Re-thinking CNN Frameworks for Time-Sensitive Autonomous-Driving Applications: Addressing an Industrial Challenge

Ming Yang[1], Shige Wang[2], Joshua Bakita[1], Thanh Vu[1], F. Donelson Smith[1], James H. Anderson[1], and Jan-Michael Frahm[1]

[1]*The University of North Carolina at Chapel Hill*
[2]*General Motors Research*

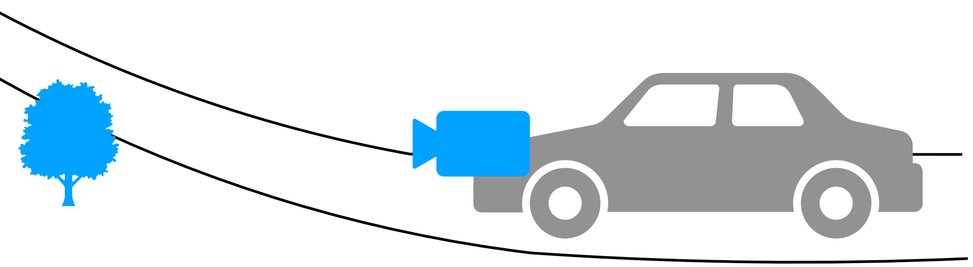
THE UNIVERSITY *of* NORTH CAROLINA *at* CHAPEL HILL

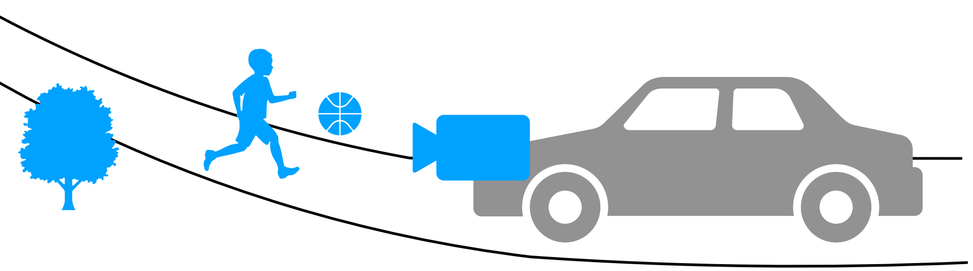
Re-thinking CNN Frameworks for Time-Sensitive **Autonomous-Driving** Applications: Addressing an **Industrial Challenge**

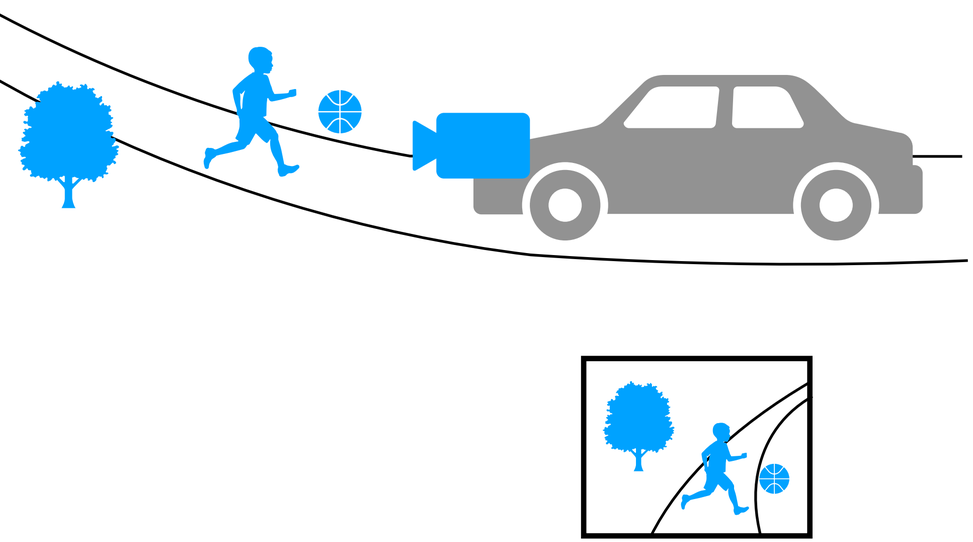
Ming Yang[1], Shige Wang[2], Joshua Bakita[1], Thanh Vu[1], F. Donelson Smith[1], James H. Anderson[1], and Jan-Michael Frahm[1]

[1]*The University of North Carolina at Chapel Hill*
[2]*General Motors Research*

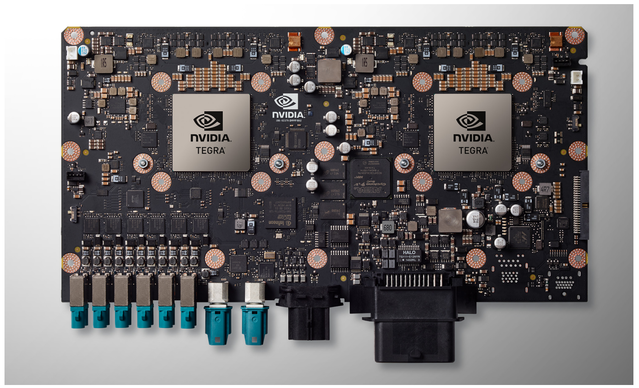
THE UNIVERSITY *of* NORTH CAROLINA *at* CHAPEL HILL

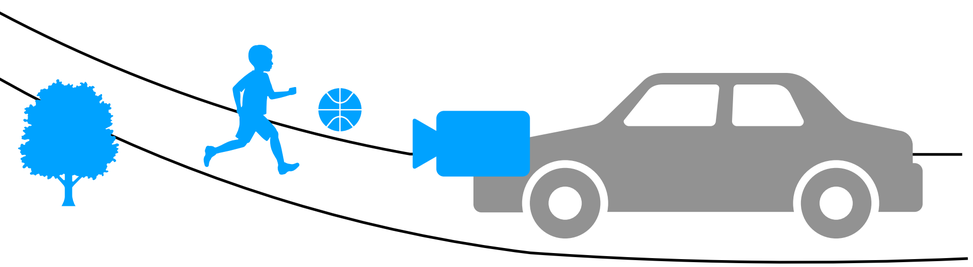


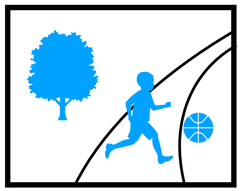


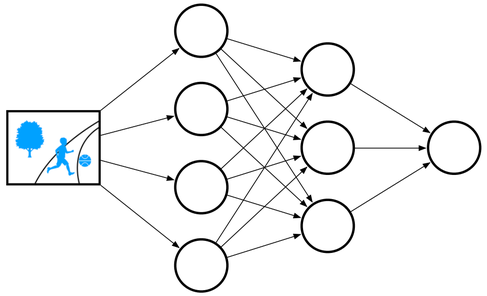


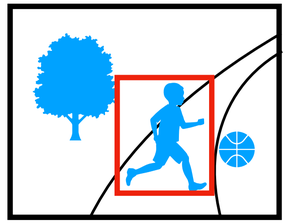
https://blogs.nvidia.com/blog/2016/01/04/automotive-nvidia-drive-px-2/



