. ACM Trans. Graph. 24, 519–526.

[6] Liu, C., Freeman, W., Szeliski, R., Kang, S. B. 2006. Noise estimation from a single image. In IEEE CVPR, vol. 1, 901 – 908.

[7] Lucas, B. D., Kanade, T. 1981. An iterative image registration technique with an application to stereo vision. In Proceedings of IJCAI, 674–679.

[8] Wadhwa, N., Rubinstein, M., Durand, F., Freeman, W. T., "Riesz Pyramids for Fast Phase-Based Video Magnification" 2014. https://doi.org/10.1109/iccphot.2014.6831820

[9] Simoncelli, E. P., Freeman, W. T., "The Steerable Pyramid: A Flexible Architecture for Multi-scale Derivative Computation," Proceedings of the 2nd IEEE International Conference on Image Processing, Washington, DC, October 1995.

[10] Chen, J. G., Wadhwa, N., Cha, Y.-J., Durand, F., Freeman, W. T., Buyukozturk, O., "Modal Identification of Simple Structures with High-Speed Video Using Motion Magnification," Journal of Sound and Vibration, Vol. 345, June 2015, pp.58-71 https://doi.org/10.1016/j.jsv.2015.01.024

[11] Wadhwa, N., Rubinstein, M., Durand, F., Freeman, W. T., "Phase-Based Video Motion Processing," ACM Trans. Graph. (Proceedings SIGGRAPH 2013), Vol. 32, No. 4, 2013. https://doi.org/10.1145/2461912.2461966

[12] Wadhwa, N., Rubinstein, M., Durand, F., Freeman, W. T., "Quaternionic Representation of the Riesz Pyramid for Video Magnification," 2014.

[13] Kim, H. M., Bartkowicz, T. J., "An Experimental Study for Damage Detection Using a Hexagonal Truss," Computers and Structures, Vol. 79, 2000, pp. 173–182.

[14] Meirovitch, L., Fundamentals of Vibration, McGraw-Hill Companies, Inc., 2001.

[15] Liu, C., Beyond Pixels: Exploring New Representations and Applications for Motion Analysis, Doctoral thesis, Massachusetts Institute of Technology, Cambridge, MA, 2009.

[16] github.com/nedeisenberg/GAMA

[17] www.cg.tuwien.ac.at/courses/Visualisierung2/HallOfFame/2018/Wu2012/html/index.html

[18] github.com/hilaprat/evm-imagej2

[19] github.com/itberrios/phase_based

[20] github.com/tschnz/Live-Video-Magnification

[21] github.com/wzpan/QtEVM

[22] github.com/saracen/magnification

[23] github.com/bramton/opencl-motion-magnification

[24] github.com/flyingzhao/PyEVM

[25] github.com/Rya-Sanovar/Eulerian-Video-Magnification

[26] github.com/chrdiller/VideoMagnification