

Mátrixműveletek integrálása a gpCV++ képfeldolgozási könyvtár fejlesztése során

Prohászka Zoltán and Fodor Péter

Budapesti Muszaki és Gazdaságtudományi Egyetem
Irányítástechnika és Informatika tanszék 1117 Budapest, Magyar Tudósok krt. 2
<http://www.iit.bme.hu/~prohaszka>

Kivonat Cikkünkben a BME IIT-n fejlesztett, nyílt forráskódú gpCV++ programkönyvtár aktuális készültségi állapotát mutatjuk be, elsősorban a mátrixokkal való munka kérdéseit vitatjuk meg. Felsoroljuk az ezekkel kapcsolatos szempontjainkat, és bemutatjuk az általunk vizsgált, leginkább megfelelő meglévő könyvtárak tulajdonságait. Ezután bemutatjuk a választott megoldást, egy minta forráskóddal illusztráljuk a felhasználhatóságot.

Kulcsszavak: GPU C++ könyvtár, numerikus programkönyvtár, gpCV++, mátrix template-ek, TensorLib

1. Bevezetés

A BME Irányítástechnika és Informatika tanszékén 2006 nyarán kezdtünk bele a gpCV++, grafikus gyorsítókártyát használó képfeldolgozó könyvtár fejlesztésébe C++ nyelven. Abban az időben nem állt rendelkezésre hasonló megoldás, mely kitűzött céljainknak megfelelt volna. A programkönyvtár elsődleges célja a GPU-k által kínált számítási teljesítmény hatékony kiaknázása a C++ nyelvtől elvárható kényelmi szolgáltatások biztosításával. Ez a sebesség elsősorban a nagyfelbontású képek, filmek előfeldolgozásához szükséges, de szeretnénk, hogy a három dimenziós, illetve egyéb rekonstrukciós számítások elvégzéséhez is igen kényelmes megoldásokat nyújtson a könyvtár. A projekt kezdeti szakaszáról [1]-ben bővebb információ található.

1.1. Kerti Andor diplomamunkája

Miután a képek GPU-n történő feldolgozásához szükséges részeket implementáltuk és elozetes tesztnek vetettük alá, bonyolultabb feladatokon szerettük volna bizonyítani a könyvtár képességeit. Kerti Andor 2008-as diplomamunkájában [2] bemutatja a megvalósított, Bouguet-féle [3] képjellemző alapú hierarchikus Lucas-Kanade elmozdulásdetektor megvalósítását, mint összetett előfeldolgozási feladatot. Az a feladat volt még ebben az időben a könyvtár szerkezetének az egyszerűsítése, valamint hogy tesztelje a könyvtár platformfüggetlenségét Linux alatt.

2. A lineáris algebrai könyvtár fejlesztése

2.1. Elozmények, motiváció

A képfeldolgozás egyik leginkább számításigényes része az előfeldolgozás, ezt a részt gyorsítjuk a GPU-t használó könyvtárunkkal. Az előfeldolgozás után az adatokat legtöbb esetben lineáris algebrai módszerekkel dolgozzuk fel. Például a QR felbontást nem csak a sajátértékek számításához és least-square¹ feladatok megoldáshoz, hanem kameramátrix meghatározásához, homográfia faktorizálásához is használjuk. Az elmúlt évben az előfeldolgozó rutinok tesztelése és használata során vált nyilvánvalóvá, hogy a mátrixokkal kapcsolatos műveletek támogatását minél hamarabb véglegesíteni kell. Mivel a kép ki- és bevitelre használt OpenCV lineáris algebrai szolgáltatásaival nem voltunk megelégedve, ezért ezt a nyilvánvaló megoldást nem használhattuk. Igen körültekintő döntést kellett hoznunk annak eldöntésében, hogy mit is használjunk helyette.