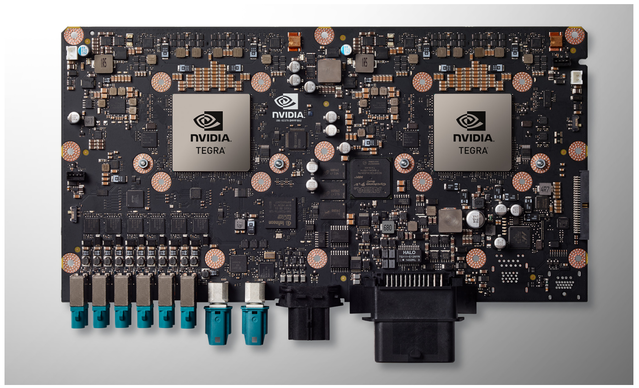




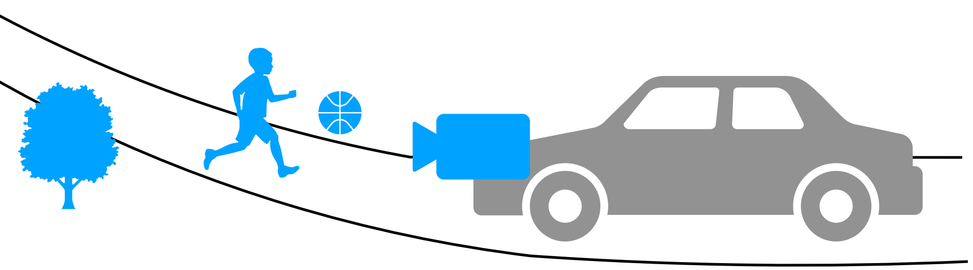


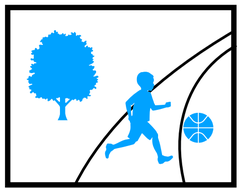


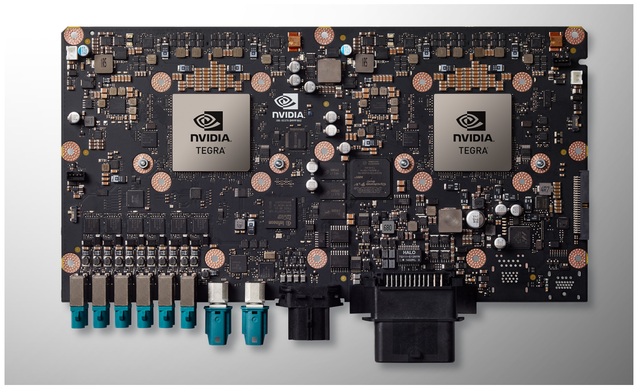
https://blogs.nvidia.com/blog/2016/01/04/automotive-nvidia-drive-px-2/

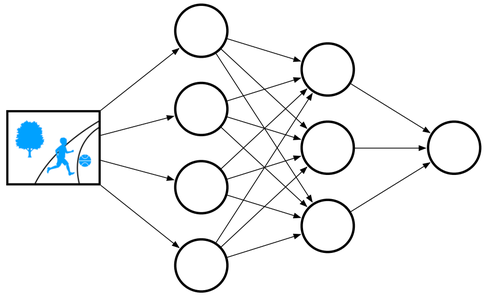


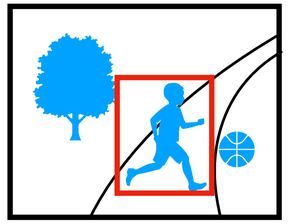
https://blogs.nvidia.com/blog/2016/01/04/automotive-nvidia-drive-px-2/





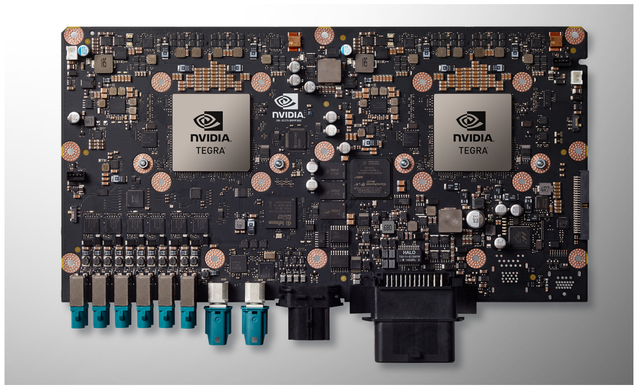


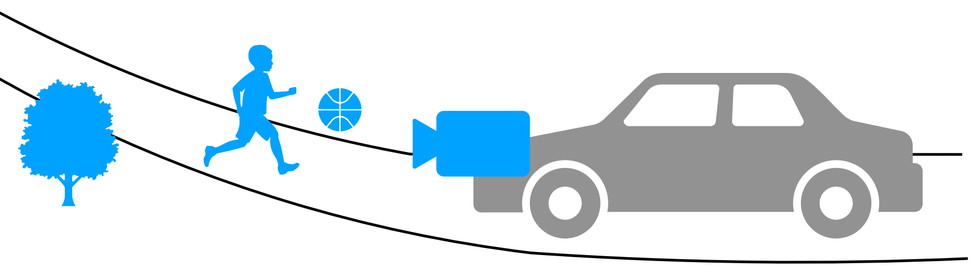


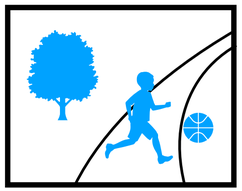


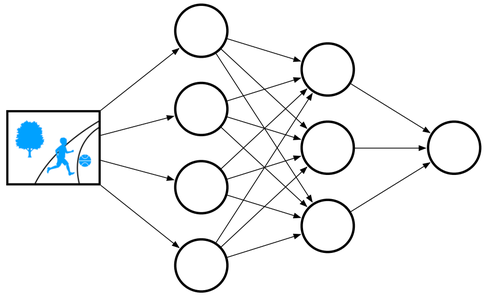
1. Response time

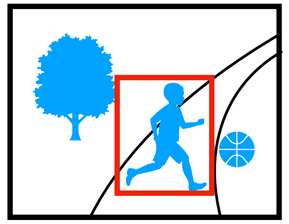
https://blogs.nvidia.com/blog/2016/01/04/automotive-nvidia-drive-px-2/





