PARALLEL VIDEO PROCESSING

Group 9
0113110 陳柏翰 (資工系 大三)
0310511 孫　誠 (資工系 大二)
0316213 蒲郁文 (資工系 大二)

INTRODUCTION

- What we initially intended to do: Face tracking

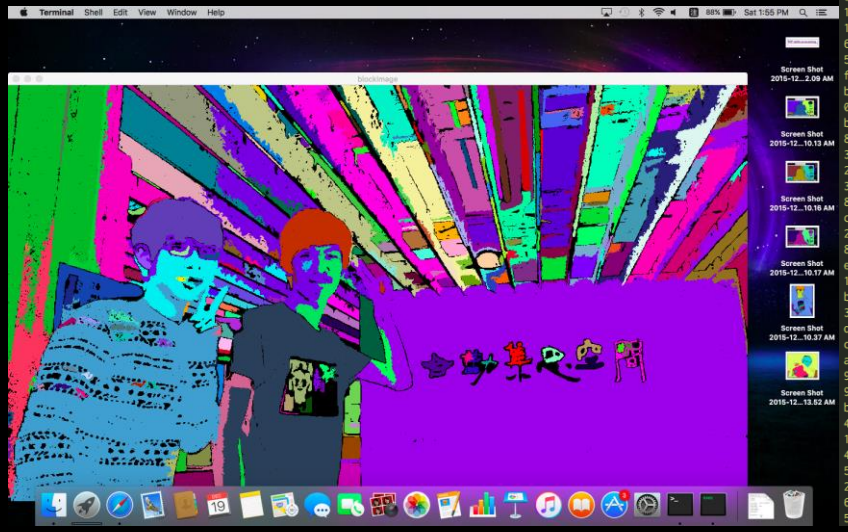
1280*720 @30fps

Track

Win API

INTRODUCTION

- Couldn't get a stable result for gesture synthesis
- Algorithm was already unsuitable for parallelization
- Looked for existing APIs. Too hard/complicated.
- O(N) speedup?

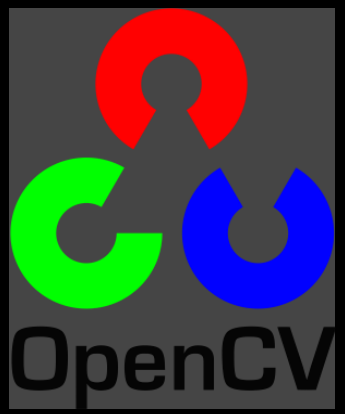


```
~/GitHub/Leilei_Face-tracking_Algorithm/Source @ lnishan-MacBook(apple): git log --format=oneline
15f4035695157fa8c4395ed2572a59a28eb242c2 Simple algorithm
6a851ac8256367a4c7b079a881a80e9959bb76fb Updated .gitignore
81f78946c258d8ae91e83f8b12d10d35d8d5c0ed Merge branch 'master' of github.com:sunset1995/Leilei_Face-tracking_Algorithm
cc6841a31c425a38e71686885d2c45acc240e99a Street Art
1affed63b6346f236b5f6b1887ea59cbbb2d691b Last commit before continuing developing
1651b8668babb6c221ee5d7c5855463e6c08af36 4-direction
68efb2572a38024437bd0dd04696d0defc13b5fb Neighbor color reference
58309ad8899faab8ba07a3e256d0a82b29f1ee4c Merge branch 'master' of github.com:sunset1995/Leilei_Face-tracking_Algorithm
fbc062d0fd83d17174e05afe0d6b3440690174cb Add tool mahatan step array
b93ba7dd1d71fb8505ebb360916c9cfae3c40f4b Deleted unused code comments
0154043a3333c5392115595ca80af844cbe3f19ee removed .swp files
b628b27a4fff4fb7266f881efc200b360a111685 Color segmentation complete
8a97c35ec2523758099c4272ee072aa518979c1b OSX makefile
3c45fae29d3632e68e3ebcad1f93a9571d3ae8d0 Fixed a bug with dp boundary check, enlarged WIDTH limit
2441801d350114e5276ebf26f75995fb0894cfd2 Add noBrain algorithm
352e3364cf3133126c706b3b7027e10e263ff830 Update .gitignore
8a6760536e21f11a84150da7fea7f4da9943edef Remove facedetect's clear noise
cbbe00356409f973401255750c6d03eaacbb16f9 Edit gitignore
27994c732bcc8d0628a12289f167c348d3ca61ed Add makefile
883fa5390ae955734abb734428e073cade05ab14 Seperate facedetect
eebfe7f99c69a362f87091519b7cabbd489d1bab Add config.h
16bae0722c102fdd581e635946bd91a4bd82d378 Add config.h
b2958d51b9a845c9e050fc626152f7a51288bb58 Seperate testing skincolor
3caf80893e653d3fb6d67a5a531c9f79cacc960c Seperate bgrtester
d49c7c8736800e88a183995cc9c8b2f3991aa414 Edit install script. Add dependency
c06a264450247f2b4ea3be4261550e25ac5215c4 Add gitignore. Ignore build
a6ab785205f55f29818fd3e026f7045e4bfea692 Merge pull request #1 from lnishan/master
921a9200728a309ee0525915d3e8b97700e7b425 Removed obsolete compile script
9ca258f2364677d58222fadbd121c95f3afdbd13 Fixed compile script bug
b6512874034fc8935913238f5058d46a18b647f6 Add annotation in compile script
4b1bc72fa54954a764ed9747f16b7b9cfb7d57b4 Add clear noise
16dbc99ca0f7a537aa0c605a60bbd1141b0b0655 Functionize skin color judge
4c2a427c041039652fef601ea4bf7b34267cf9cc Arrange worksapce
563e5bd3ad47568cf72ca336dbbc46f257230a98 modulize
27ee9e551ad3c9f4ede1bb06d5d6cabc21590393 detect skinColor via euler dif
63e386f9f3f55046f21c9e1392fd59f63fdb5129 findSkin via BGR
5d07075af1b462b96edacd4b220a5daf11c1c5ed findSkin color
```

https://github.com/sunset1995/Leilei_Face-tracking_Algorithm

ENVIRONMENT SETTINGS

- Tools used:
  Pthread OpenMP NVIDIA CUDA FPGA
- APIs: OpenCV 2.4.11, OpenCV 3.0



- PC: Win 10, i7-3770, 8GB *2, GTX670
  VS12(2013), Cygwin gcc 5.2.0
- NB1: Ubuntu 14.04, i5-4200H, 8GB, gcc 4.8.4
- NB2: Mac OSX El Capitan, i5-4260U, 8GB, gcc 5.2.0
- VPS: Ubuntu 15.10, 2 vCores, 2GB, gcc 5.2.0
- FPGA: ML506 Evaluation Platform

AN INVESTIGATION INTO O(N) PARALLELIZATION

- Previous experience says

```
#pragma omp parallel for
for (int i = 0; i < SIZE; i++) a[i] = something;
```