Eddig egy statikus könyvtárat használtunk, ami a Newmat könyvtárral végeztette a numerikus műveleteket. A statikus jelző arra utal, hogy a vektorokat és mátrixokat fix méretű tömbben tároljuk. Ez a könyvtár nem felelt meg az igényeinknek, mert bár gyors volt, nem támogatott magasabb szintű függvényeket és a használata sem volt igazán kényelmes. Ezek után felállítottunk egy követelmény listát, hogy mi szükséges mindenképpen, illetve opcionálisan egy ideális lineáris algebra könyvtárhoz, majd körülnéztünk az Internet kínálatában.

Az alkalmazandó programkönyvtárral szemben felállított követelményeink a következők voltak:

1. C++ programnyelven legyen megírva, legyenek felüldefiniált mátrix operátorok. A forráskódbeli képletek minél jobban hasonlítsanak a matematikai alakjukra.
2. Licenz: Legyen ingyenesen felhasználható, és be lehessen építeni a gpCV++ könyvtárba. Egyes licenzek szabad felhasználást biztosítanak, de nem engedik, hogy egy csomagban lehessen letölteni a forrásokat. Ez a verzió nem kedvező, mert a jelenlegi gpCV++ verzióhoz már most szükség van az OpenGL, OpenCV, wxWidgets könyvtárakra.
3. Legyen statikus tárolás a kis (de nem korlátozott) méretű mátrixokra.
4. Legyenek dinamikus tárolók.
5. Ne hívjon feleslegesen konstruktorokat, legyen gyors. Lehetőleg konstruktor visszatérés legyen az operátor függvények végén (a felesleges másolások elkerülése érdekében). Legyen valamilyen mechanizmus, ami megakadályozza a felesleges objektum-másolásokat.
6. Legyen paraméterezhető a skalár változó. A sebesség növelése érdekében lehessen float számokkal számolni (2D előfeldolgozás) és csak ott double-lel, ahol tényleg fontos a double pontosság (3D feldolgozás).
7. Legyenek magasabb szintű megbízható numerikus rutinok (sajátérték, QR felbontás stb.). Fontos a numerikus pontosság.

¹ legkisebb négyzetes hiba

8. Külön el lehessen érni a mátrixok és vektorok részeit, például egy homogén mátrix, orientációért felelos részét. A rész mátrixokból össze lehessen állítani egy nagyobb mátrixot (konkatenáció).

Opcionálisan teljesítendo:

9. Lehessen az egyenlőségjel bal oldalára is bizonyos kifejezéseket írni.
10. Legyenek egyenértékű indexekkel rendelkező tenzorok.

Az alkalmazási terület miatt a nagy, de ritka mátrixok külön támogatása és az elosztott számolás lehetősége nem jelent számottevő elonyt.

2.2. A kipróbált könyvtárak

A következő bekezdésekben bemutatunk néhány általunk vizsgált, az Internetről letölthető matematikai könyvtárat. A vizsgálódásunk messze nem volt kimerítő. [5,6].

GSL - GNU Scientific Library. A GSL a GNU projekt része és GPL licensszel rendelkezik. A matematikai függvények és módszerek széles tárházát felöleli. Jelenleg is aktívan fejleszti az on-line közösség. Hátránya, hogy elsodlegesen a C ill. a FORTRAN nyelvet és a UNIX-szerű operációs rendszereket támogatja. Tipikus a GSL-ben, hogy mivel C fordító alatt is szeretnék fordítani, ezért a többféle skalár típust egy header fájl többszöri include-álásával érik el. [7]

Octave. Az Octave is a GNU projekt része és ez is GPL licensszel rendelkezik. A GSL-ben megtalálható numerikus rutinokat használja. Mivel a GSL nagy része az OCTAVE forrásnak is része, ezért az egyes algoritmusok több helyen is implementálva vannak, csakúgy, mint a GSL-ben. [8]

Numerical Recipes. A Numerical Recipes egy könyvsorozat és ezzel együtt algoritmusok halmazának gyujtoneve. Több programozási nyelvet támogatnak köztük a C-t és a C++-t is, de az algoritmusok licensze nem ingyenes, ezért nem használhatjuk fel a könyvtárunkban. [9]