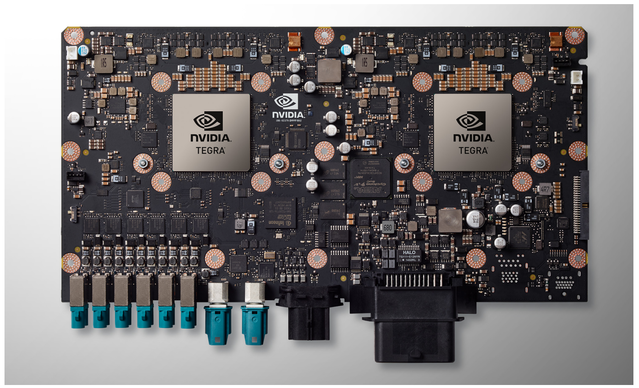


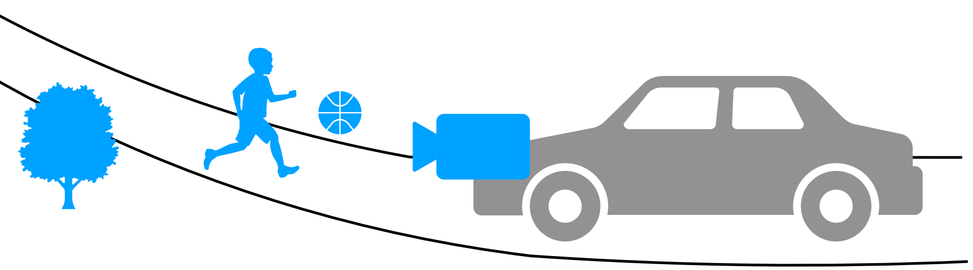


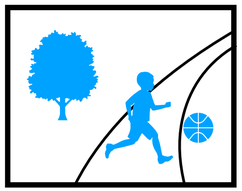


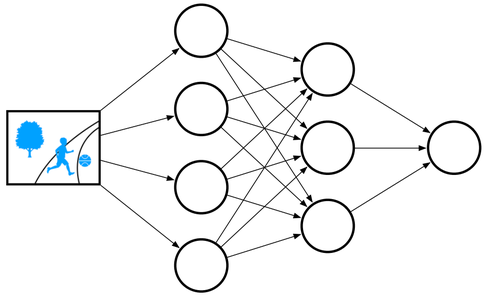
1. Response time
2. Accuracy

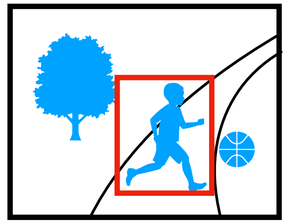
https://blogs.nvidia.com/blog/2016/01/04/automotive-nvidia-drive-px-2/









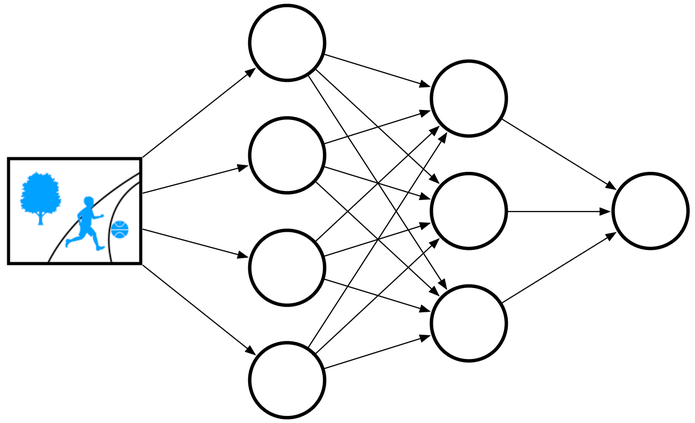


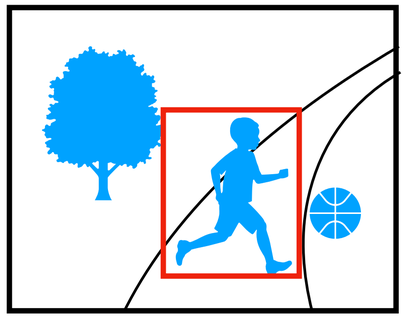
1. Response time
2. Accuracy
3. Throughput

Our focus

1. Response time
2. Accuracy
3. Throughput

https://blogs.nvidia.com/blog/2016/01/04/automotive-nvidia-drive-px-2/

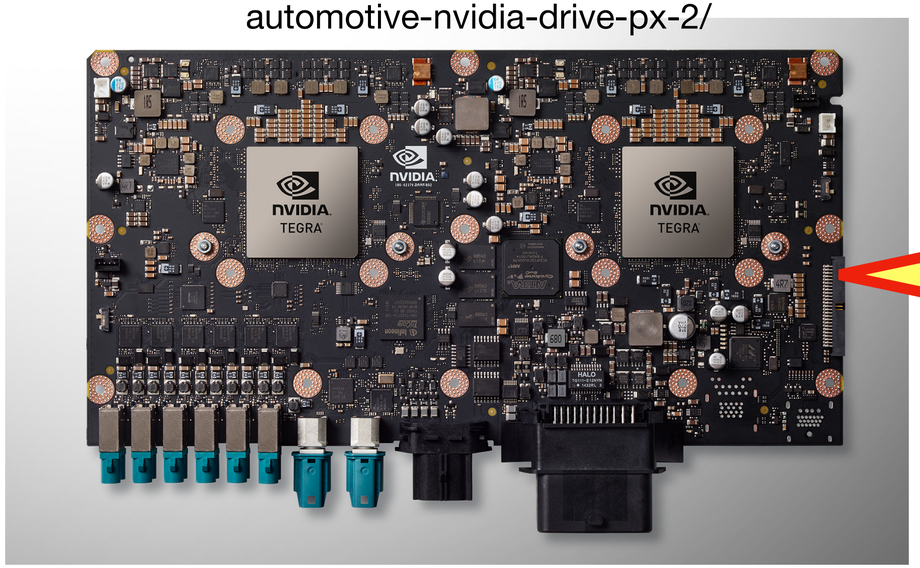




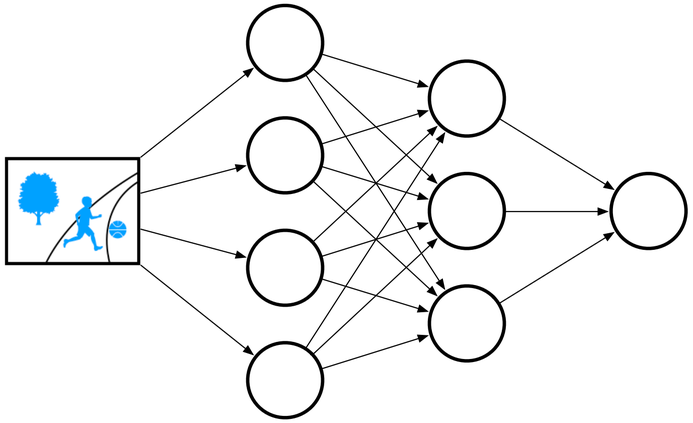
Our focus

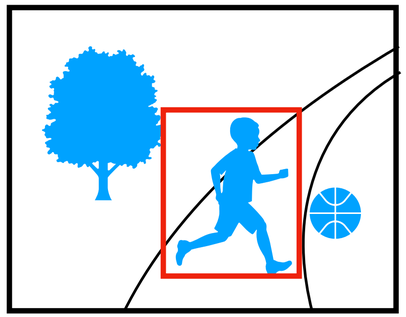
1. Response time
2. Accuracy
3. Throughput

https://blogs.nvidia.com/blog/2016/01/04/automotive-nvidia-drive-px-2/



Hardware resources are **constrained and expensive**.