  => gives no speedup

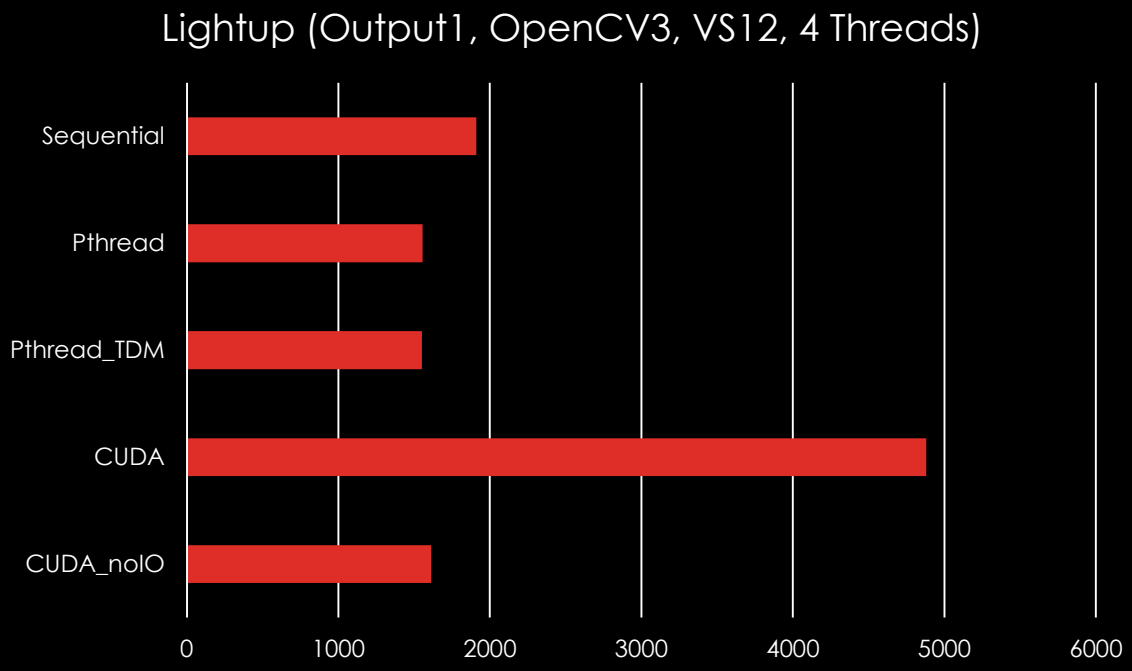
- Color Correction Matrix: Linear Transformation
  For each pixel…

$$\begin{bmatrix} R_c \\ G_c \\ B_c \end{bmatrix} = \begin{bmatrix} K_{11} & K_{12} & K_{13} \\ K_{21} & K_{22} & K_{23} \\ K_{31} & K_{32} & K_{33} \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} O_1 \\ O_2 \\ O_3 \end{bmatrix}$$

=> O(N)

LIGHTUP: AN O(N) EXPERIMENT

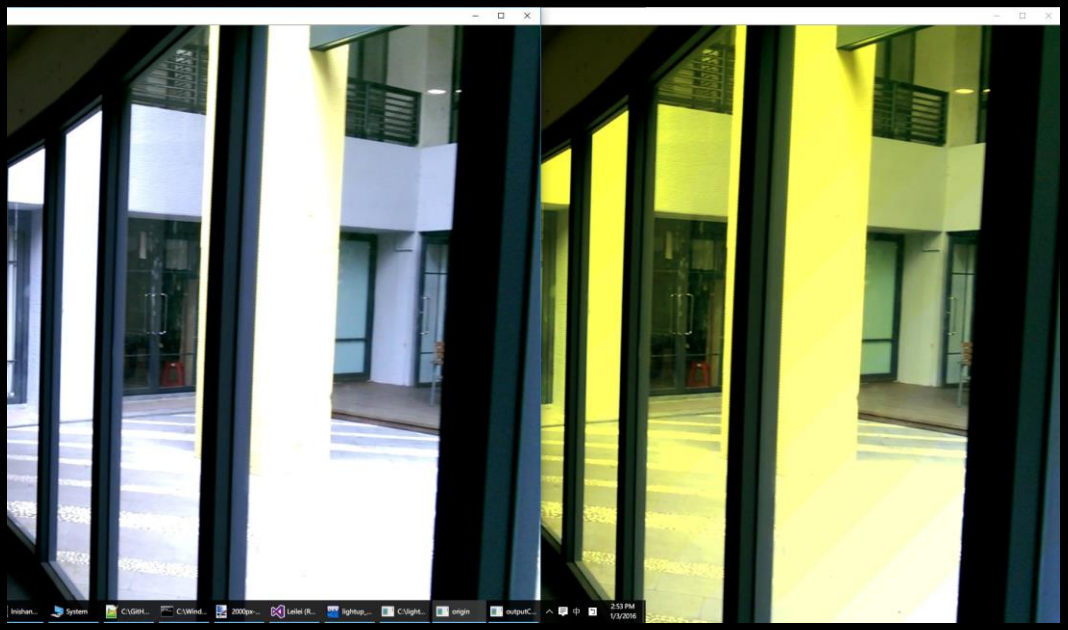
- Input: 360*240/1280*720/1920*1080 @30fps, 148 frames
- Output 1: For each pixel, [R', G', B'] = 2 * [R, G, B] + 5
  Then normalize the RGB values back to [0, 255].



- Slowdown when input is small
- Tiny speedup even when input is large
- Similar results across OpenCV 2.4, Ubuntu, OSX.

