Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar

Wavelet alapú előfeldolgozás deep direkt vizuális odometriánál

TDK DOLGOZAT

Készitette:
Dámsa Levente

Konzulensek:
Dr. Szegletes Luca
Dr. Blázovics László

2021. október 29.

Tartalomjegyzék

1. Bevezetés

Az autonóm robotok témaköre jelenleg egy aktívan kutatott tudományos terület, aminek egyik fő oka a szenzorok gyors fejlődésének köszönhető. Ahhoz, hogy különböző önvezető funkciókkal lássunk el egy autót, vagy egy drónt, elengedhetetlen a környezetet valamilyen módon feltérképezni. Egyik lehetséges módszer a szimultán lokalizáció és feltérképezés, vagy röviden SLAM (Simultaneous Localization and Mapping), melynek feladata a robot haladása során a különböző szenzoradatokból folyamatosan feltérképezni az ismeretlen területet, és ezen térképen elhelyezni a járművet. Ehhez számos szenzort alkalmaznak a szakirodalomban, ilyen például a Lidar szenzor, különböző kamerarendszerek (mono, sztereo), inerciális szenzorok, illetve GPS. A vizuális odometria feladata a robot poziciójának, és orientációjának meghatározása a kamera képeiből.

A konvolúciós neurális hálózatokat sikeresen alkalmazzák különböző számitógépes látórendszer feladatoknál, illetve robotikánál. Ilyen terület például az objektum detektálás, osztályozás, szemantikus szegmentáció. Ezen deep learning alapú algoritmusok felváltották a klasszikus módszereket, azonban a SLAM, és a vizuális odometria területén még nem történt meg az áttörés.

A szakirodalomban számos helyen alkalmaznak monokuláris kamerát, illetve különböző IMU, és GPS adatokat, ami jó kompromisszumot nyújt költségben és pontosságban. Az ilyen rendszerek fő kihívása a bejövő szenzoradatok hatékony fúziója, szinkronizálása, zajszűrése, illetve cél a gyors lefutás a limitált számítási kapacitás miatt.

Dolgozatomban áttekintést nyújtok ezen módszerekről, majd a wavelet transzformáció alkalmazhatóságát vizsgálom meg, amely a neurális hálózat bemenetét adja. A képeken végzett wavelet transzformáció segítségével egy tömörebb reprezentáció nyerhető, igy csökkentve a hálózat méretét. Az inerciális szenzoradatokon végzett wavelet transzformáció pedig segít a zajszűrésben, illetve az igy kapott scaleogram összefűzhető a kamera képével.

2. Visual odometry

A vizuális odometria feladata a robot poziciójának és orientációjának meghatározása kamerafelvételekből. Főbb alkalmazási területei a vezetést segítő rendszerekben, autonóm járművekben jelenik meg, Lidar szenzor esetén pontfelhő feltérképezéshez. A legfőbb nehézség a kamera alapú rendszereknél a megfelelő megvilágítás biztosítása, illetve szükség van képen jól elkülönülő objektumokra. Számos fajtája, és csoportosítása létezik, amit ebben a fejezetben fogok ismertetni.

2.1. Csoportositások

2.1.1. Kamera típus

A roboton elhelyezett hagyományos kamerák konstrukciója alapján beszélhetünk monokuláris(Mono VO) és sztereó(Stereo VO) rendszerekről. A monoku-

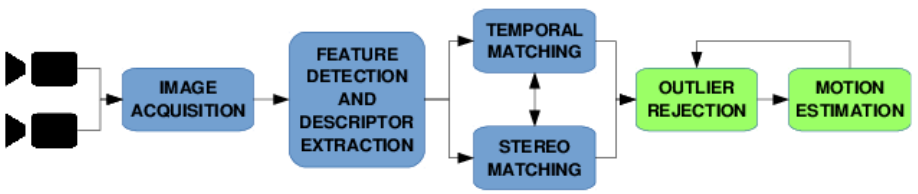
láris rendszerek esetén egy kamera áll rendelkezésre, amelynél nem lehet skálahelyes predikciót biztosítani, míg sztereó esetben közvetlenül megkapható a mélységi információ, de ez a távolságok növekedésével egyre jobban közelít a monokuláris esetre. Azonban bizonyos esetekben[1] nincs nagy pontosságbeli különbség a két rendszer között, igy a költségeket is figyelembe véve a monokuláris rendszer jobb választás lehet. Ezen kívűl megtalálhatóak Lidar szenzorok, Time of flight szenzorok, illetve RGB-Depth kamerák, amelyek mélységi információt tudnak nyújtani a környezetről. Ezek alkalmazása autóknál, nagyobb robotoknál jellemzőbb.

2.1.2. Direct és feature alapú

Egy másik lehetséges csoportositási rendszer a módszerek különbözőségén alapul, ami lehet direkt vagy feature-based. A feature based módszer lehet a bejövő képekből kinyert jellemző pontok, és ezen pontokat követő rendszer. A direct módszer közvetlenül a képet használja, mint bemenet, jellemzően valamilyen machine learning módszerrel.

2.2. Általános felépités

A hagyományos vizuális odometria 6 fő feladatra bontható fel [2]. Elsőként a kép kinyerése a kamerából, majd valamilyen képfeldolgozási módszer(torzitás csökkentése, szűrés). Ezután a feature detektálás, ami a képek közötti korrelációt, azonos pontok megkeresését jelenti. Ezen fázis végén valamilyen optikai flowfield-et kapunk. A flowfield korrekciója után a kamera mozgását kell megbecsülni ezek alapján, amire számos lehetőség létezik, például a Kálmán filter, vagy a Particle Filter.



1. ábra. Általános vizuális odometria architektúra [3]

3. Wavelet transzformáció

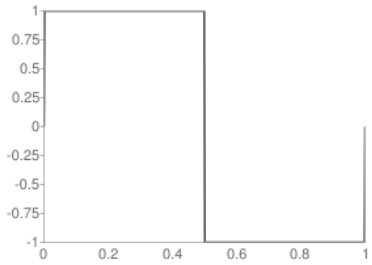
3.1. Bevezetés

A wavelet transzformáció egy olyan $L^2(R) \rightarrow L^2(R^2)$ leképezés, amely a Short Term Fourier Transzformációval(STFT) ellentétben jobb idő-frekenvia lokalizációt nyújt. A wavelet transzformációnak két fő csoportját különböztetjük meg, a folytonos idejű(CWT) és a diszkrét idejű(DWT) wavelet transzformációt.

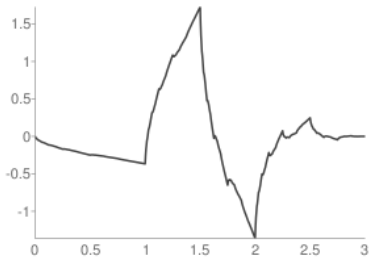
A wavelet egy hullámszerű oszcilláció, ami időben lokalizált, amelynek két fontos paramétere van: skála(scale) és localization(időtartományban értve).