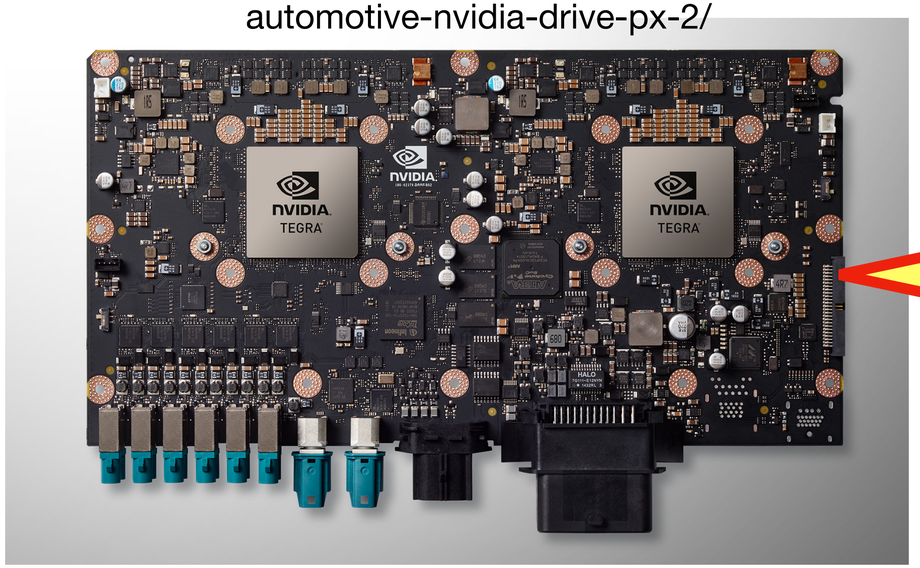




Our focus

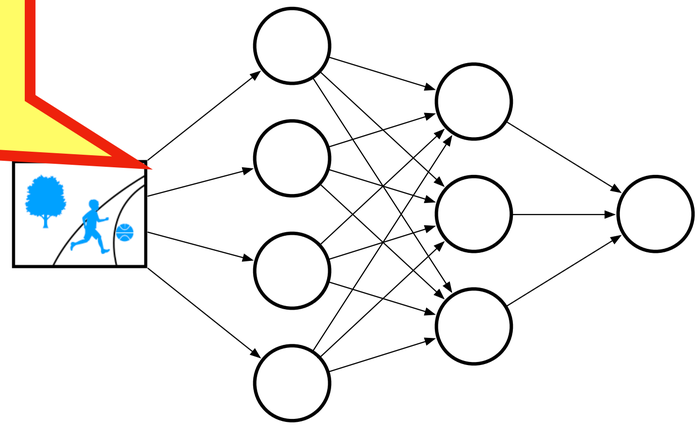
1. Response time
2. Accuracy
3. Throughput

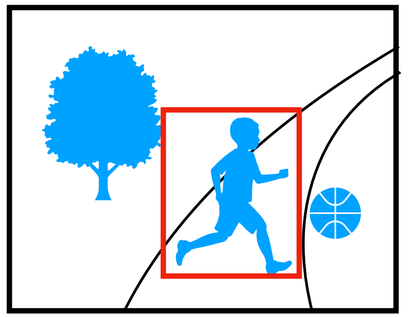
https://blogs.nvidia.com/blog/2016/01/04/automotive-nvidia-drive-px-2/



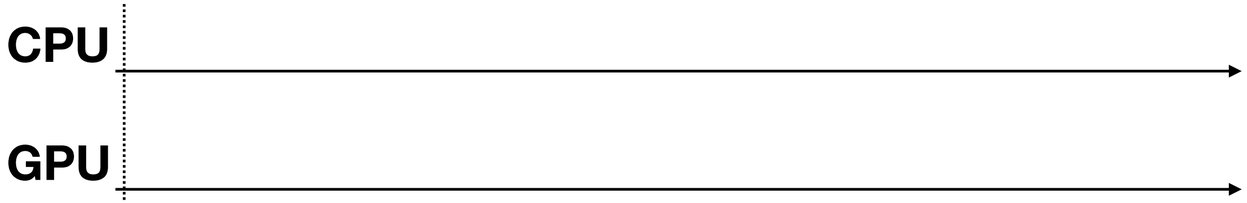
Hardware resources are **constrained and expensive**.

CNN software **underutilizes** the hardware.