IT++. Az IT++ lineáris algebrához, jelfeldolgozáshoz és kommunikációs rendszerek szimulációjához használható függvényeket és osztályokat nyújt. A szintaxisa hasonló a Matlab szintaxisához. Teljes funkcionalitásának kihasználásához külső könyvtárakra (pl. lineáris algebra, BLAS, Intel Core Math Library) van szüksége. Tetszett, hogy van benne Bessel függvény is, tehát a numerikus rutinok nem merülnek ki mátrix algebrában. [10]

Newmat. Egy ausztrál kutató írta a statisztikai kutatásaihoz. Késleltetett kiértékelést használ a sebesség növelésére, és külön kezeli a speciális eseteket (pl.: szimmetrikus mátrixok). A licenszében nincs megkötés. Nincs benne az általános sajátértékfeladat megoldása. Csak egy skalár típus használható, csak dinamikus mátrixokat ismer. [11]

OpenCV. Az OpenCV nem kifejezetten lineáris algebrai könyvtár, de tartalmaz egyszerű mátrix műveleteket. A beépített műveletek csak függvényhívásokkal érhetők el. Létezik C++ burkoló osztály OpenCV-hez (CvMat), de annak sincsenek operátorai, így nem felel meg a mi céljainknak. Nem kielégítően teljes a numerikus rutinok készlete sem, bár tisztán képfeldolgozáshoz elegendő. [12]

μ BLAS. A μ BLAS a BLAS (Basic Linear Algebra Subprograms) könyvtár C++-os kiegészítése. Támogatja a dinamikus és statikus tárolókat, valamint a range jellegű nézetek létrehozását. Sajnos nincsenek operátorok mátrix műveletekre, és csak a legemlibb numerikus rutinokat tartalmazza. Kedvezo a licensze. [13]

BLITZ++. A μ BLAS bevallottan örökölt a BLITZ++-tól, nem meglepő tehát, hogy a mi szempontjaink alapján teljesen hasonlóknak tunnek. Lehet több mint két indexű tenzorokat létrehozni. A honlap nagyon büszke rá, hogy statikus tárolók használatával olyan kód fordul, amit assembler nyelven is alig lehet gyorsítani. Hátrányai: nincs operatoros mátrixszorzás, se numerikus analízis. [14]

Vector Library (VL). Egyszemélyes fejlesztés, kevés hátránya van, ezek (fontosságai sorrendben): Nem kielégítő a numerikus analízis függvények halmaza. Dinamikus objektumok esetén feleslegesen konstruktorokat hív. Megfelelo numerikus függvények hiányában figyelmünk a JAMA könyvtárra, és annak hordozhatóvá tételére terelődött. [15]

TNT, JAMA. A JAMA az amerikai Nemzeti Szabványügyi és Technológiai intézet (NIST) fejlesztése, a Matlabbal együttműködve készítették ezt a lineáris algebrai függvénykönyvtárat. A JAMA a TNT (Template Numerical Toolkit) dinamikus tároló osztályát használja. A TNT-ben C++ operátorral hívhatóak az elemenkénti műveletek, és sajnos nem a mátrixműveletek. Csak dinamikus mátrixai vannak. Csak header fájlokból áll, így könnyebb fordítani. Szinte semmilyen korlátozás nincs a licenszben.

A JAMA a Matlab alapvető numerikus analízis defaktorizáló rutinjainak (Cholesky, LU, QR, SVD, Eig felbontások) a neve, melyet eredetileg JAVA nyelven tettek közzé. A NIST honlapon ezek C++ változatát találjuk, de csak annyit használ a C++ újdonságaiból, hogy megadható a skalár típusa, és egy és kétdimenziós dinamikus aritmetikai tároló kell a működéséhez, melyeknek kizárólag az elem-elérés funkcióját használja. A sajátértékfelbontásban mind a Wilkinson-féle véletlen lépés, mind egy Matlabos véletlen lépés benne van. Még permutációs mátrixokra is jól működik, konvergenciájának sebessége megközelítőleg azonos a Matlab bináris rutinjaiéval. [16]