3.2. Mother wavelet

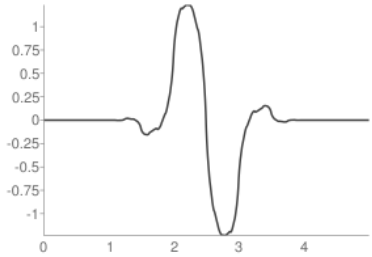
Fontos jellemző még az úgynevezett mother wavelet ($\Psi(t)$), amelyet a transzformáció során skálázunk, és időben eltolunk. Számos motherwavelet verzió létezik, mindegyik különböző tulajdonságokkal, egy adott alkalmazáshoz nincs egzakt módszer a motherwavelet kiválasztásához. A dolgozatom során használt pywavelets könyvtár számos motherwavelettel rendelkezik, ahol a tulajdonságaikat át lehet tekinteni. Az alábbi ábrákon bemutatok pár lehetséges mother waveletet:



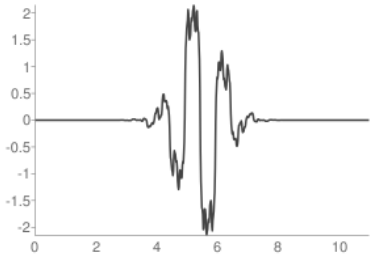
(a) Haar wavelet



(b) db2 wavelet



(a) bior1.3 wavelet



(b) bior3.5 wavelet

2. ábra. Motherwaveletek

3.3. Continous wavelet transform

A wavelet kifejezése adott skála(a) és eltolás(b) esetén az alábbi képlettel irható fel:

$$\Psi_{ab}(t) = \frac{1}{\sqrt{a}}\Psi(\frac{t-b}{a}) \tag{1}$$

A folytonos idejű wavelet transzformáció az f függvényen az alábbi képlettel adható meg:

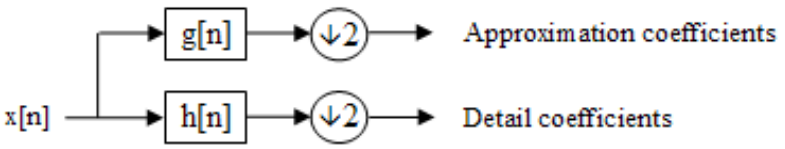
$$W_f(a,b) = \int_{-\infty}^{\infty} \Psi_{ab}(t) f(t) dt \tag{2}$$

3.4. Discrete wavelet transform

A folytonos wavelet transzformáció minden lehetséges waveletet felhasznál, vagyis mind a két paramétere folytonos, igy számuk végtelen. Ezt a gyakorlatban nehézkes alkalmazni, ezért általában a diszkrét wavelet transzformációt szokták használni, ami véges elemszámú wavelettel rendelkezik:

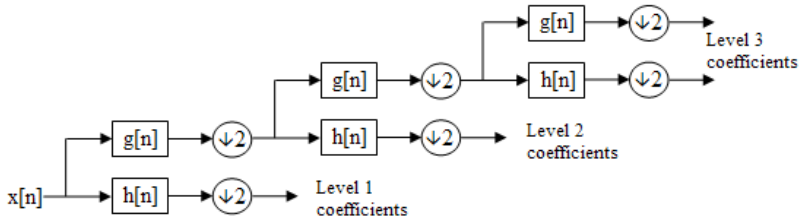
$$W_{m,n} = \int_{-\infty}^{\infty} f(t) \Psi_{m,n}(t) dt \tag{3}$$

A diszkrét wavelet transzformációt lehetséges úgynevezett filterbankokkal elvégezni, ami kisebb számitási komplexitással rendelkezik, mint a Fast Fourier Transzformáció. Egy adott bejövő jelet minden lépésben két részre bontunk, egy aluláteresztő szűrővel megkapjuk a jel átlagát (approximation), és egy felüláteresztőszűrővel a jelhez tartozó részleteket (details), melyet 2 hatványai szerint csökkentve újramintavételezünk(downsampling).



3. ábra. Filter bank egy szinten[4]

Ezen filter kaszkádositását pedig az alulátáresztű szűrő kimenetének további felbontása eredményezi:



4. ábra. Több szintű felbontás filter bankokkal[4]

3.5. Alkalmazási területek

A Wavelet transzformációt számos helyen alkalmazzák, a fő területek a zajszűrés, tömörités, idősorok elemzése, képek, videók vizjelezése, agyi jelek (EEG, ECG) analízise, betegségek szűrése.[5]

4. Deep learning

4.1. Bevezetés

A mély neurális hálók (Deep learning) a gépi tanulás egyik alfaja, amelyben beszélhetünk felügyelt, ilettve felügyelet nélküli tanulásról[6].

Különböző deep learning architektúrákat(konvolúciós neurális hálózatok, rekurrens neurális hálózatok) számos területen alkalmaznak, köztük számítógépes látásnál, szövegértésnél, természetes nyelvfeldolgoznásnál, fordításnál, bioinformatikánál.

Az alábbi fejezetben ismeretetem a Deep learning területét, melyben először bemutatom a gépi tanulást, majd az elemi perceptron ötletétől áttekintem a manapság gyakran elterjedt komplex neurális hálózatokat.

4.2. Kezdetek: Gépi tanulás

A gépi tanulás a mesterséges intelligencia egyik alfaja [7]. Célja a gép adott feladatának az elvégzését javítani, az adatokban összefüggéseket keresni, általánositani, igy számára ismeretlen bemenetre is képes válaszolni.

A gépi tanulásnak két alosztálya van: felügyelt és felügyelet néküli tanulás. A fő különbség a bemenő adatokban rejlik. Felügyelt tanulás esetén az adatok címkézettek, vagyis minden bemenethez tartozik egy elvárt kimenet is. Ellenben felügyelet nélküli tanulásnál az adatok címkézetlenek. Így a felügyelt tanulás esetén az ügynök a bemenő adat struktúrájára tanul rá.

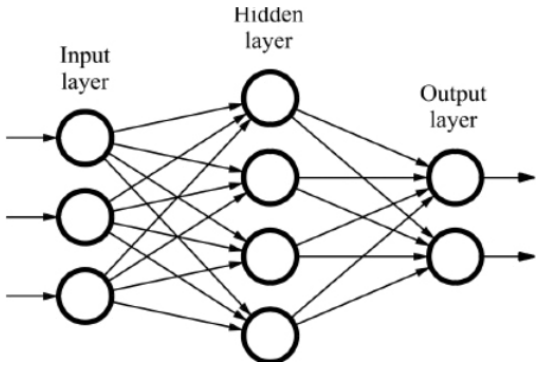
4.3. Deep learning

A mélytanulás(Deep learning) a gépi tanulásnak egy részhalmaza, illetve tekinthetünk úgyis rá mint egy továbbfejlődése. A deep learning neuronhálózatok alkalmazásán alapszik, azokon a statisztikai modelleken, amelyeket 1943-ban Warren McCulloch és Walter Pitts alkotott meg a biológiai ideghálózatok mintájára. Egymásba ágyazott neuronhálózatokat először 2006-ban használtak mélytanuló rendszerként és a modell matematikája 2012-ig csiszolódott, amikor a GPU-k kapacitásának kihasználásával igazi robbanás történt a deep learning alkalmazásában.

Az alábbi fejezetben ismertetem a neurális hálózatok felépitését, és működési elvét, majd a vizuális odometriához releváns ismert architektúrákat mutatom be.

4.3.1. Neurális hálózatok

Mesterséges neurális hálók, melyet gyakran csak neurális hálónak neveznek egy olyan számitási gráf, amelyet a biológiai neurális hálózatról modelleztek [8].



5. ábra. Egy általános neurális hálózat felépitése[9]

Egy neurális hálónak a csúcspontjai az úgynevezett elemi neuronok, melyek a biológiai neuronokról lettek modellezve. Gyakran ezeket az elemi neuronokat oszlopokba vannak rendezve, ahol egymás között nincs kapcsolat közöttük.Ezeket a neurális hálózat rétegeinek nevezünk. A hálózat bemenetén található réteget input layernek , a kimenetét output layernek nevezzük. Az input és output layerek között tetszőleges számú úgynevezett rejtett réteg helyezkedik el.

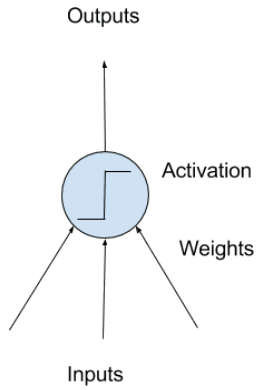
4.3.2. Elemi perceptron

Az elemi perceptron, vagy elemi neutron tartalmaz tetszőleges számú bemenetet, és egy kimenetet. Minden bemenethez tartozik egy súly érték illetve egy b bias érték. Ahhoz, hogy a neurális hálózat ne egy egyszerű lineáris hálózat legyen, szükség van egy nemlineáris függvényre, ilyen például a sigmoid függvény.