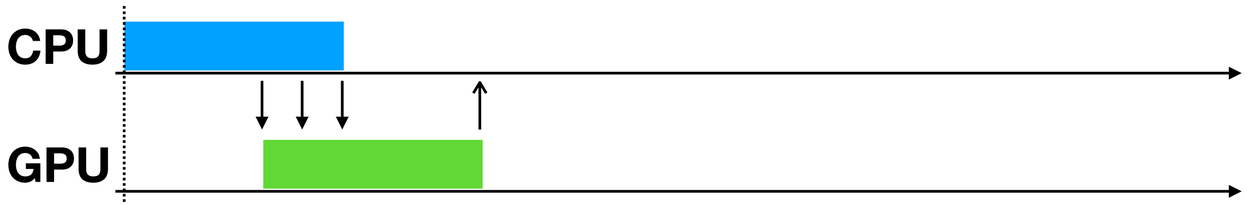




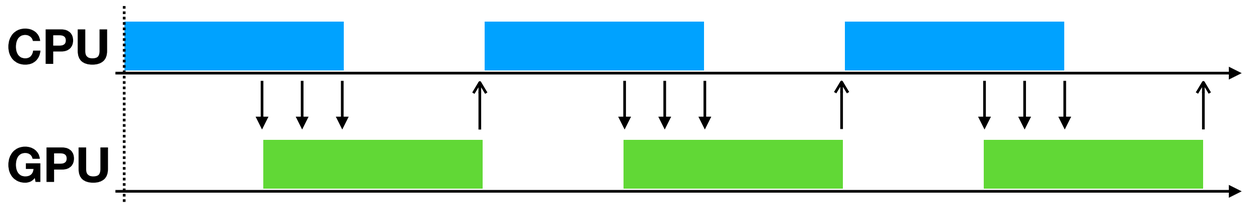
Current CNN frameworks



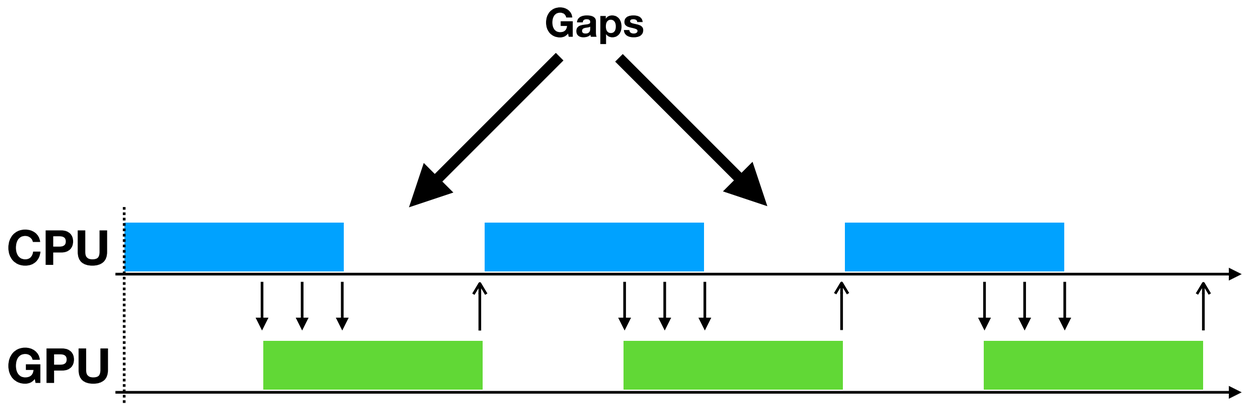
Current CNN frameworks



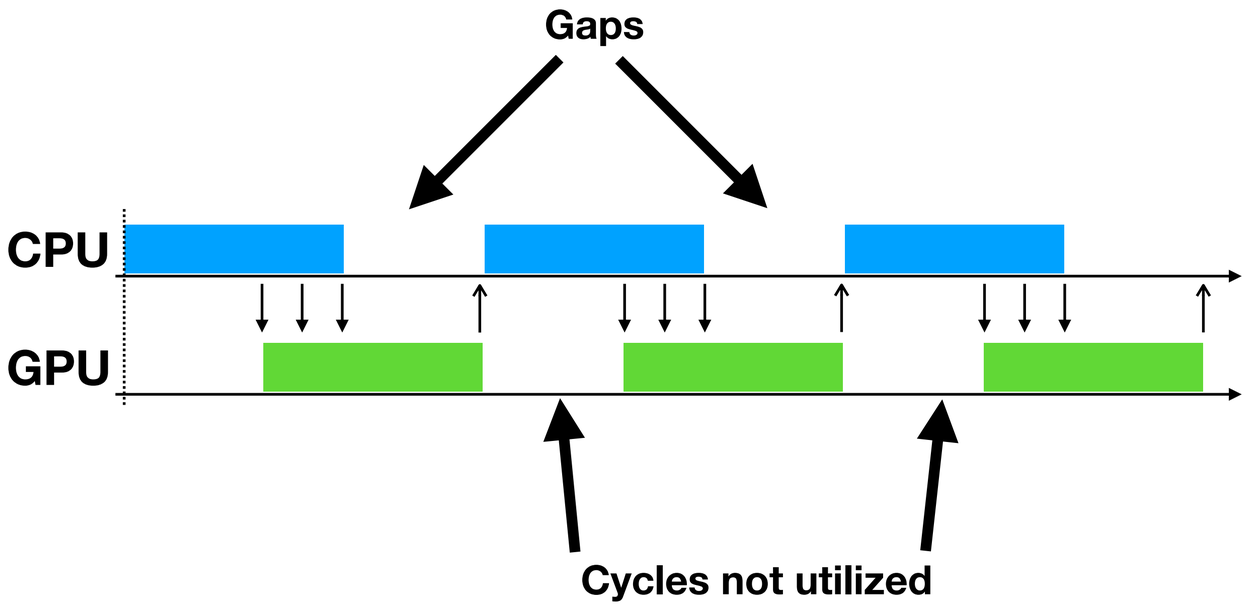
Current CNN frameworks



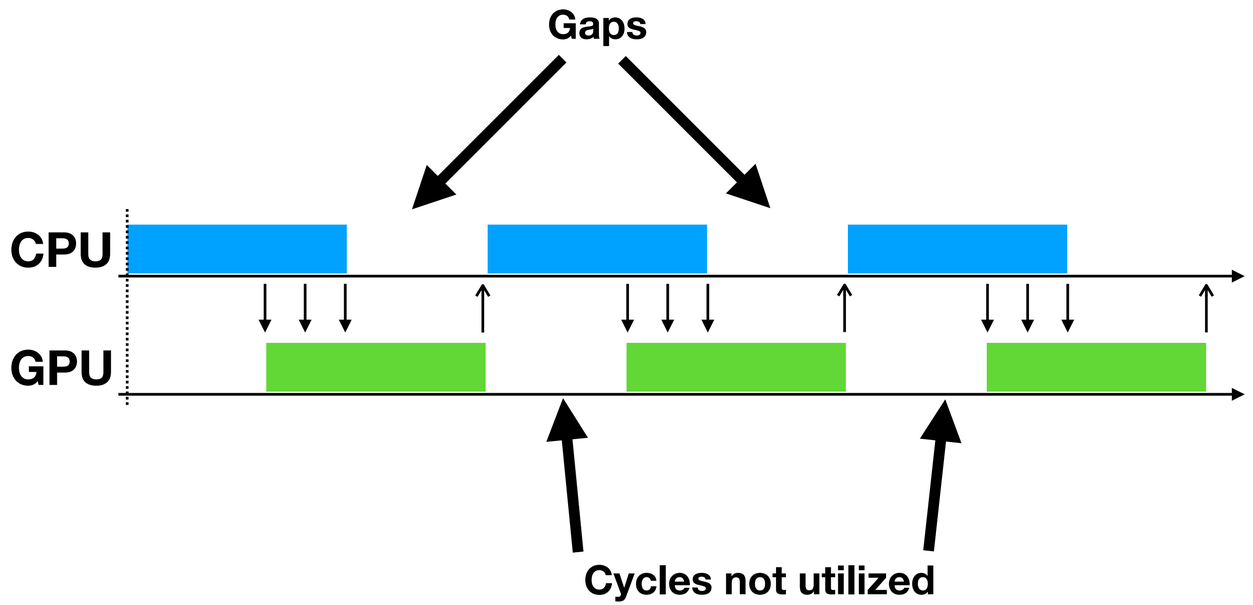
Current CNN frameworks