2.3. Következtetések

Az vizsgálatok alapján a következőket állapíthatjuk meg: A TNT/JAMA kombináció kiemelkedik a többi közül a numerikus rutinok tekintetében (JAMA),

1. táblázat. A tanulmányozott könyvtárak tulajdonságai. ✓: megfelel a céljainknak, ~: használható, de nem teljes, -: nem felel meg a céljainknak az adott követelmény szempontjából, üres mező: nem minősítjük adott követelmény tekintetében

Könyvtárak	Szempontok									
	1	2	3	4	5	6	7	8	9	10
GSL, Octave	-	✓	-	~		-	~	-		
Numerical Recipes	-	-					✓			
IT++	✓	-					✓			
Newmat	✓	✓	-	✓	~	-	~		~	-
OpenCV	-	✓	-	~		~	~			~
μ BLAS	-	✓	✓	✓		✓	-	✓		-
BLITZ++	-	✓	✓	✓		✓	-	✓		~
VL Vector Library	✓	✓	✓	✓	-	✓	~	~	-	-
TNT/JAMA	-	✓	-	✓		✓	✓	-		-
TensorLib/CJAMA	✓	✓	✓	✓	✓	✓	✓	~	✓	~

viszont a tároló képességei messze nem a legjobbak a mezonyben. Mivel a JAMA függvények alig függenek az általuk használt tároló osztálytól, nyilvánvaló hogy az optimális megoldást a JAMA rutinok más tárolóval való kombinációja jelenti.

Ezek után csak az maradt a kérdés, hogy milyen, alapszervelet operátoros írását támogató tárolót használjunk. Szempontjaink alapján a vizsgált halmazból kiemelkedik a VL (Vector Library), de ez sem valósít meg minden, általunk hasznosnak és kívánatosnak ítélt megoldást, ami fellelhető a bemutatott gyűjteményben.

Két választás előtt álltunk: Vagy feljavítjuk a VL-t a hiányos funkcióval, vagy a meglévő statikus könyvtárunkat javítjuk fel az abból hiányzó funkciókkal, valamint olyan extrákkal, mint a multidimenziós tárolók és transzponálásuknak támogatása, mint kezdetleges tenzor implementáció. A döntést a tri- és quadrifokális tenzorok implementálásának igénye az utóbbi irányba vitte el. Ezek használatával tetszőleges összefüggő fotósorozat radiális torzítása meghatározható fényképenként [4].

2.4. Configurable JAMA

Az előző fejezetben tárgyalt szempontok szerint, és mivel gyakran felmerült konstans méretű mátrixokon dolgozó numerikus függvények beágyazott rendszerben való megvalósításának a kérdése a környezetünkben, elkészítettük a JAMA könyvtár konfigurálható változatát CJAMA néven. Preprocesszor direktívákkal lecserélhető a tárolók foglalása, felszabadítása, valamint méretük lekérdezése és átméretezésük. Így a CJAMA függvények fordíthatóak igen sokféle tároló objektum használatával. Akár statikus tárolók is használhatóak, bár a JAMA eredetileg úgy van strukturálva, hogy minimális a dinamikus memóriaműveletek miatti veszteség. (Foglalás, felszabadítás csak kilépéskor, belépéskor, vö. Matlab-Real Time Workshop). Az implementáció alatt vettünk észre pár kezeletlen potenciális

futásideju problémát, ezeket egyelőre kivételkezeléssel orvosoltuk, de a közeljövben a teljes javításukat is el fogjuk végezni.

2.5. TensorLib

A 'TensorLib'-ben egy új statikus és dinamikus tárolót valósítottunk meg, a magasabb szintu muveletekhez pedig felhasználtuk a CJAMA algoritmusait. A statikus tároló a stack-en, a dinamikus pedig a heap-en foglal helyet az elemeinek. Mindkét esetben osztálysablon paraméterként állítható a skalár típusa, statikus esetben plusz sablonparaméter(ek)ben adható meg az osztály mérete, dinamikus esetben az indexek száma.