Ennek a struktúrának a matematikai leirása a következő:

$$y_k = \phi * (\sum_{j=0}^{m} w_k j x_j), \tag{4}$$

ahol ϕ a nemlineáris függvény, w_{kj} az x_j-hez tartozó súly.



6. ábra. Elemi neuron felépítése[10]

4.3.3. Multi layer perceptron

A legegyszerűbb neurális hálózat az úgynevezett multi layer perceptron, amely egy olyan előrecsatolt neurális hálózat, amely tartalmaz egy input réteget, egy output réteget, és közte legalább egy rejtett réteget. Az MLP egy megerősitéses tanuláson alapú módszerrel tanul, a backpropagationnel. Ez a hálózat képes egy lineárisan nem szeparálható adathalmazon tanulni, ellenben a lineáris perceptronnal.

4.3.4. Backpropagation

A backpropagation algoritmus a 70-es években lett kifejlesztve[11], egy általános optimalizációs módszerként, ahol automatikus differenciálásra alkalmazták egymásba ágyazott függvényeken. Elterjédésére, illetve népszerűségének kezdete azonban csak 1986-ban kezdődött Rumelhart Hinton publikációjával, ami után a machine learning világ felfigyelt rá.

A backpropagation algoritmushoz[12] szükség van egy adathalmazra, ami bemenet,kimenet párokat tartalmaz (x_i, y_i). Legyen X egy ilyen párokból álló N elemű halmaz:

$$X = (x_1, y_1), ..., (x_N, yN) \tag{5}$$

Ezen kivűl adott, az imént bemutatott előrecsatolt neurális hálózat. Az állitható paramétereket, vagyis a súlyokat jelöljük θ -val. Továbbá tegyük fel, hogy a különböző rétegekek között minden kapcsolat megvan, vagyis a fully connected layerekről beszélhetünk. Jelölje $w_i j^k$ az k-adik réteg i-edik és k-1-edik

réteg j-edik neuronjához tartozó súly. b_i^k jelölje a k-adik réteg i-edik neuronjához tartozó bias-t. A rétegek között továbbra sincs kapcsolat.

A hibafüggvény(error function), A neurális hálózat kimenete és az eredeti bemenet közötti eltérést fejezi ki. Ilyen lehet például a négyzetes eltérés, vagyis az l2-es norma. Jelölje $\hat{y_i}$ a hálózat alapján kiszámitott értéket:

$$E(X, \theta) = \frac{1}{2N} \sum_{i=1}^{N} (\hat{y_i} - y_i)^2 \tag{6}$$

Ennek a loss functionnek vesszük a parciális deriváltját minden súlyra:

$$\frac{\partial E(X, \theta)}{\partial w_{ij}^k} = \frac{1}{2N} \sum_{d=1}^{N} \frac{\partial}{\partial w_{ij}^k} (\frac{1}{2} (\hat{y_d} - y_d)^2) = \frac{1}{N} \sum_{d=1} N \frac{\partial E_d}{\partial w_{ij}^k} \tag{7}$$

A backpropagation algoritmus a lánc szabály alkalmazásával történik:

$$\frac{\partial E}{\partial w_{ij}^k} = \frac{\partial E}{\partial a_j^k} \frac{\partial a_j^k}{\partial w_{ij}^k} \tag{8}$$

ahol a_{ij}^k az aktivációs függvény a nemlinearitás előtt. Ez a kifejezés azt jelenti, hogy a hiba változása a súly függvényében egyenlő a hiba változása az aktivációs függvény, és az aktivációs függvény változása a súly függvényének szorzatával.

Az első tényezőt általában hibának nevezzük, amelynek jelölése:

$$\delta_j^k = \frac{\partial E}{\partial a_j^k} \tag{9}$$

A második tényezőt felirhatjuk az aktivációs függvény definiciójából:

$$\frac{\partial a_j^k}{\partial w_{ij}^k} = \frac{\partial}{\partial w_{ij}^k} (\sum_{l=0}^{r_k - 1} w_{lj}^k * o_l^{k-1}) = o_i^{k-1} \tag{10}$$

Ezzel a parciális derivált felirható az alábbi formában:

$$\frac{\partial E}{\partial w_{ij}^k} = \delta_j^k o_i^{k-1} \tag{11}$$

vagyis a parciális deriváltja egy súlynak a k-adik rétig j-edik eleme, és az előző réteg i-edik kimenete.

A számitásokat a kimeneti réteggel kezdi az algoritmus, vagyis kiszámitjuk δ_1^m-et, ahol m az utolsó réteg.

$$E = \frac{1}{2} (\hat{y} - y)^2 = \frac{1}{2} (g_o(a_1^m) - y)^2), \tag{12}$$

ahol g(x) az aktivációs függvény a kimeneti rétegen.

A parciális derivált alkalmazása a lánc szabály felhasználásával:

$$\delta_1^m = (g_o(a_1^m) - y) g_o'(a_1^m) = (\hat{y} - y) g_o'(a_1^m). \tag{13}$$

E meghatározása:

$$\frac{\partial E}{\partial w_{ij}^k} = \delta_j^k o_i^{k-1} = (\hat{y} - y) g_o'(a_1^m) o_i^{m-1} \tag{14}$$

A rejtett rétegeknél is alkalmazható a láncszabály. Az error term kiszámitása:

$$\delta_j^k = \frac{\partial E}{\partial a_j^k} = \sum_{l=1}^{r^{k+1}} \frac{\partial E}{\partial a_l^{k+1}} \frac{\partial a_l^{k+1}}{\partial a_j^k} \tag{15}$$

ahol az összegzés a k-adik réteg összes elemén megy végig. A bias értékét a nulladik elemnek vettük, amelynek értéke nem függ az előző layerek értékétől.

A következő réteg hibatényezőjét belevíve az előző egyenletbe, az alábbi kifejezést kapjuk:

$$\delta_j^k = \sum_{l=1}^{r^{k+1}} \delta_l^{k+1} \frac{\partial a_l^{k+1}}{\partial a_j^k} \tag{16}$$

majd felhasználva a_l^{k+1} definicióját:

$$a_l^{k+1} = \sum_{j=1} r^k w_{jl}^{k+1} g(a_j^k) \tag{17}$$

$$\frac{\partial a_l^{k+1}}{\partial a_j^k} = w_{jl}^{k+1} g'(a_j^k) \tag{18}$$

Ezt behelyettesitve δ_j^{k+1} kifejezésébe:

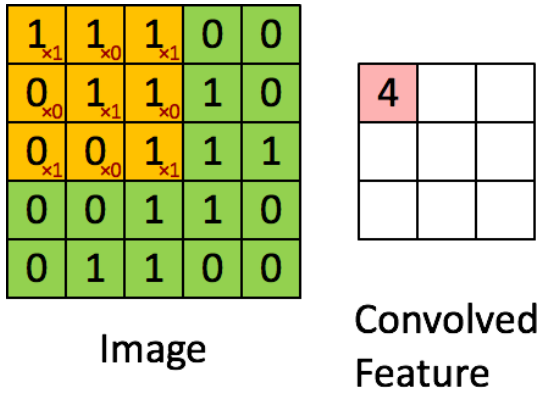
$$\delta_j^k = \sum_{l=1}^{r^{k+1}} \delta_l^{k+1} w_{jl}^{k+1} g'(a_j^k) = g'(a_j^k) \sum_{l=1}^{r^{k+1}} w_{jl}^{k+1} \delta_l^{k+1} \tag{19}$$