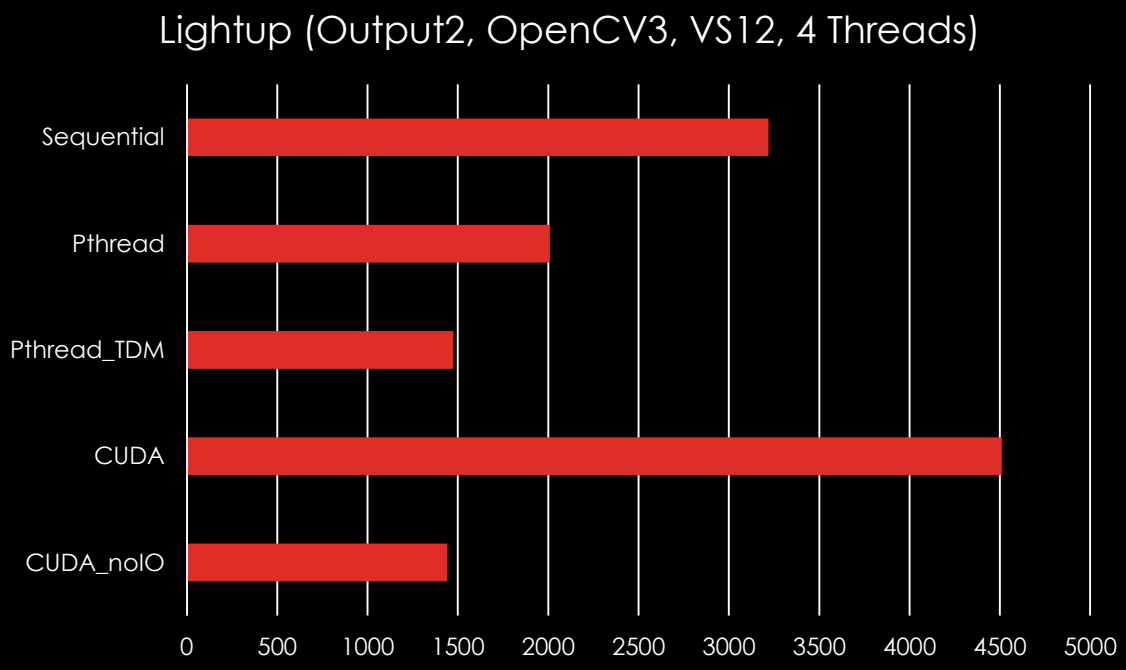
LIGHTUP2: AN O(N) EXPERIMENT

- Input: 360*240/1280*720/1920*1080 @30fps, 148 frames
- Output 2: For each pixel,
  [R', G', B'] = [R, G, B] * 1.125^[(x + y) / 128]



LIGHTUP2: AN O(N) EXPERIMENT

- Results are clearly better
- Time Division seems to be better than Frame Division



So, Impossible to speedup O(N) ?

LIGHTUP2.5: AN O(N) EXPERIMENT

- How about FPGA?
- Output 2.5: [R', G', B'] = [R, G, B] * 1.125^1024

Staggering results: **> 300x** speedup (256*256)
> 985x speedup (1920*1080)
… 100x faster than other methods

- Hardware friendly

```
for (int i = 0; i < 1024; i++)
      value = value + (value >> 3);
```

- Leave it till later~

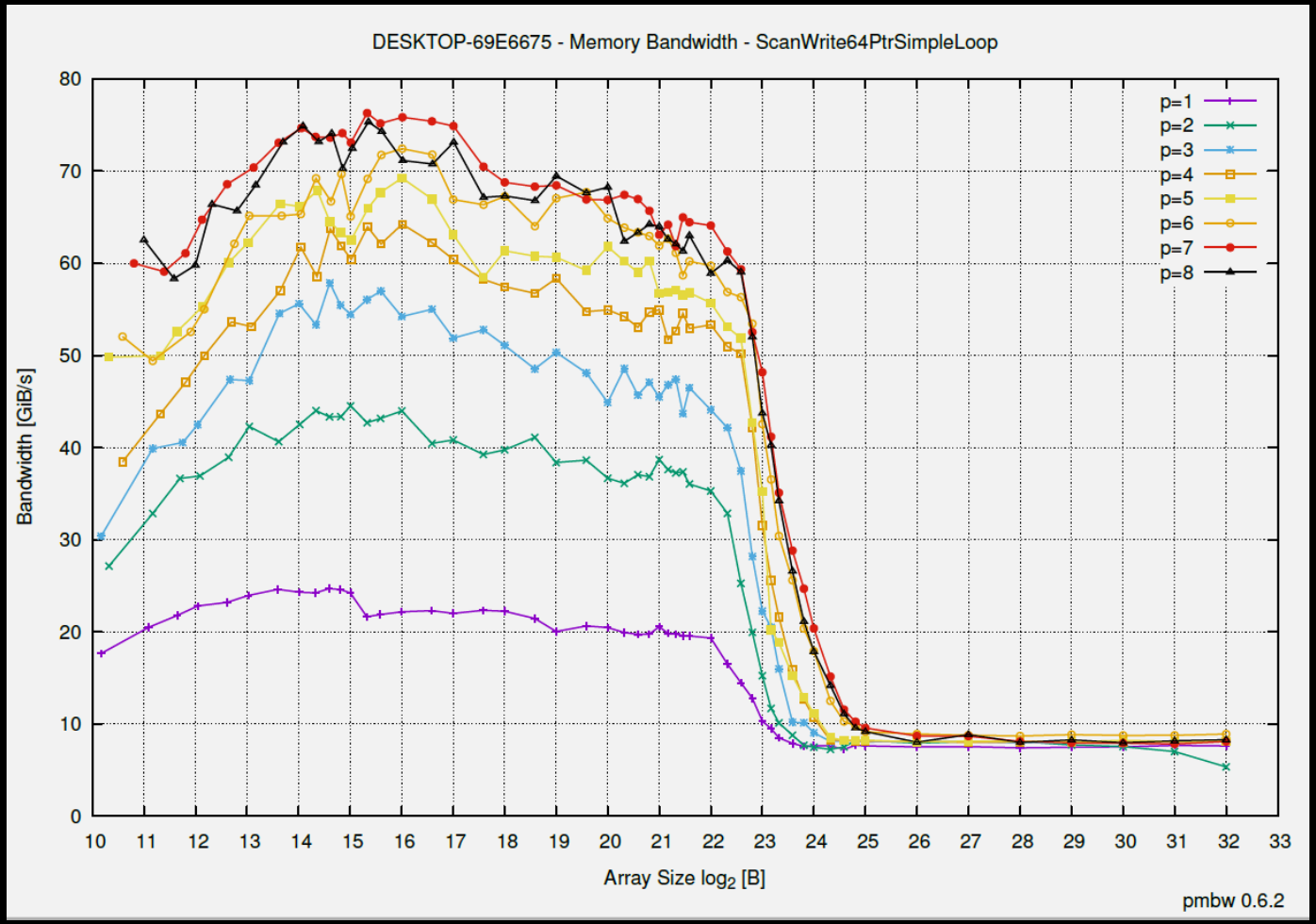
WHY NO SPEEDUP ON CPU!?

- Can different platforms be the issue?
- No, we've tested on Windows, Ubuntu, and MacOSX.

- Can optimization options be the issue?
- No,
  Both -O0 and -O2 indicated that sequential is almost as efficient. We also tried –fno-reorder-blocks.

- Can memory bandwidth be the issue?
- We thought so, but actually no, even though there's a certain limit in place.