A mátrixmuveletekhez felhasználtuk a rendelkezésre álló C++ operátorokat. Az inverz és a transzponálás jelölhető hatványozással is. Az elemenkénti muveletek is leírhatók operátoros formában. A statikus tároló muveleteit ciklus nélkül, kifejtve valósítottuk meg, ez assembly hatékonyságú kódra fordul optimalizálva. (MS Visual C++ 8.0).

A következő programrészlettel szeretnénk demonstrálni a TensorLib meglévő szolgáltatásait.

```
//SArray1D: statikus sorvektor
//SArray2D: statikus mátrix

SArray1D<float,3>   a3 = 1, b3=2; //feltöltés azonos elemekkel
SArray1D<float,6>   a6 = a3 | b3; //konkatenáció
//a _NONE_ azonosító bevezetésével tudtuk megoldani
SArray1D<float,2>   a4 = _NONE_ | 1.0f | 2.0f;
SArray2D<float,2,3> a2x3 = _NONE_ & a3 & b3;
SArray2D<float,3,2> a3x2 = a2x3 ^ _T_ ; // a2x2*(a2x2^_T_) zárójelezendo.

//DArray<typename,int N>: dinamikus N indexu tároló

DArray<float,2> B1 = a2x3;
DArray<float,2> B2(1,4);
DArray<float,2> B3(3,12);

TensorRef<float,2> ref1(a2x3); //kétindexu referencia,
TensorRef<float,2> ref3(B3);

range r1(2); //intervallumok
range r2(1,2);
range r3(0,3,11);

// Matlab A = B(:,2:3) //a range-indexelés opcionálisan kezdodhet 1-tol is
DArray<float,2> A2D = ref1(_ALL_, r2);

ref3(r1,r3) = B2; //Matlab B3(3,1:3:12) = B2

SArray2D<float,3,3> Q, R, A=_NONE_ & a3 & b3 & a3;
```

```

SJAMA::QR<float,3,3> qr_A(A);
qr_A.getQ(Q);qr_A.getR(R);

cout<<"A:"<<endl<<A; //kiírás
cout<<"Q:"<<endl<<Q;
cout<<"R:"<<endl<<R;
cout<<"Q*R:"<<endl<<Q*R;

DArray<float,2> AA=A, D, U;

DJAMA::Eigenvalue<float> eig_AA(AA);
eig_AA.getD(D);eig_AA.getV(U);

cout<<U*D-A*U; //ellenorzés

```

Az egyes mátrixok a `|` és a `&` operátorokkal egymás mellé és alá fuzhetok. A `_NONE_` üres mátrix mögé fuzve a skalárokat új mátrix hozható létre. A transzponálásra az előre definiált `_T_` operátor segítségével lehetséges. Az `range`-ek háromféleképp adhatók meg: egy paraméter esetén egy koordinátát jelöl, ketto esetén egy intervallumot, három esetén egy intervallumot, ahol a középső paraméter adja meg a lépésközt (a lépésköz lehet negatív is). Ezen kívül definiáltunk egy `_ALL_` `range`-t, amivel az egész oszlopot ki lehet jelölni.

A numerikus muveletek jelenleg osztályokon keresztül érhetok el. A pillanatnyilag támogatott muveletek: `det()`, `eig()`, `QR()`, `LU()`, `SVD()`, `Cholesky()`, `Solve()`, `Solve_LS`.

A `TensorLib` preprocesszor `define`-okon keresztül állítható, az egyes részei külön engedélyezhetok. A mátrixok kezdő indexelése beállítható nullára vagy egyre a `()` indexelés és a `'range'` esetében, a `||` indexelés mindig nullától indul.

A `TensorLib` könyvtárat Microsoft Visual C++ 8.0 és 9.0 alatt teszteltük.