Ezek alapján az E parciális deriváltja egy súly alapján a következő képlet alapján adható meg:

$$\frac{\partial E}{\partial w_{ij}^k} = g'(a_j^k) o_i^{k-1} \sum_{l=1}^{r^{k+1}} \delta_l^{k+1} w_{jl}^{k+1} \tag{20}$$

A tanitás utolsó lépéseként frissiteni kell a súlyokat. Ezt egy α paraméterrel szoktuk jelölni:

$$\delta w_{ij}^k = -\alpha \frac{\partial E(X, \theta)}{\partial w_{ij}^k} \tag{21}$$



7. ábra. Konvolúció művelete[14]

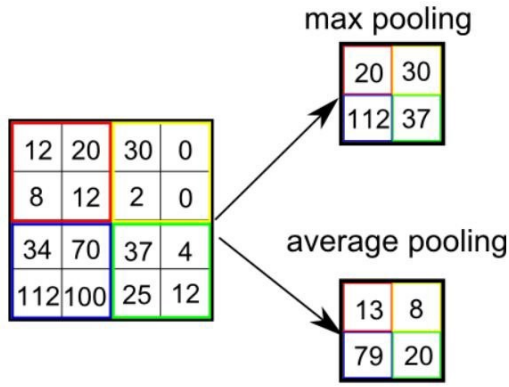
4.4. Elterjedt architektúrák

4.4.1. Konvolúciós neurális hálók

A konvolúciós neurális hálózatok, vagy röviden CNN, lehetővé teszik a számítógépes rendszerek számára az emberekhez hasonló látást, amelynek segítségével el lehet végezni olyan feladatokat, mint képen, vagy videón való objektumfelismerés, osztályozás, nyelvfeldolgozás, és hasonlók[13].

A konvolúciós neurális hálózatok bemenete általában egy kép, amelyeken a hagyományos módszerekhez képest kevesebb előfeldolgozási lépés szükséges. A konvolúciós neurális hálózat alapját a konvolúciós rétegek adják, amelyek lehetővé teszik a képekből magasabb szintű tulajdonságok kinyerését, illetve lényegesen kevesebb paraméterrel rendelkeznek, mint egy hagyományos MLP hálózat. A konvolúciós rétegek úgynevezett kernelekből állnak, amelyeknek a súlyai lesznek a hálózat paraméterei. Hiperparamétere ennek a kernelnek a mérete, a padding, ami a képet kiegészítő kitöltésért felel, általában 0 értékekkel. Ez lehetővé teszi, hogy a kép szélén található elemek ugyanannyiszor szerepeljenek, mint a belső elemek. Egy másik hiperparaméter a stride, vagyis a szűrő mozgatásának a léptéke. Ha elértük a kép végét, a szűrő a következő sorra ugrik. Ezeknek a rétegeknek beszélhetünk a mélységéről is, ami azt jelenti hogy szűrőt tartalmazzon a réteg.

A konvolúciós rétegek mellett megtalálhatóak úgynevezett Pooling layerek, amelyeknek a feladata az, hogy a konvolúciós réteg méretét redukálják. Itt is megadható egy ablak, amin belül valamilyen technikával, például a maximális érték kiválasztásával egy értéket tárolunk el. Ezekkel a rétegekkel a zajt is csökkenteni tudjuk.

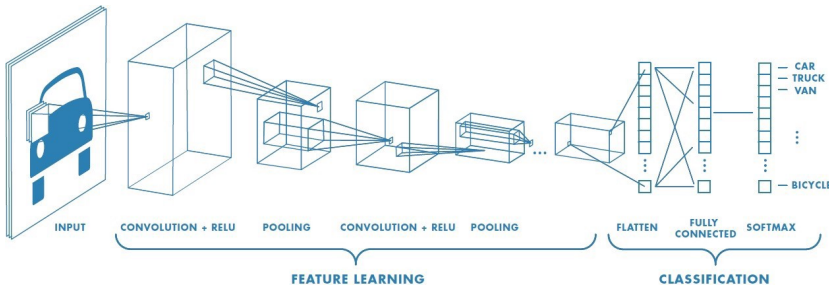


8. ábra. Maxpooling[14]

Tehát a konvolúciós neurális háló feladata különböző tulajdonságok kinyerése a képből, például élek detektálása. Ha ezekből a rétegekből többet egymás után rakunk, akkor magasabb szintű tulajdonságokat tudunk kinyerni, például emberi arc detektálása.

Végül megtalálható valahány fully connected layer a hálózat kimenetén, ami klasszifikáció esetén a különböző osztályok valószínűségét tárolhatja.

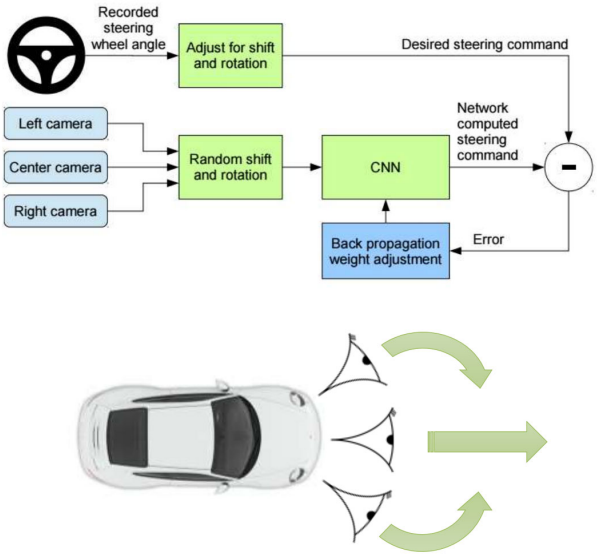
Egy általános konvolúciós neurális hálózat architektúrája az alábbi ábrán látható:



9. ábra. Konvolúciós neurális háló architektúra[14]

Az elmúlt évek során jelentősen javult a képek osztályozásának a pontossága a különböző új architektúrák miatt[13]. Ezen kívűl megfigyelhető egy olyan tendencia is, hogy egyre kisebb hálózattal voltak képesek azonos, vagy jobb eredményt elérni különböző osztályozó feladatokon. Pár ismert architekúra a LeNet[15], AlexNet[16], VGGNet[17], GoogleLeNet[18], és ResNet[19].

Autonóm járműveknél gyakran használnak konvolúciós neurális hálózatot, például az úgynevezett End2End [20] driving esetén a konvolúciós neurális hálózat kimenetén a beavatkozó jelek szerepelnek, bemenete pedig a kamera képe.

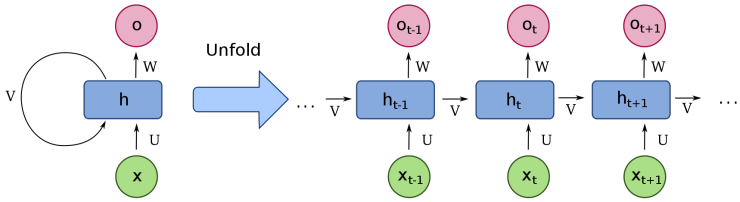


10. ábra. Nvidia E2E driving architektúra

4.4.2. RNN, LSTM

Az eddig ismertetett architekúrákat úgynevezett feedforward neural network-öknek nevezzük, aminek az egyik tulajdonsága az, hogy nincsen belső állapota, memóriamentes. A Rekurrens neurális hálózatoknál(RNN) már van belső állapot, ami lehetővé teszi olyan feladatok elvégzéséhez, mint a szövegfelismerés, vagy folyásírás felismerés.

Az RNN[21]-ek ábrázolásánál úgynevezett fold-al kiterítik időben ezt a belső állapotot, ahol a rekurrenciát így megszüntetjük, így könnyebb tárgyalásmódhoz jutunk.