PMBW - PARALLEL MEMORY BANDWIDTH BENCHMARK

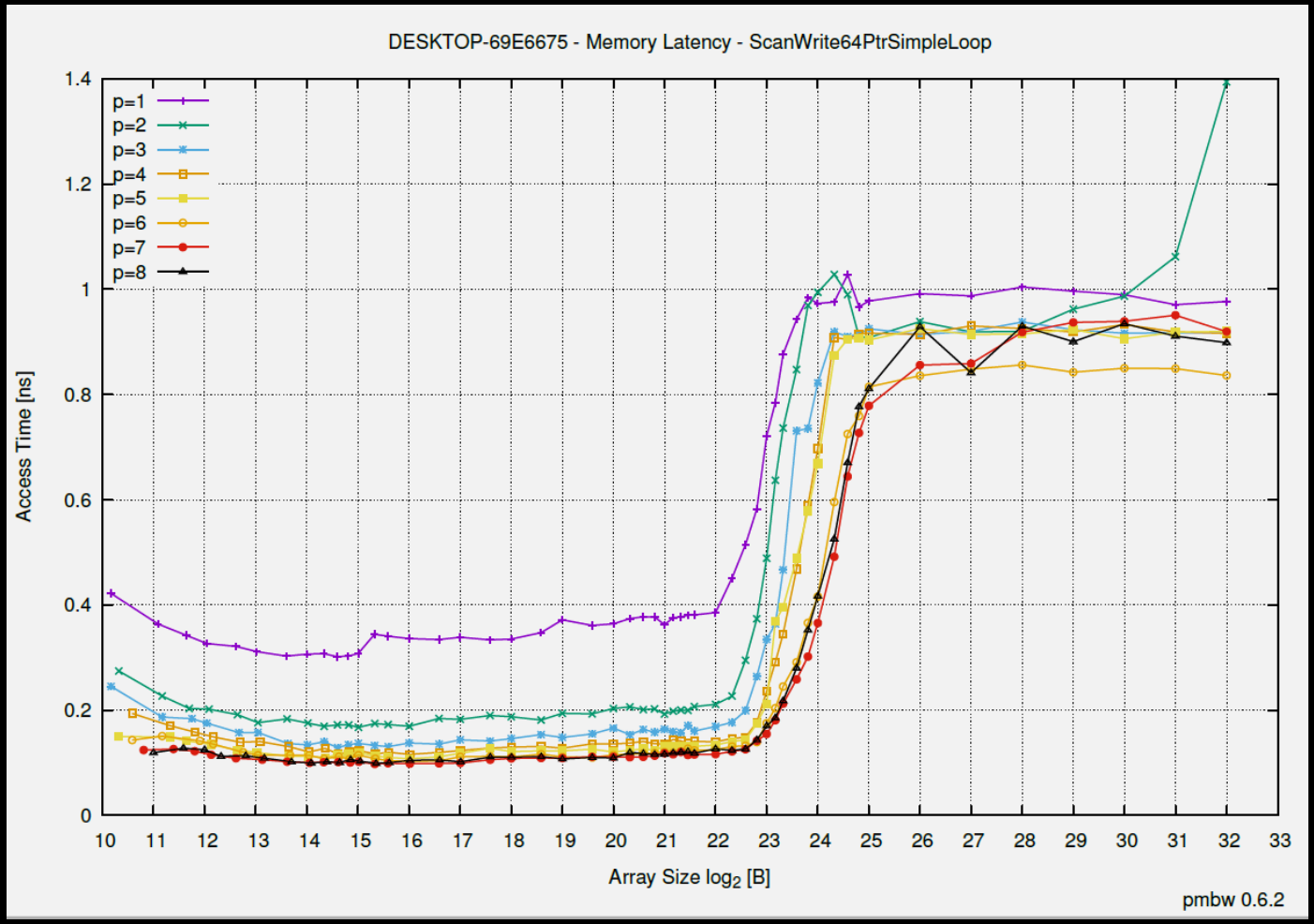


PMBW - PARALLEL MEMORY BANDWIDTH BENCHMARK

- I/O is parallelizable
- Speedup is definitely possible
- The reason for no speedup seems to be Automatic Vectorization (by the compiler)

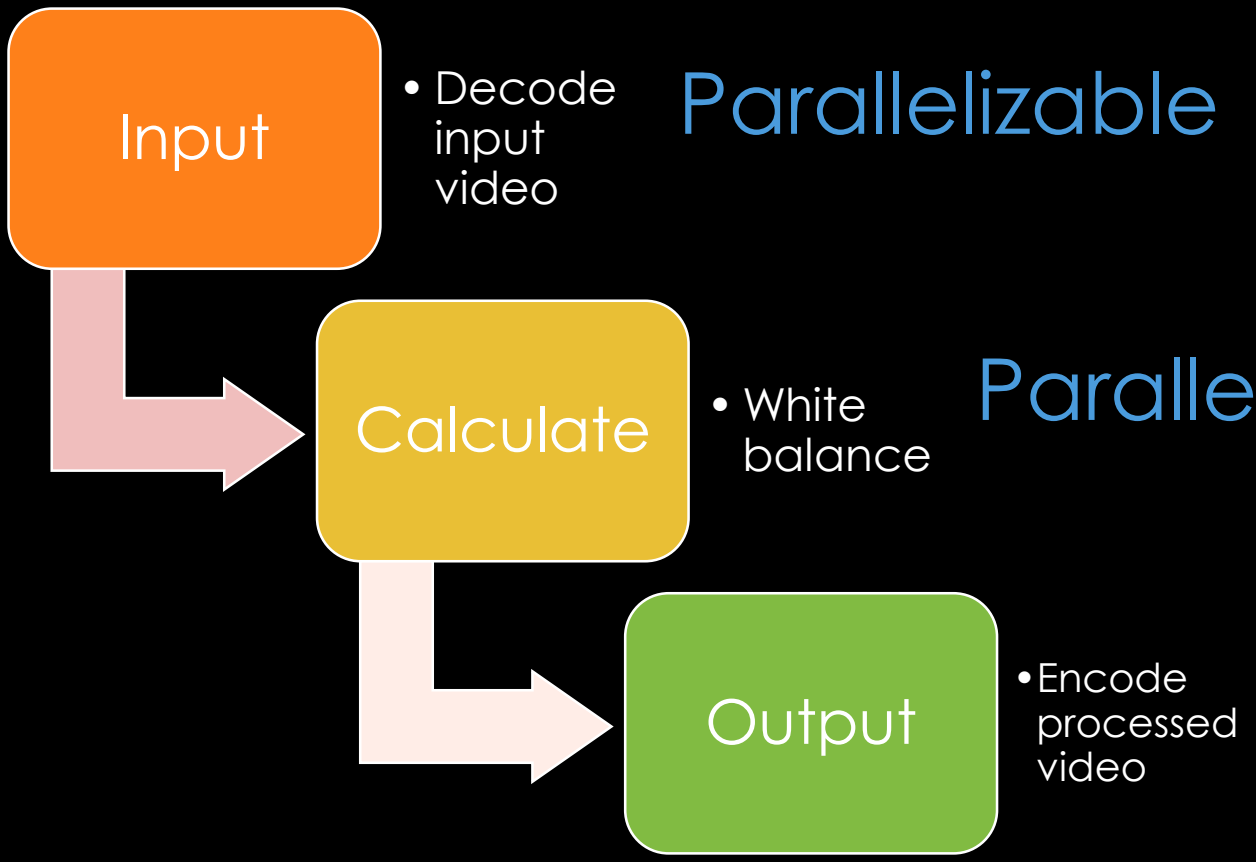
- The speedup for I/O or I/O bound applications is ~3.1x maximum on the PC (i7, Win10)
- ~2x on NB1(i5, Ubuntu 14.04)