Current CNN frameworks

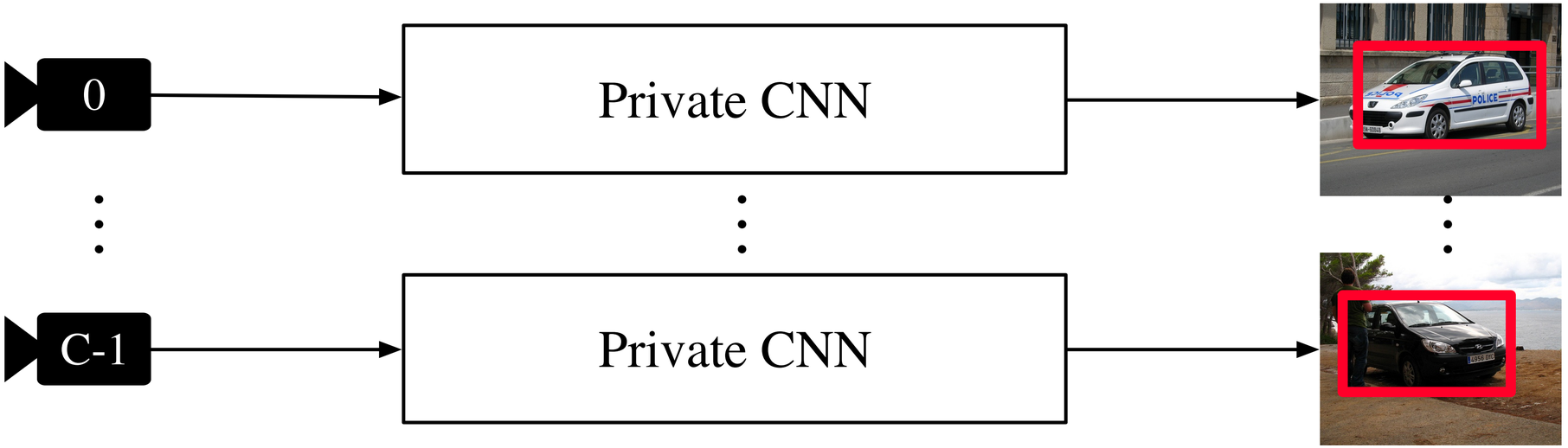


Current CNN frameworks

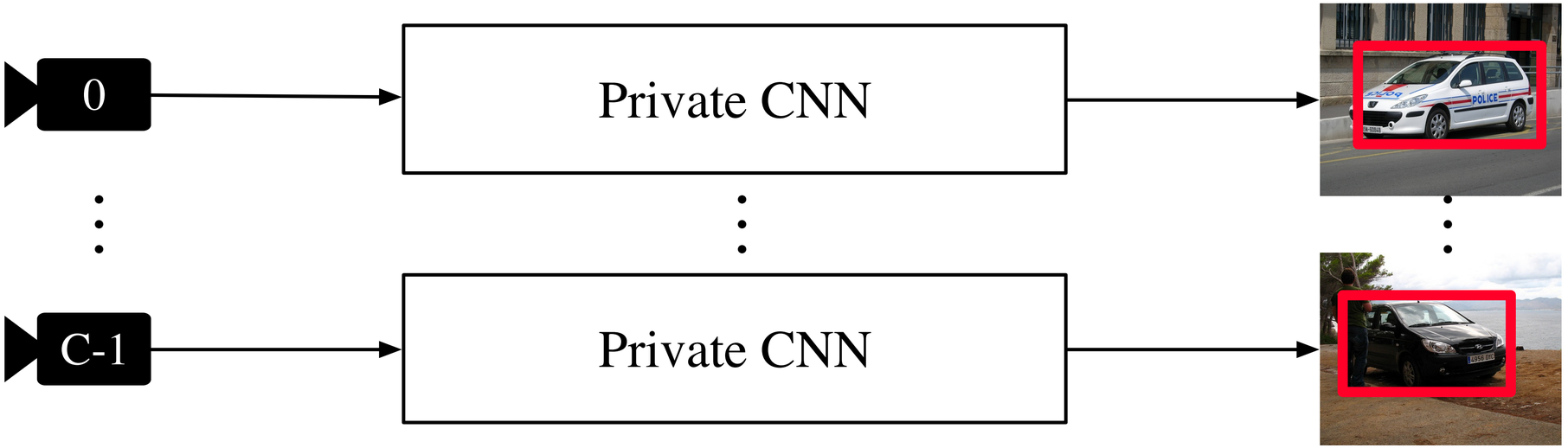


Single CNN underutilizes the hardware.

Traditional Multiple-Camera Processing Setup

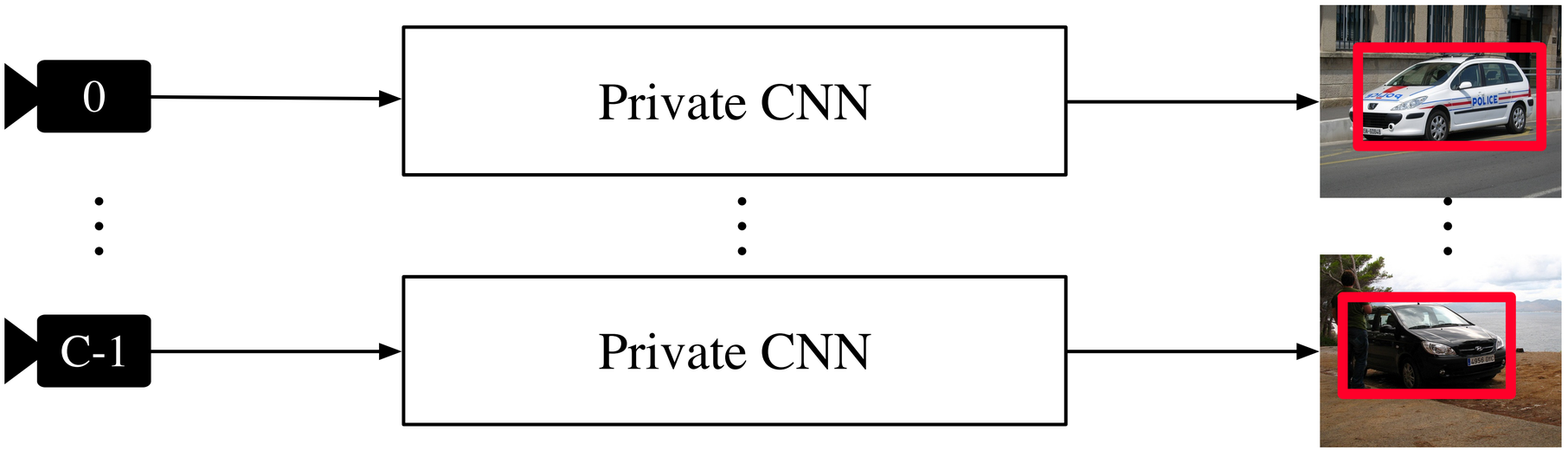


Traditional Multiple-Camera Processing Setup



Issues:

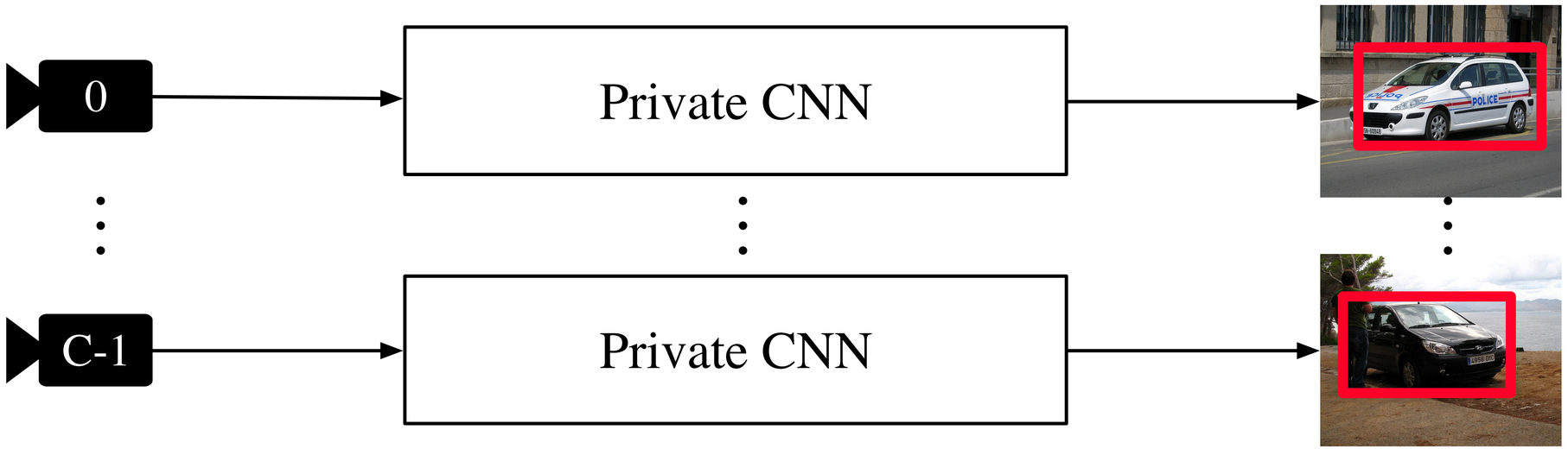
Traditional Multiple-Camera Processing Setup



Issues:

1. Memory requirements multiply, limiting the number of instances.

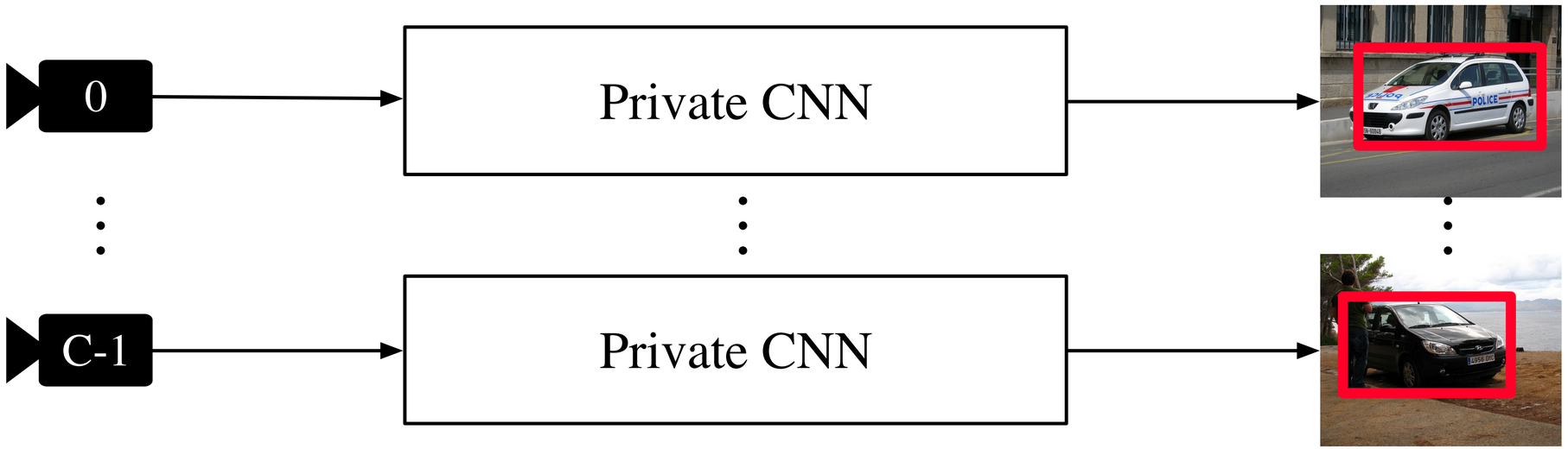
Traditional Multiple-Camera Processing Setup



Issues:

1. Memory requirements multiply, limiting the number of instances.
2. Context switches on GPU cause overheads.

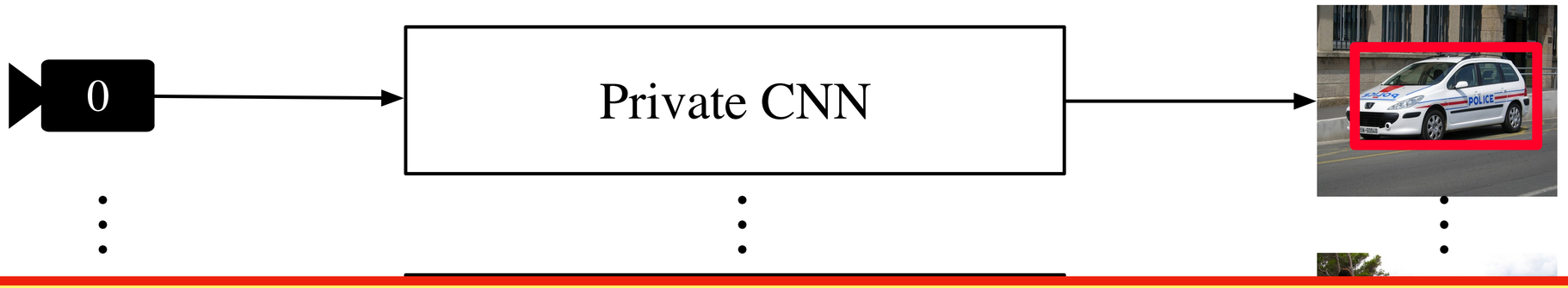
Traditional Multiple-Camera Processing Setup



Issues:

1. Memory requirements multiply, limiting the number of instances.
2. Context switches on GPU cause overheads.
3. Fast synchronization between cameras becomes harder.

Traditional Multiple-Camera Processing Setup



Parallelism through multi-process isn't helping.

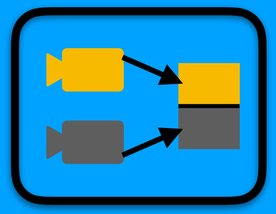
1. Memory requirements multiply, limiting the number of instances.
2. Context switches on GPU cause overheads.
3. Fast synchronization between cameras becomes harder.

Proposed Solutions

Part I:

Parallel Execution for CNN frameworks

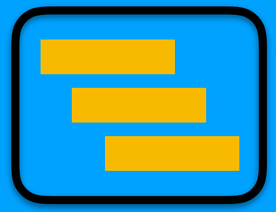
Part II:



Multi-camera Composite Images to provide high throughput for multiple cameras.

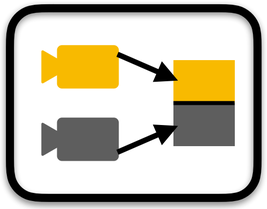
Proposed Solutions

Part I:



Parallel Execution for CNN frameworks

Part II:



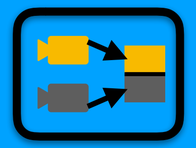
Multi-camera Composite Images to provide high throughput for multiple cameras.