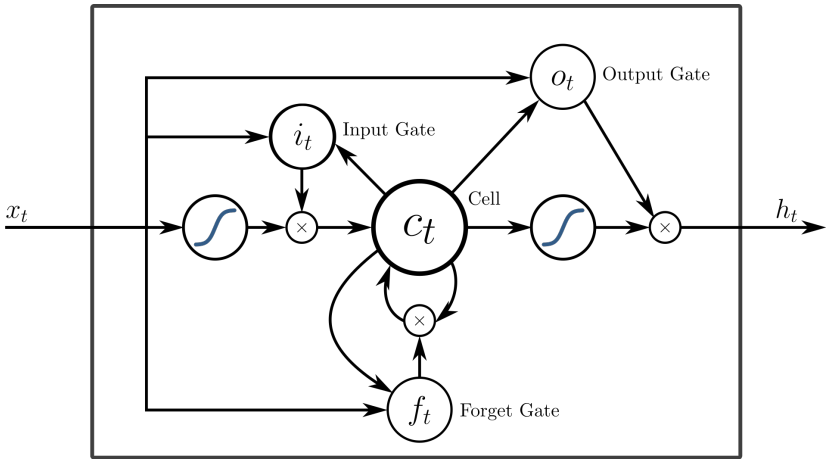
11. ábra. RNN[22]

Az aktuális állapothoz tartozó függvény ezek alapján:

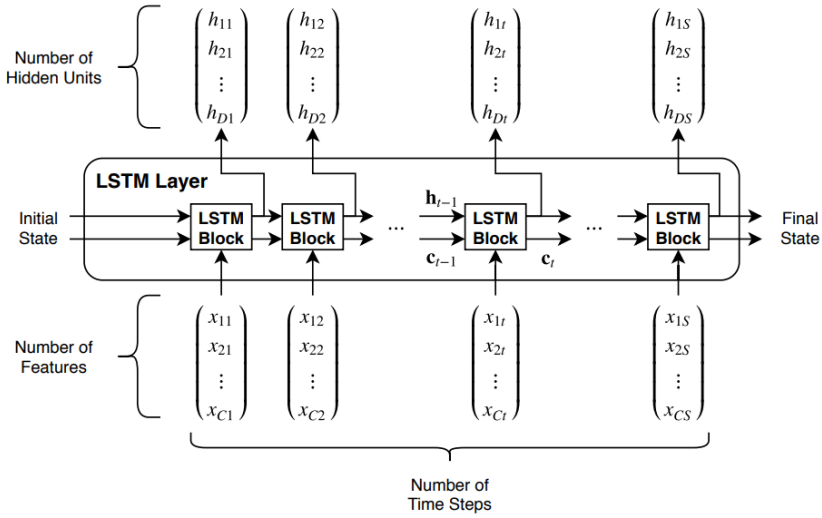
$$h_t = f(h_{t-1}, x_t) \tag{22}$$

A rnn tehát képes idősorok elemzésére, mivel tartalmaz információt az előző bemenetekről, illetve az imént bemutott konvolúciós neurális hálózatoknál is alkalmazzák, ilyen például a ResNet. Hátránya viszont, hogy nehéz a tanítása, illetve viszonylag kis szekvenciával működik jól tanh vagy relu aktivációs függvény esetén.

Az RNN továbbfejlesztésének tekinthető az úgynevezett Long Short Term Memory(LSTM), ami lehetővé teszi a múltbeli értékek hatékonyabb eltárolását, illetve megoldja az rnn-ek érzékenységét az eltünő gradiensekre.



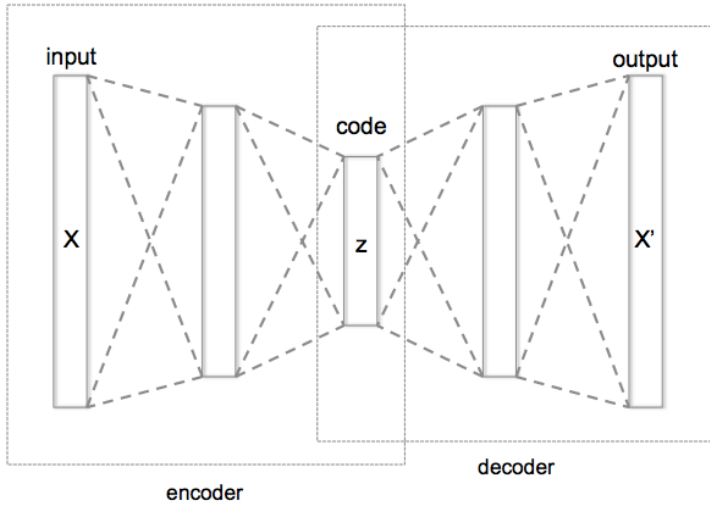
12. ábra. LSTM cella felépítése[22]



13. ábra. Lstm layer felépítése[23]

4.4.3. Autóenkóderek

Az autóenkóder egy viszonylag egyszerű neurális hálózat. Az összes variánsáról elmondható, hogy egyfajta tömörítés a feladata. A hagyományos módszerek közül a Principal Component Analysis(PCA)-hoz áll legközelebb. A hálózat középső rétege, amit kódnak nevezünk, felel meg a főkomponenseknek PCA elemzésnél. Ahogyan az RNN-nél is láthattuk, itt is szerepelhetnek Convolúciós rétegek az architektúrában.



14. ábra. Autóenkóder általános felépítése[24]

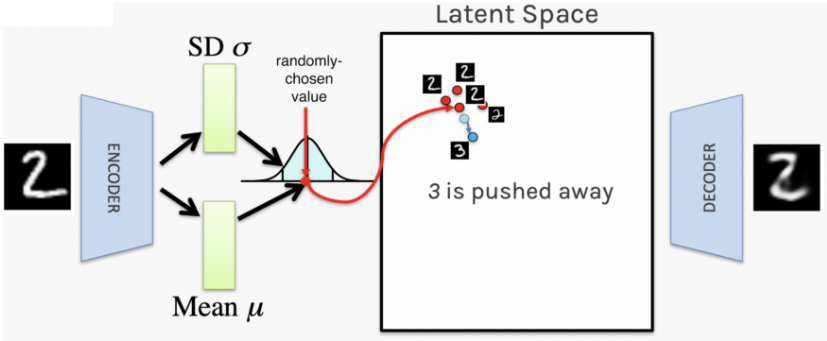
Az enkóder matematikai leírása a következő:

$$\begin{aligned} \phi &= \mathcal{X} -> \mathcal{F} \\ \psi &: \mathcal{F} -> \mathcal{X} \\ \phi, \psi &= \underset{\phi,\psi}{argmin} ||X - (\psi o \phi) X||^2 \end{aligned} \tag{23}$$

Tehát az enkóder függvény feladata az eredeti bemenetet leképeznie a látens térbe, ami egy tömörebb reprezentáció, mint az eredeti. A dekóder feladata pedig helyreállitani a látens térből az eredeti bemenetet. Ebben az esetben a bemenet és kimenet azonos. Egyik lehetséges felhasználása az autóenkódereknek az úgynevezett Denoising Autóenkóder, aminek a bemenetét mesterségesen zajjal terheljük, kimenete pedig az eredeti kép lesz.

A Variational Autoencoder, vagy röviden VAE, esetén a hagyományos autóenkóderrel szemben egy adat generálásra alkalmas eloszlást tanul meg. Ez azt jelenti, hogy a látens térben a hasonló elemek közel vannak egymáshoz.