Jelenleg mind a `gpCV++`, mind a `TensorLib` könyvtár kísérleti stádiumban van, olyan változások is elképzelhetőek, melyek miatt még korai lenne a forráskódot jelen állapotában közzétenni. A `CJAMA` (`ConfigurableJAMA`) változatot viszont mielőbb közzétesszük, amint javítjuk a kivétel-veszélyes részeket. Amennyiben valaki egyéb célokra kipróbált numerikus függvényeket szeretne minél előbb használni, annak a `VL` és a `CJAMA` összedolgozását ajánljuk, ha nem zavaró a dinamikus mátrixok konstruktorainak felesleges hívása.

2.6. Konklúzió

A cikkben a `gpCV++` nagysebességu képfeldolgozási programkönyvtár fejlesztésének legújabb állapotát mutattuk be, nevezetesen a C++ nyelven implementálható képfeldolgozási problémákhoz szükséges mátrixkönyvtárak kérdése került tárgyalásra. A fosodrású lineáris algebrai csomagokétól eltérő szempontjainkat pontokba rendeztük, és ezen szempontok alapján bemutattuk az általunk vizsgált szabad hozzáférésu könyvtárakat. Mivel ezek egyike sem egyesítette maradéktalanul a fellelhető kedvező megoldásokat, ezért eldöntöttük, hogy a kedvező megoldásokat beleépítjük a korábban használt statikus objektumokat kezelő

könyvtárunkba. A cikkben az elkészült funkciókat egy példaprogram segítségével mutattuk be.

Eljutottunk odáig, hogy mind a GPU-t hatékonyan használjuk előfeldolgozásra, mind a mátrixkönyvtárat a geometriai számítások elvégzésére. A közeljövő feladata, hogy olyan demonstratív alkalmazásokon (mint például a beltéri négy rotoros helikopter térbeli helyzetének meghatározása, képsorozat regisztrációja) finomítsuk a könyvtárakat, hogy azután elozetes verziót oszthassunk meg a képfeldolgozással foglalkozó közösséggel.

Köszönetnyilvánítás

A szerzők ezúton fejezik ki köszönetüket az OTKA K 71762 projekt által nyújtott támogatásért.

Hivatkozások

1. Prohászka, Z., Kerti, A.: Development of the GPU based gpCV++ Image Processing Library, IV. Magyar Számítógépes Grafika és Geometria Konferencia, Budapest (2007)
2. Kerti, A.: Nagysebességu képfeldolgozó környezet implementálása 3D grafikus kártyán, Diplomamunka, Budapesti Műszaki Egyetem, Budapest, (2008)
3. Bouguet, J.-Y.: Pyramidal implementation of the Lucas-Kanade feature tracker: Description of the algorithm, Technical Report 1999.2, Intel Research Laboratory. (1999)
4. Thirithala, S., Pollefeys, M.: Multi-view geometry of 1D radial cameras and its application to omnidirectional camera calibration, Tenth IEEE International Conference on Computer Vision, 17-21 Oct. 2005 pp. 1539 - 1546 Vol. 2 (2005)
5. Jack Dongarra összehasonlító táblázata <http://netlib.org/utk/people/JackDongarra/la-sw.html>
6. Wikipedia http://en.wikipedia.org/wiki/List_of_numerical_analysis_software
7. GSL honlapja <http://www.gnu.org/software/gsl/>
8. <http://www.gnu.org/software/octave/>
9. Numerical Recipes honlapja <http://www.nr.com/>
10. IT++ honlapja <http://itpp.sourceforge.net/>
11. Robert Davies honlapja www.robertnz.net
12. OpenCV könyvtár honlapja <http://sourceforge.net/projects/opencvlibrary/>
13. μ BLAS honlapja <http://www.boost.org/libs/numeric/ublas/doc/index.htm>
14. BLitz++ honlapja <http://www.oonumerics.org/blitz/>
15. Vector Library honlapja <http://www.cs.cmu.edu/~ajw/doc/vl.html>
16. TNT, JAMA <http://math.nist.gov/tnt/index.html>