PMBW - PARALLEL MEMORY BANDWIDTH BENCHMARK



WHITE BALANCE

BLOCK DIAGRAM



ALGORITHM

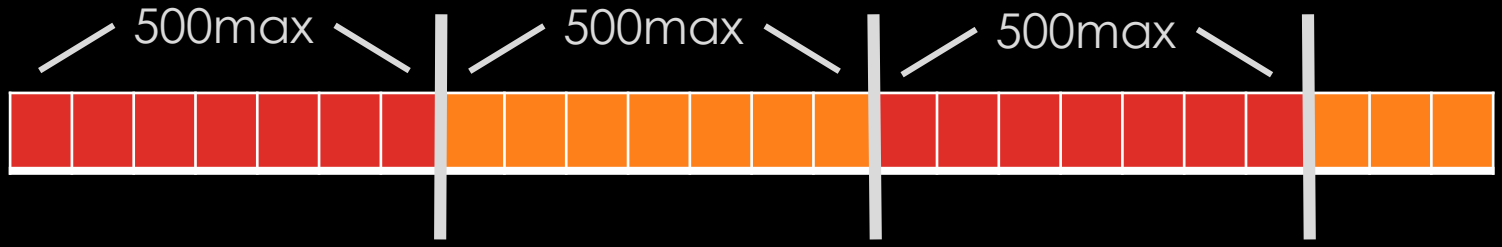
- Top-level idea: make the averages of Red, Green and Blue values equal on an image
- Algorithm:

1. Calculate the averages of Red, Green and Blue values, as denoted by (AvgR, AvgG, AvgB)
2. Using Green as reference, for each pixel,
   Red(new) = Red * AvgG / AvgR
   Blue(new) = Blue * AvgG / AvgB
3. If Red/Blue(new) > 255 Then Red/Blue(new) = 255

Gray World Algorithm, from
http://web.stanford.edu/~sujason/ColorBalancing/grayworld.html

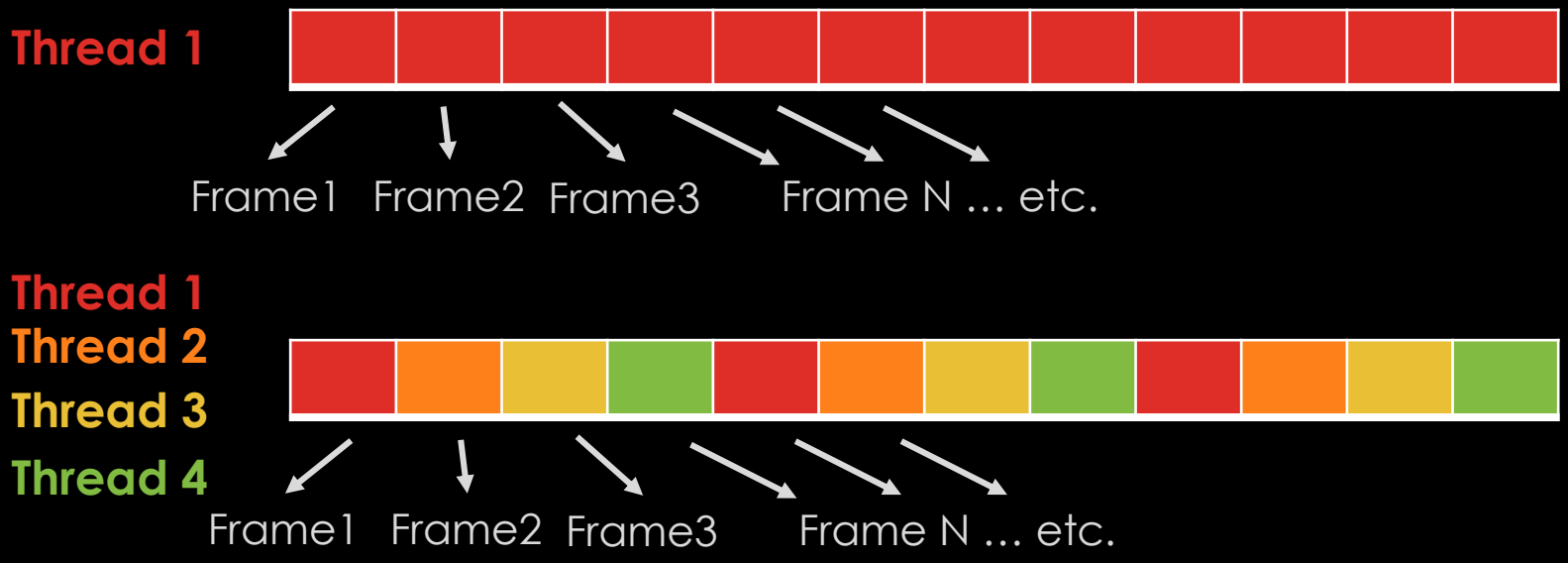
PARALLEL METHODS

- With scalability in mind and to prevent memory exhaustion,
- We divide the input video into chunks with each containing no more than 500 frames.



PARALLEL METHODS

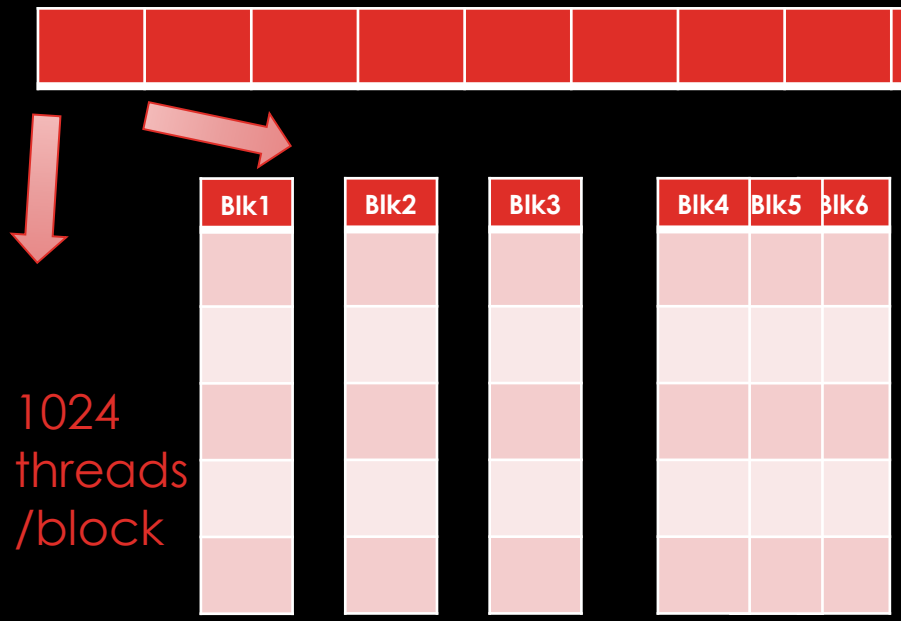
1. Pthread_TDM: C++11 Thread and Divide video by time