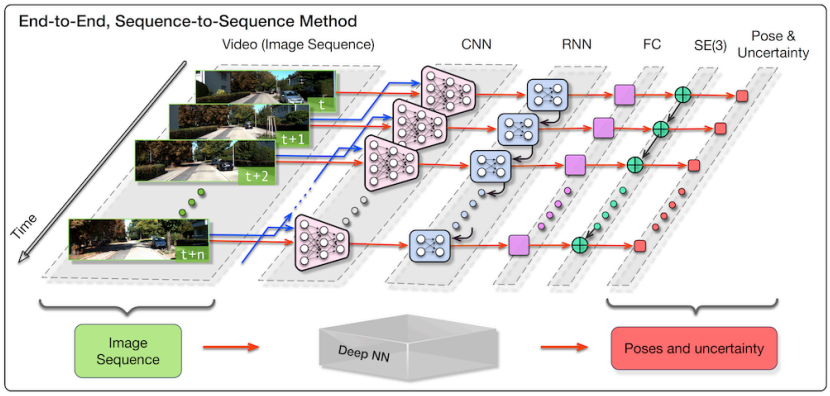
A VAE középső rétegénél már nem egy code, hanem egy átlag, és szórás szerepel, ami alapján generálható a kép.



15. ábra. VAE[24]

5. Deep learning Vizuális Odometriánál

Ahogyan a számítógépes látás más területein, mint például az objektum detektálás, szegmentálás, annotálás, képjavítás, felskálázás, itt is kezdenek megjelenni olyan direct módszerek, amelyek felváltják a hagyományos módszereket. Ilyen tendenciát láthatunk az önvezető autók, vagy a természetes nyelvfeldolgozás területén, ahol az End2End módszerek egyre nagyobb népszerűségnek örvendenek, hátrahagyva a hagyományos algoritmusokat. A VO és VSLAM területen azonban még nem történt meg az áttörés, illetve robotika területén a számitási kapacitás is limitált. Azonban kezdtek megjelenni olyan E2E megoldások, amik hasonló eredményeket érnek el a hagyományos módszerekkel. Dolgozatom alapjául a DeepVO algoritmust használtam fel, ami egy konvolúciós háló és egy RNN hálót kombinál össze.



16. ábra. DeepVO architektúra[25]

Az alábbi fejezetben áttekintem a deep learning alapú módszereket, amelyeket monokuláris vizuális slam algoritmusoknál használnak.

5.1. Módszerek áttekintése

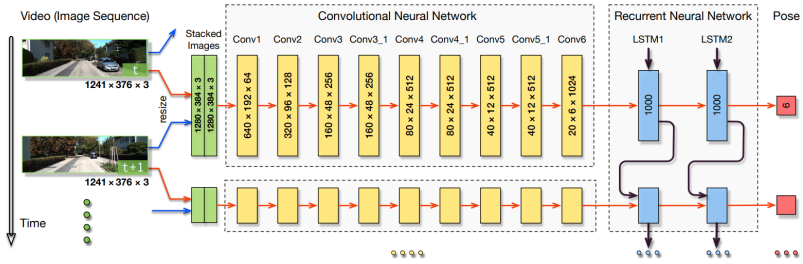
A supervised módszerek esetén rendelkezésre áll valamilyen formában egy ground truth adat, a KITTI adathalmaznál rendelkezésre áll az aktuális póz ground truth, de számos helyen elegendő egy sztereó, vagy depth map az eredeti pozició előállitására. A DeepVO [26] az elsők között volt az E2E módszerek között a vizuális módszereknél, amely habár nem haladta meg a State of the art-nak tekintett ORB-SLAM [27] architecturát, azonban ezzel a módszerrel lényegesen kevesebb feature engineering szükséges, illetve adott feladathoz, adathalmazhoz sem szükséges manuális adaptálás.

A WaveletMonoDepth [28] algoritmus egy depth map-et prediktál, amelyben a Wavelet transzformációt a konvolucios rétegek között végzik el, amelyet az inverz wavelet transzformációval állitanak vissza, ezzel megkerülve a Pooling layerek szüksegességét.Igy konvolúciós rétegek a wavelet együtthatókat próbálják becsülni.

Ezen kivűl létezik számos unsupervised learning módszer is, amik általában egy depth és egy pose netből állnak [29] [30]. Általánosságban elmondható, hogy egy sztereó kamerarendszerből áll össze a depth map és a pose ground truth adat.

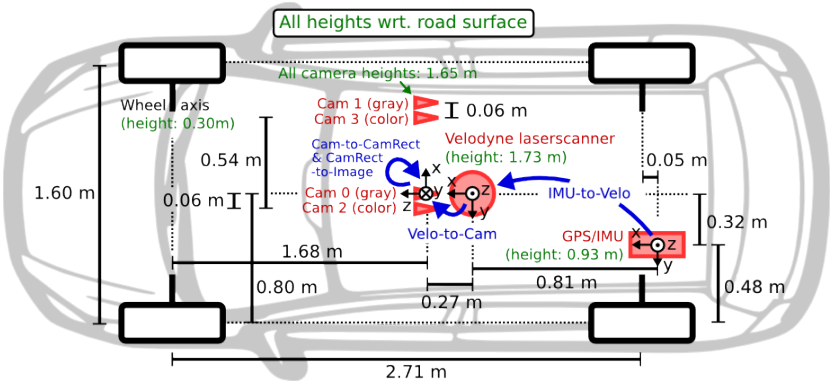
6. Wavelet transzformáció vizuális odometriánál

Dolgozatomban a DeepVO architectúrát vettem alapul. A hálózat architectúrája az alábbi ábrán látható.



17. ábra. DeepVO architektúra időben kiterített képe[25]: A bemeneti képpárokból alkotott bemenet, CNN hálózat, RNN hálózat, kimenetén az xyz irányú elmozdulás és a forgatás euler szögekben

A DeepVO bemenete egy hagyományos RGB képpár, ahol a csatornák mentén van összefűzve a kép. Ezen bemenetre illeszkedik egy CNN hálózat pár réteggel. Ezt követ egy RNN hálózat, ami két LSTM rétegből áll, melynek kimenetén előáll a 6 elemű kimenet, a transzláció és rotáció euler szögek formájában. A tanitást a KITTI adathalmazon végezték el, ezért következő lépésként ezt vizsgáltam meg. A KITTI adathalmaz egy önvezető autonóm platformhoz készült, melynek célja különböző számitógépes látórendszer feladatokhoz egy benchmarkot nyújtani. Ilyen feladatok például az optical flow becslés, vizuális odometria, 3d objektum detektálás és követés. Az autón megtalálható 3 kamera, melynek pozicióit az alábbi ábra mutatja. Az adatok a GPS/IMU adatok kivételével 10Hz-es mintavétellel lettek felvéve.



18. ábra. KITTI adatgyűjtő rendszere [31]

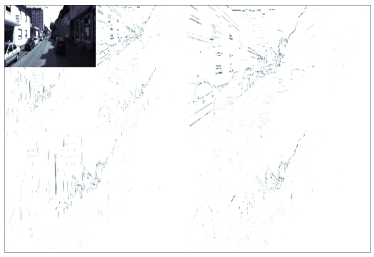
A DeepVO során a nyers, szinkronizált RGB képekre volt szükség, illetve a eredeti póz adatokra.

6.1. Wavelet tipus kiválasztása

A mother wavelet, illetve a dekompoziciós szint megválasztására nincs exact módszer, viszont vannak általános megfontolások, amik alapján le lehet szűkiteni a potenciálisan jól használható paramétereket.

Feature extractionhoz olyan waveletet érdemes választani, amely kicsi tartóval rendelkezik, ilyen például a haar, db2, vagy a sym2. A kis tartó segitségével könnyen elválaszthatóak a fizikailag közel lévő objektumok.

Az alábbi két ábra szemlélteti a Daubechies wavelet db2 és db10-es közötti különbséget. Az elnevezés dbn a waveletben megtalálható ”vanishing moment” számát adja meg, ami egyben a tartó hosszát is meghatározza. Az eredeti kép bal alsó sarkában található autón jól látszik, hogy db10 esetén lényegesen ”halványabb” képet kapunk, vagyis kevesebb együttható szerepel benne.



(a) db2 dekompozíció

(b) db10 dekompozíció

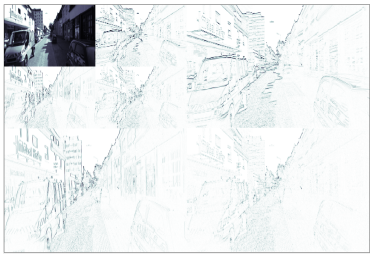
19. ábra. Különböző méretű tartók összehasonlítása: A db2-es ábra több információt szolgáltat

6.2. Wavelet szint kiválasztása

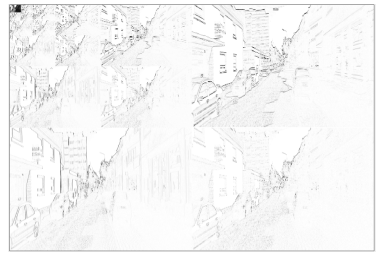
Emellett fontos paraméter a wavelet dekompoziciós szint. Mivel az együtthatók egy konvoluciós neurális hálózatra bemenetét adják, igy túl nagy szint választása esetén az alsó dekompoziciós együtthatók dimenziója túl kicsik lesznek ahhoz, hogy érdemben tudnának hasznos információt nyújtani.

6.3. Bemenet előállítása

A Wavelet transzformációt színcsatornánként elvégezve, az egyes együtthatókat egy 2 dimenziós tömbbé alakitottam, aminek a mérete megegyezik az eredeti képpel. Az eredeti képet összehasonlítva az igy kapott képen jól látszik, hogy egy lényegesen jobb reprezentációt kaptunk. Az elmozdulást különböző dekompoziciós szinteken, függőleges, vizszintes és diagonális felbontásban is megkapjuk, amelyet a konvoluciós szűrő több helyen tud detektálni, mint az eredeti kép



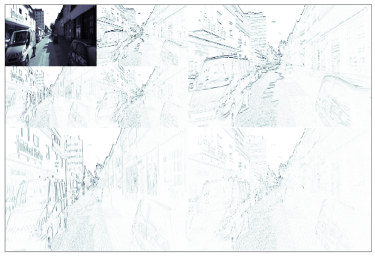
(a) Haar wavelet 2 szint



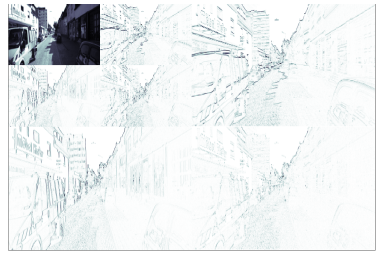
(b) Haar wavelet 10 szint

20. ábra. Wavelet dekompozíciós szintek összehasonlítása

esetén. Ezek mellett a bemenet lényegesen ritkább reprezentáció (sparse), ami azt jelenti hogy kevés 0-tól különböző érték van.



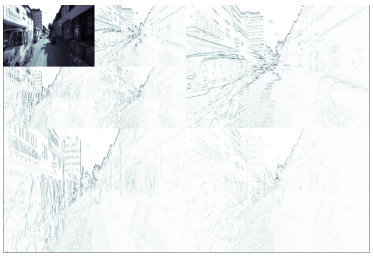
(a) t-1 időpontban vett kép Haar wavelet dekompozíciója



(b) t időpontban vett kép Haar wavelet dekompozíciója

21. ábra. Tervezett bemenet: A két kép közötti elmozdulás a kép számos helyén előfordul

A két időpontban vett kép dekompozicióját összefűzve nyomon követhető az elmozdulás.



(a) Fúzió: az elmozdulás jól látható a két kép között az összes szinten

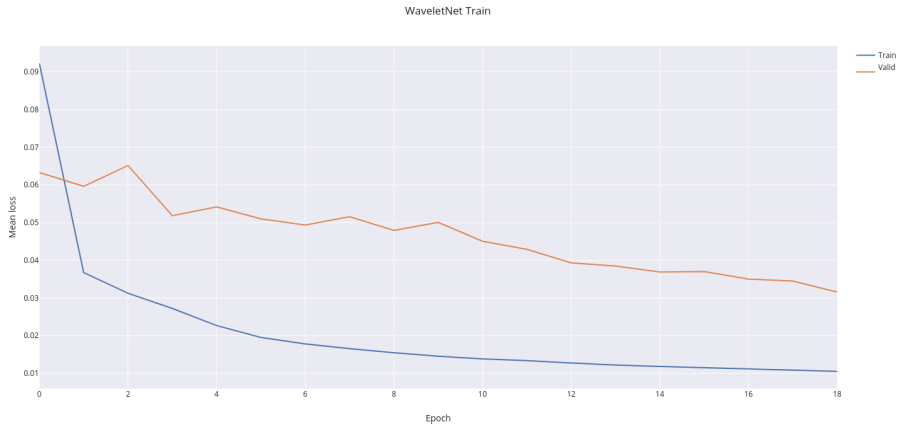
(b) Rekonstukció

22. ábra. Fúzió együtthatóknál és a rekonstrukciója

7. Eredmények

7.1. Kiértékelési módszer

A kiértékeléshez a DeepVO által implementált, a KITTI[31] VO/SLAM benchmarkját használtam. Az egyes útvonalokon összegzett transzlációs hibát ([B | W]_translation), és a rotációs([B | W]_rotation hibát hasonlítottam össze. Baseline-nak az eredeti képen tanitott hálózatot vettem, amelyben a konvoluciós hálózatnak egy előre betanitott, úgynevezett flownetet vettem, majd 10 epochig tanitottam. A wavelet transzformációt az eredeti model paramétereivel inicializáltam, majd 19 epochig tanitottam. A tanítás során Adagrad optimalizálót használtam, 0.0001-es learning rate-el. Az alábbi ábrán a Wavelet alapú tanitás train és validációs loss értékeinek alakulása látható az epochok során. A tanítás rövidsége miatt nehéz eldönteni azt, hogy overfitting állt-e elő. A bemeneti kép mindkét esetben csökkentve lett 608x184 pixelre. A batch size 8 volt.



23. ábra. Wavelet hálózat tanítása

7.2. Transzlációs és rotációs hiba

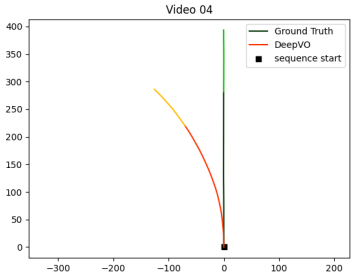
Az alábbi táblázatban összefoglaltam az eredményeket. A táblázat alapján jól látszik, hogy a transzlációs hiba lényegesen kisebb a Wavelet transzformáció esetén, azonban a rotációs hiba bizonyos esetekben nagyságrendekkel nagyobb. Az eredeti cikk 200 epochig tanitotta a hálózatot, ami lényegesen hosszabb a mostani kiértékelésnél.

Route	B_translation	W_translation	B_rotation	W_rotation
4	20.01	7.29	462.98	1931.91
5	575.40	200.89	31452.44	47205.97
7	286.71	204.87	4811.23	4081.76
9	504.15	190.41	19843.53	252777.87
10	358.78	153.42	7069.71	86960.68

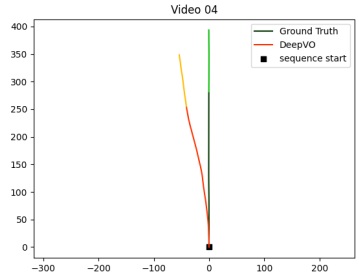
1. táblázat. MSE hibaértékek baseline vs wavelet

7.3. Prediktált trajektóriák

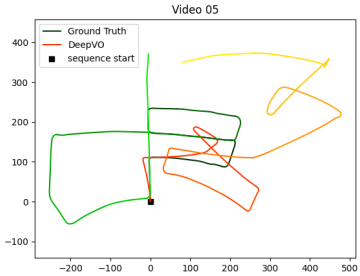
A modellek összehasonlításához az általuk prediktált trajektóriák különbségét ábrázoltam, az eredeti bejáráshoz képest. Az útvonalak között mind sebességbeli, mind komplexitásbeli különbségek voltak. Az egyenes úthoz tartozó predikciók közül a Wavelet transzformáció nélküli modell bizonyult jobbnak. Az 5. útvonalon mindkét modell nagy eltérést mutat, a sok visszacsatlakozás egy különösen nehéz feladatnak bizonyult. A 7. útvonalon hasonló eredményeket kaptam, ahol jól látszik a wavelet-es előfeldolgozás esetén a transzlációs hiba csökkenése az eredetivel szemben. Az utolsó két teszt szekvencián egyértelműen az eredeti modell teljesített jobban.