2. OpenMP_TDM: Same but with OpenMP as the API

PARALLEL METHODS

3. CUDA: 1frame/iteration, Each thread processes 1 pixel

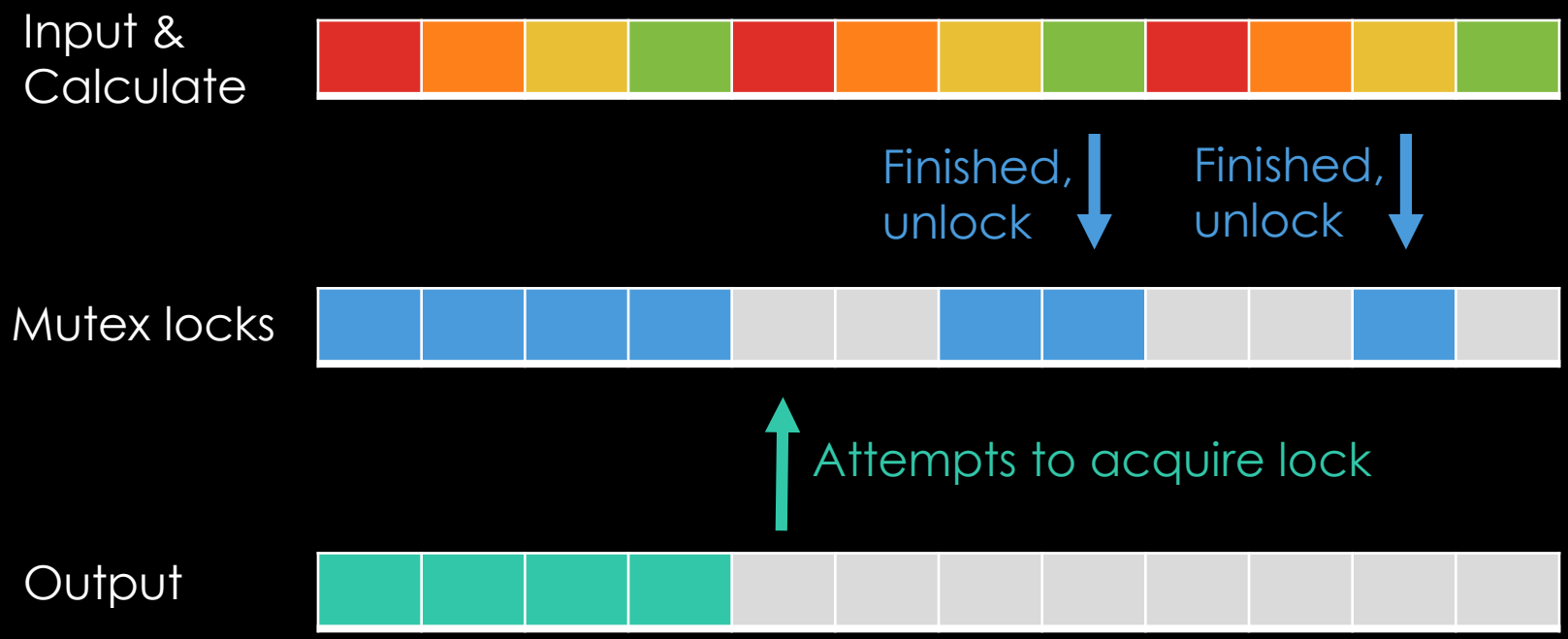


4. CUDA_TDM

Block1 Block 2 Block 3 Block 4

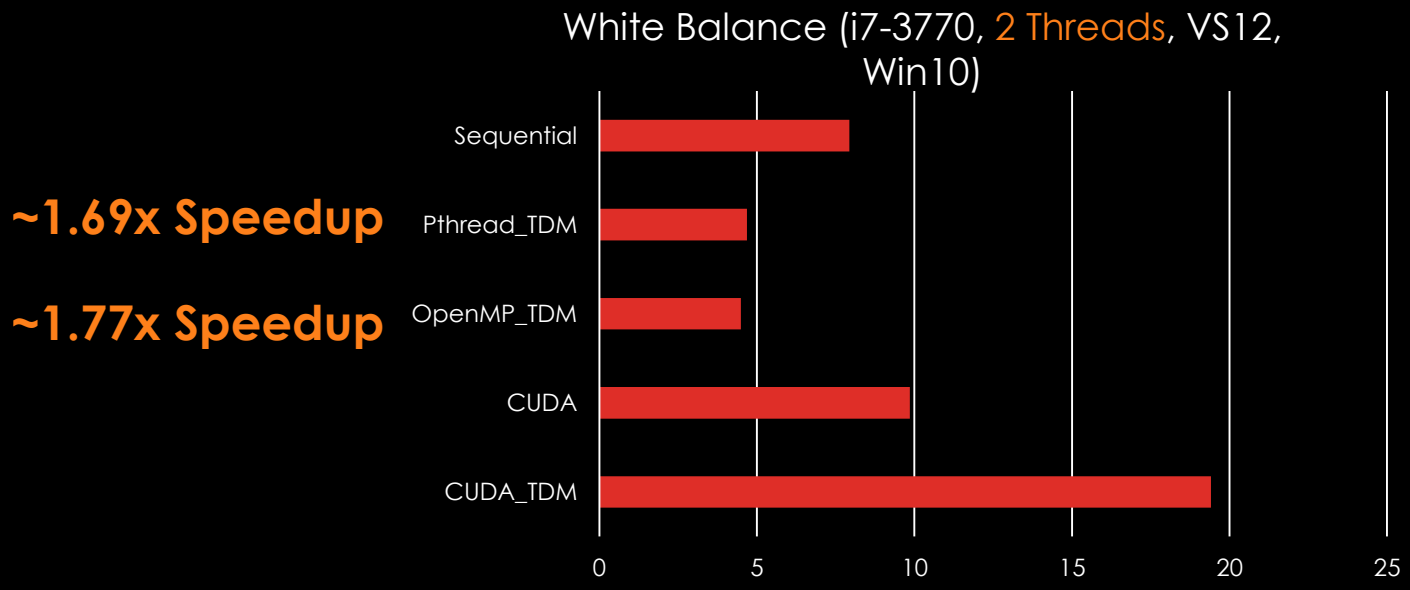
PARALLEL METHODS

5. Task parallel: Extended from Pthread_TDM (divide by time). Rather than waiting for Input and Calculate to finish, Create a separate thread specifically for output.



EVALUATION

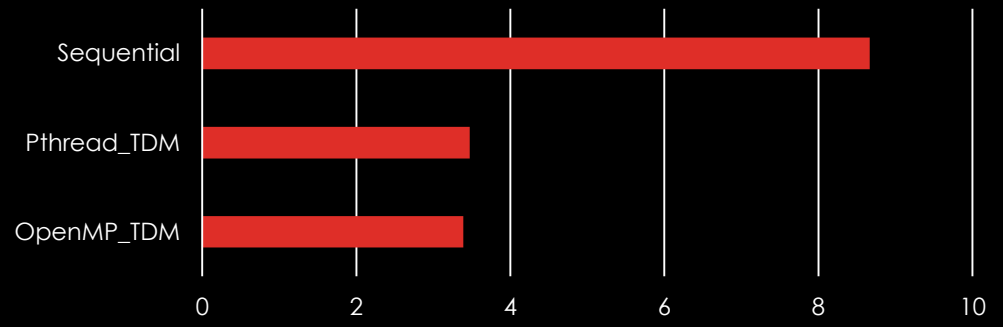
- First assume we don't need to output (encode). Perhaps we would like it as a module
- Data: 1280 * 720, 1422Frames (AVI)
- Environment: i7-3770(4C8T), 2 Threads, VS12, Win10