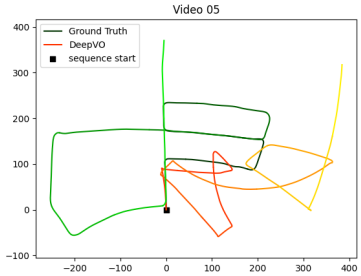
(a) Route 4 wavelet



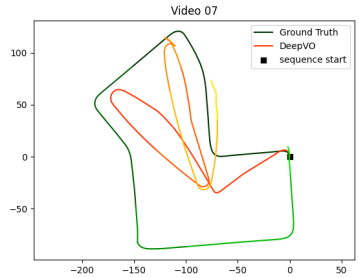
(b) Route 4 eredeti



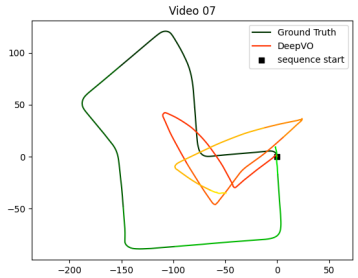
(c) Route 5 wavelet



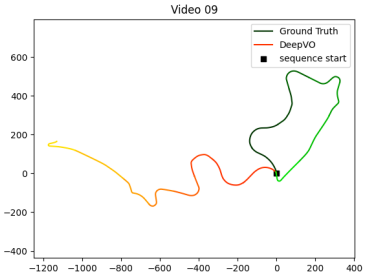
(d) Route 5 eredeti



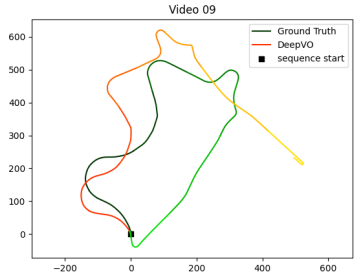
(a) Route 7 wavelet



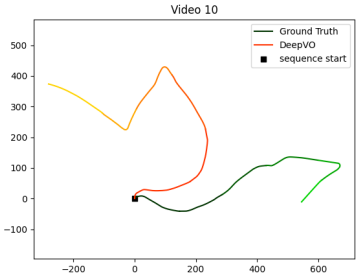
(b) Route 7 eredeti



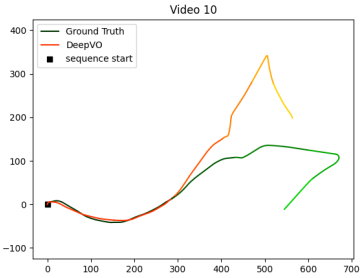
(c) Route 9 wavelet



(d) Route 9 eredeti



(e) Route 10 Wavelet



(f) Route 10 eredeti

24. ábra. Prediktált útvonalak összehasonlitása, bal oldalon wavelet, jobb oldalon eredeti kép

8. Összefoglalás és továbbfejlesztési lehetőségek

Dolgozatomban ismertettem a vizuális odometria problémáját, a Deep Learning alkalmazását a szakirodalomban, majd a wavelet transzformációt, mint lehetséges feature extractor-t vizsgáltam meg a KITTI adathalmazon.A DeepVO architectúrán kipróbáltam a Wavelet transzformációt db2-es, 2-es szintű diszkrét wavelet transzformáció által kapott előfeldolgozást, amit ugyanazon a hálózaton futtattam. A rövid tanitástól függetlenül általánosságban elmondható, hogy a Wavelet transzformáción alapuló hálózat a baseline-hoz képest a transzlációs hibát csökkentette, azonban a rotációra érzékeny volt. Következő lépésként egyrészt érdemes lenne a cikkben emlitett 200 epochig tanitani a Wavelet alapú hálózatot, másrészt megvizsgálni az eredeti hálózat méretének csökkentését, például az LSTM hidden layer méretét, vagy a konvolúciós rétegek számát. Ezen kívül az újabban megjelent, felügyelet nélküli architectúrákon szeretném a diszkrét wavelet transzformációt tesztelni.

Hivatkozások

[1] R. Giubilato, M. Pertile, and S. Debei, „A comparison of monocular and stereo visual fastslam implementations," 06 2016.

[2] Wikipedia, „Visual odometry," 2021.

[3] S. Segvic, „A typical viusal odometry pipeline."

[4] dsprelated, „Discrete wavelet transform filter bank implementation." `https://www.dsprelated.com/showarticle/115.php`, 2021.

[5] R. R. Merry, „Wavelet theory and applications : a literature study," 2005.

[6] L. Shiloh-Perl and R. Giryes, „Introduction to deep learning," 2020.

[7] Wikipedia, „Machine learning," 2021.

[8] J. Chen, „Neural network," 2021.

[9] databricks, „Neural network image." `https://databricks.com/glossary/neural-network`.

[10] J. Brownlee, „Model of a simple neuron." `https://machinelearningmastery.com/neural-networks-crash-course/`.

[11] Wikipedia, „Backpropagation," 2021.

[12] Brilliant, „Backpropagation," 2021.

[13] Z. Li, W. Yang, S. Peng, and F. Liu, „A survey of convolutional neural networks: Analysis, applications, and prospects," 2020.

[14] T. datasciene, „A comprehensive guide to convolutional neural networks."

[15] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, „Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton, „Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25* (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), pp. 1097–1105, Curran Associates, Inc., 2012.

[17] K. Simonyan and A. Zisserman, „Very deep convolutional networks for large-scale image recognition," 2015.

[18] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, „Going deeper with convolutions," 2014.

[19] K. He, X. Zhang, S. Ren, and J. Sun, „Deep residual learning for image recognition," 2015.

[20] A. Tampuu, T. Matiisen, M. Semikin, D. Fishman, and N. Muhammad, „A survey of end-to-end driving: Architectures and training methods," *IEEE Transactions on Neural Networks and Learning Systems*, p. 1–21, 2020.

[21] „Recurrent neural networks." `https://en.wikipedia.org/wiki/Recurrent_neural_network`.

[22] A. Mittal, „Understanding rnn and lstm."

[23] Matlab, „Long short term memory networks."

[24] T. D. Science, „Generating images with autoencoders."

[25] S. Wang, R. Clark, H. Wen, and N. Trigoni, „Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2043–2050, 2017.

[26] S. Wang, R. Clark, H. Wen, and N. Trigoni, „Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks," in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pp. 2043–2050, IEEE, 2017.

[27] M. J. M. M. Mur-Artal, Raúl and J. D. Tardós, „ORB-SLAM: a versatile and accurate monocular SLAM system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[28] M. Ramamonjisoa, M. Firman, J. Watson, V. Lepetit, and D. Turmukhambetov, „Single image depth prediction with wavelet decomposition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 2021.

[29] R. Li, S. Wang, Z. Long, and D. Gu, „Undeepvo: Monocular visual odometry through unsupervised deep learning," *CoRR*, vol. abs/1709.06841, 2017.

[30] V. M. Babu, S. Kumar, A. Majumder, and K. Das, „Undemon 2.0: Improved depth and ego motion estimation through deep image sampling," *CoRR*, vol. abs/1811.10884, 2018.

[31] A. Geiger, P. Lenz, and R. Urtasun, „Are we ready for autonomous driving? the kitti vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.