MORE THREADS

4 Threads

White Balance (i7-3770, 4 Threads, VS12, Win10)

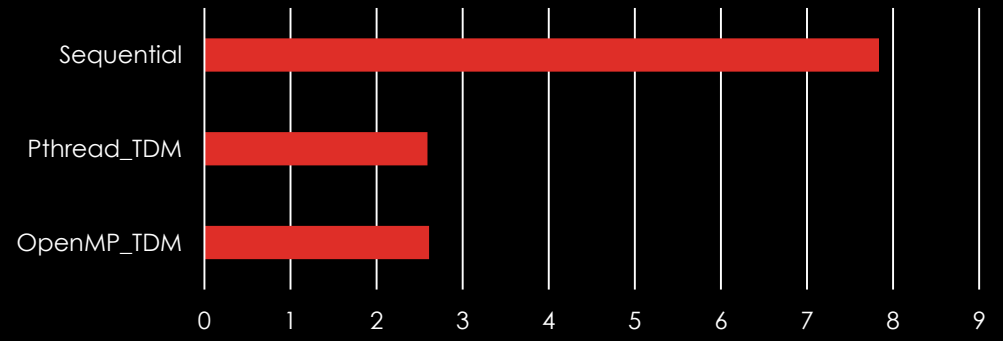


~2.49x Speed up

~2.56x Speed up

8 Threads

White Balance (i7-3770, 8 Threads, VS12, Win10)



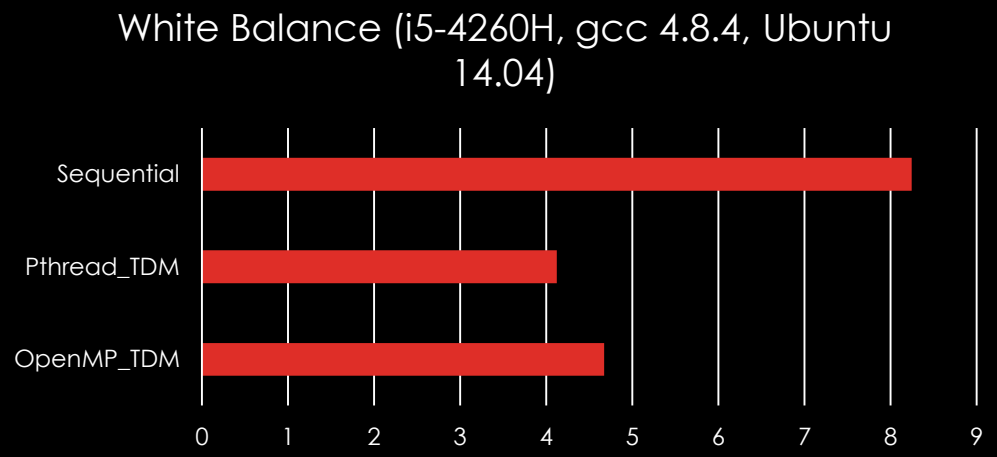
~3.02x Speed up

~3.00x Speed up

~3x matches with the memory bandwidth test result!

OTHER PLATFORMS

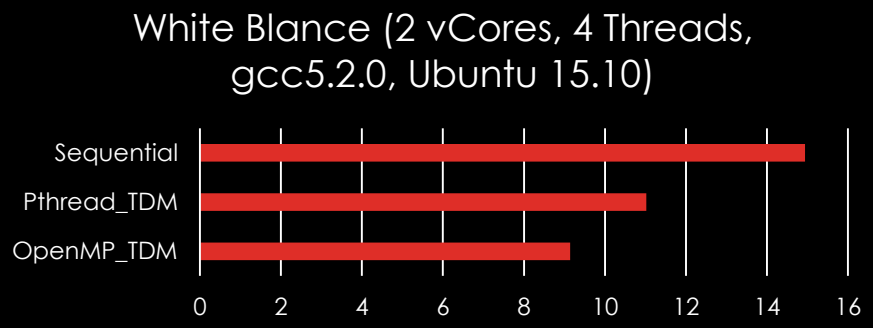
NB1: i5-4260H(2C4T), 2~4 Threads, gcc 4.8.4, Ubuntu 14.04



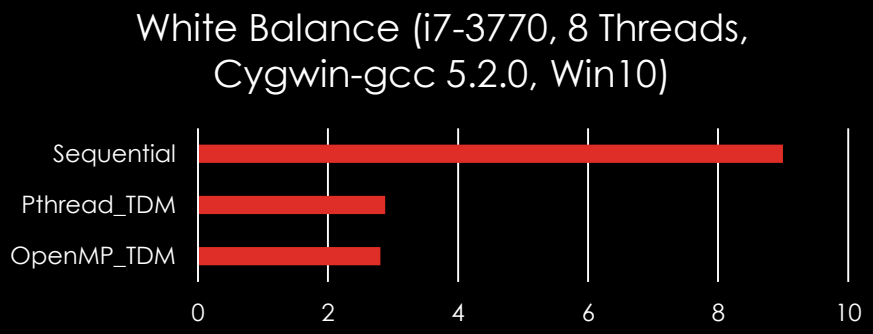
~2.00x Speedup

~1.76x Speedup

VPS (2 vCores)

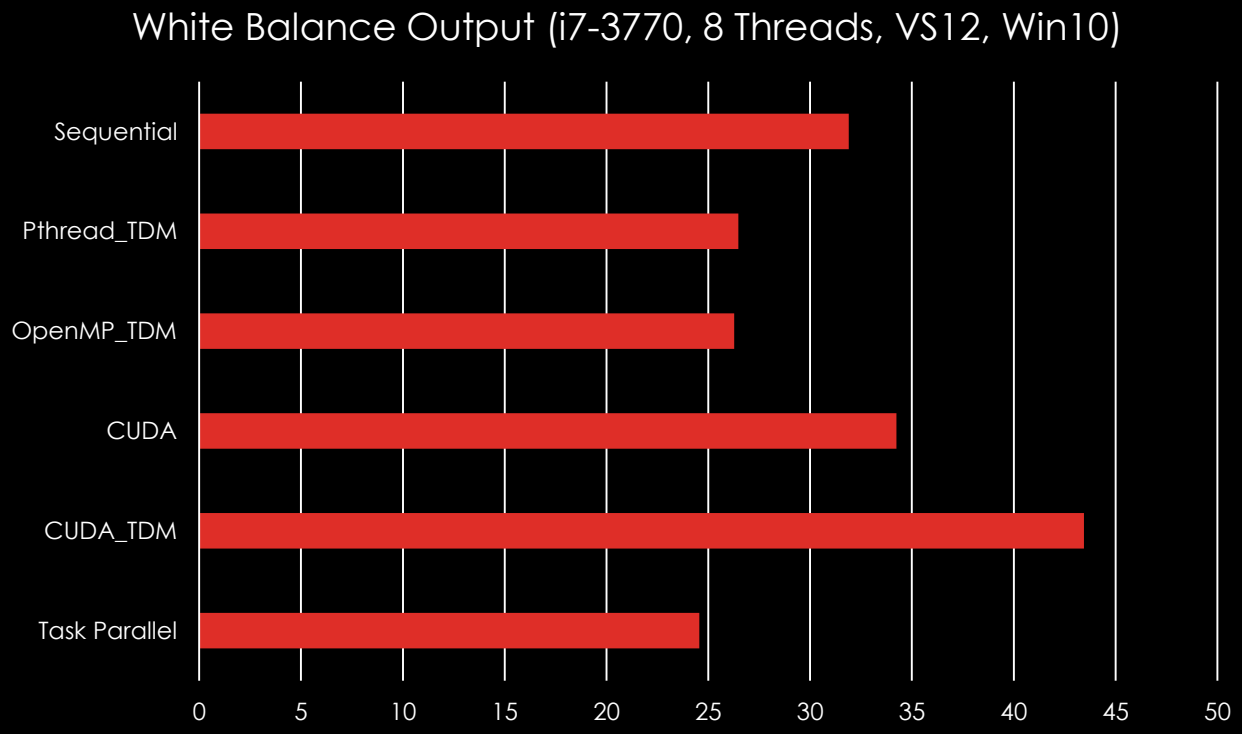


PC(i7) with Cygwin-gcc 5.2.0



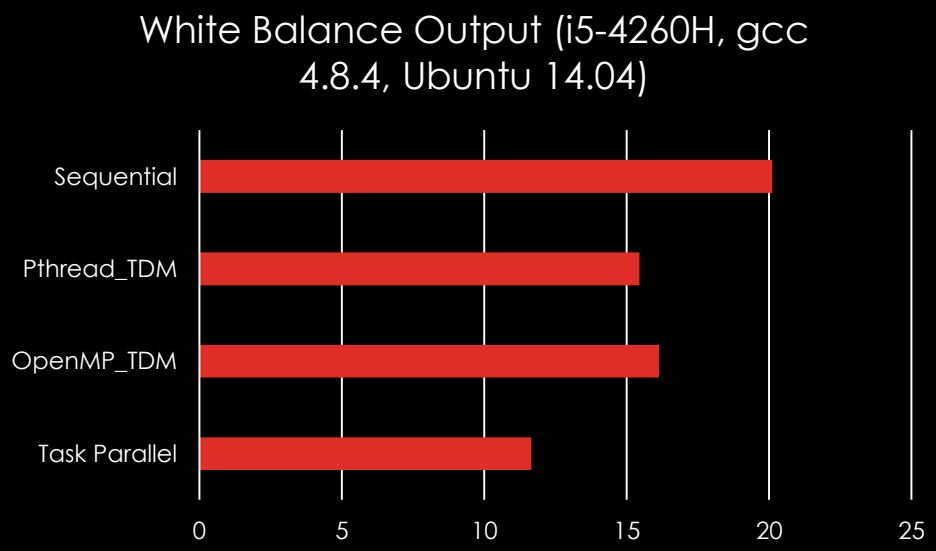
EVALUATION

- When output is taken into account, task parallel delivers better results.



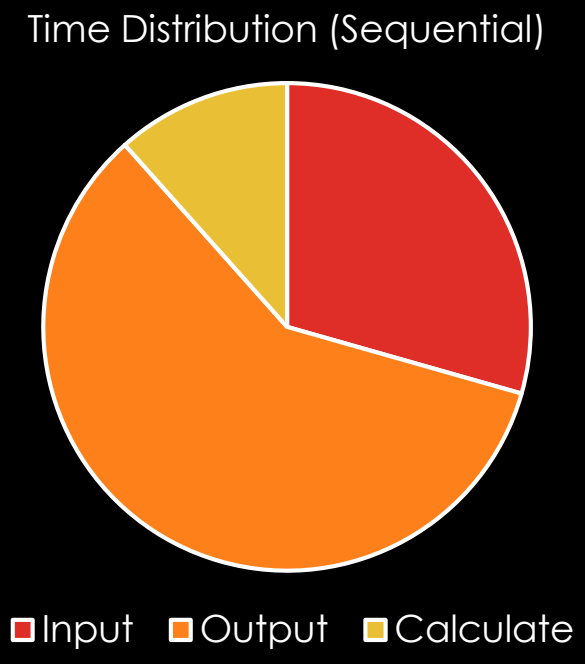
TASK PARALLEL

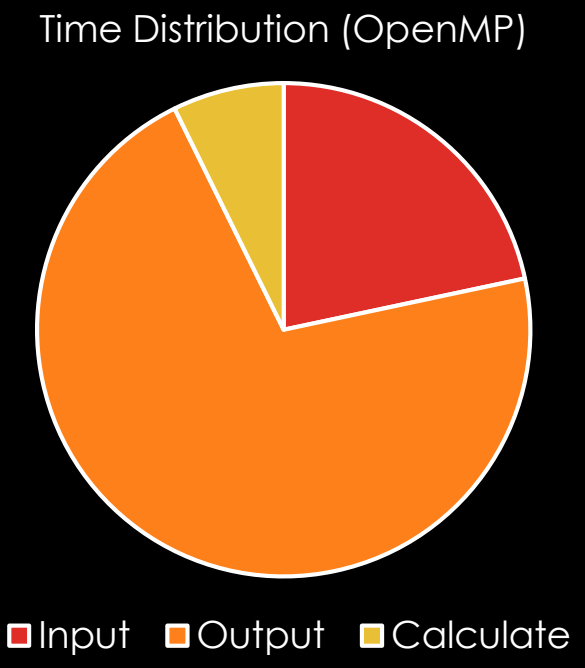
- This is especially true on NB1(Ubuntu 14.04)



TASK PARALLEL

- When output is taken into account, the whole process is actually output-bound





CONCLUSIONS

- Face tracking without trained data is hard
- Do not help the compiler, doing simple tasks sequentially is actually better sometimes
- CUDA isn't suitable for I/O bound applications
- On the CPU side of things, certain limits posed by memory bandwidth matters
- Input and Output video codec matters

RELATED WORK

- Illuminant Estimation: Gray World http://web.stanford.edu/~sujason/ColorBalancing/grayworld.html
- Ching-Chih Weng, Homer Chen, and Chiou-Shann Fuh, "A Novel Automatic White Balance Method For Digital Still Cameras" Circuits and Systems, 2005. ISCAS 2005. IEEE International Symposium on http://www.csie.ntu.edu.tw/~fuh/personal/ANovelAutomaticWhiteBalanceMethodforDigital.pdf

CONTRIBUTIONS

- 0113110 陳柏翰： CUDA, Cross platform compatibility, Automated Sync/Build/Test scripts, Evaluation, Presentation.
- 0310511 孫　誠： Pthread, OpenMP, Algorithm research and optimization, Parallelization Methods.
- 0316213 蒲郁文： Feasibility study of FPGA Parallel Heterogeneous Computing, Discussions with Prof. Chun-Jen Tsai.

See for yourself! at
https://github.com/sunset1995/parallel_analysis
https://github.com/yuwen41200/fpga-computing