Let’s re-think the design of CNN frameworks

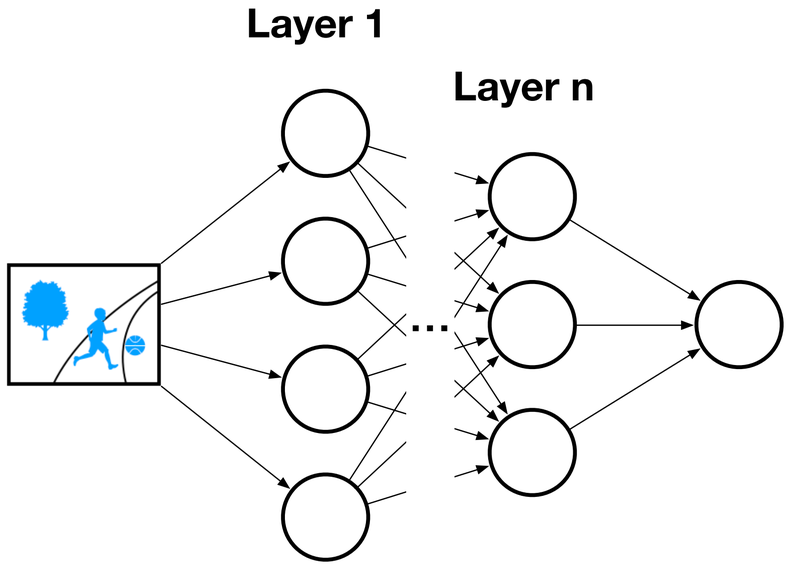
Part I:

Parallel Execution

Part II:

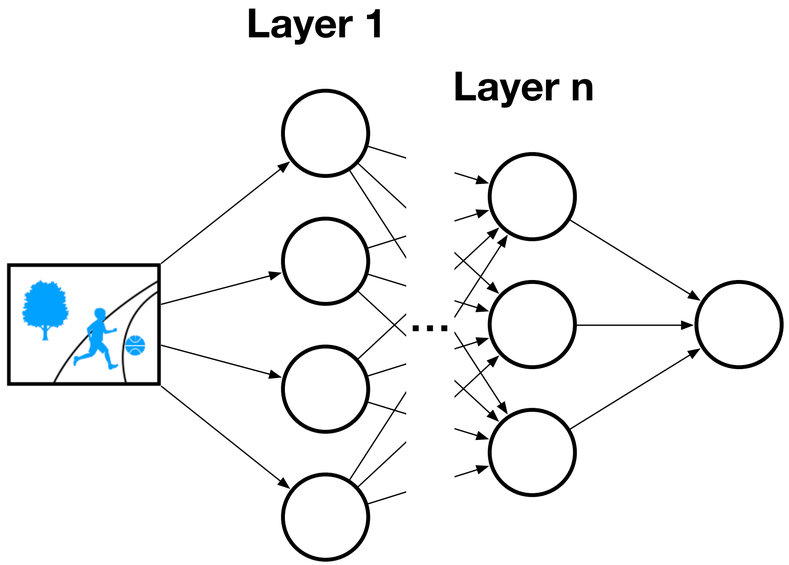


Multi-camera Composite Images



Let’s re-think the design of CNN frameworks

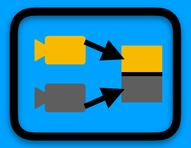
- CNN models are graphs of **layers**.



Part I:

Parallel Execution

Part II:



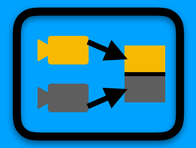
Multi-camera Composite Images

Let’s re-think the design of CNN frameworks

Part I:

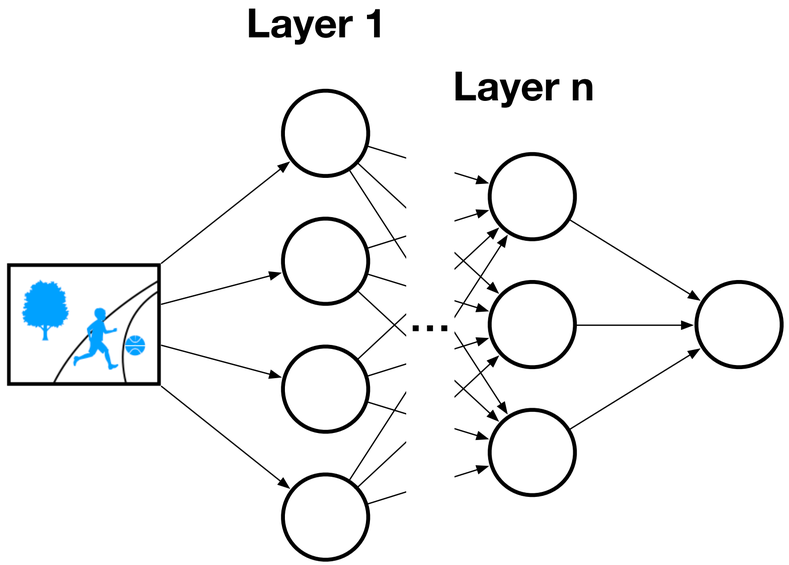
Parallel Execution

Part II:



Multi-camera Composite Images

- CNN models are graphs of **layers**.
- Processing of images can be **independent**, e.g., object detection.

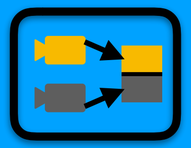


Let’s re-think the design of CNN frameworks

Part I:

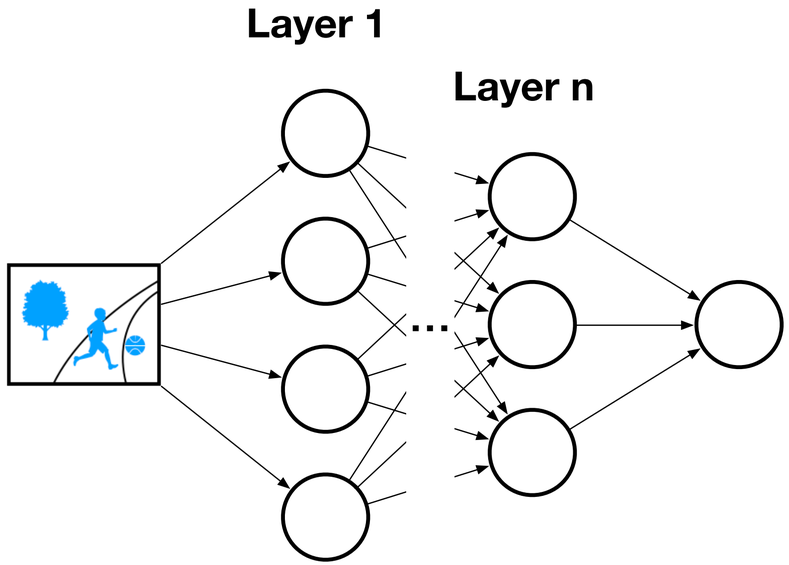
Parallel Execution

Part II:

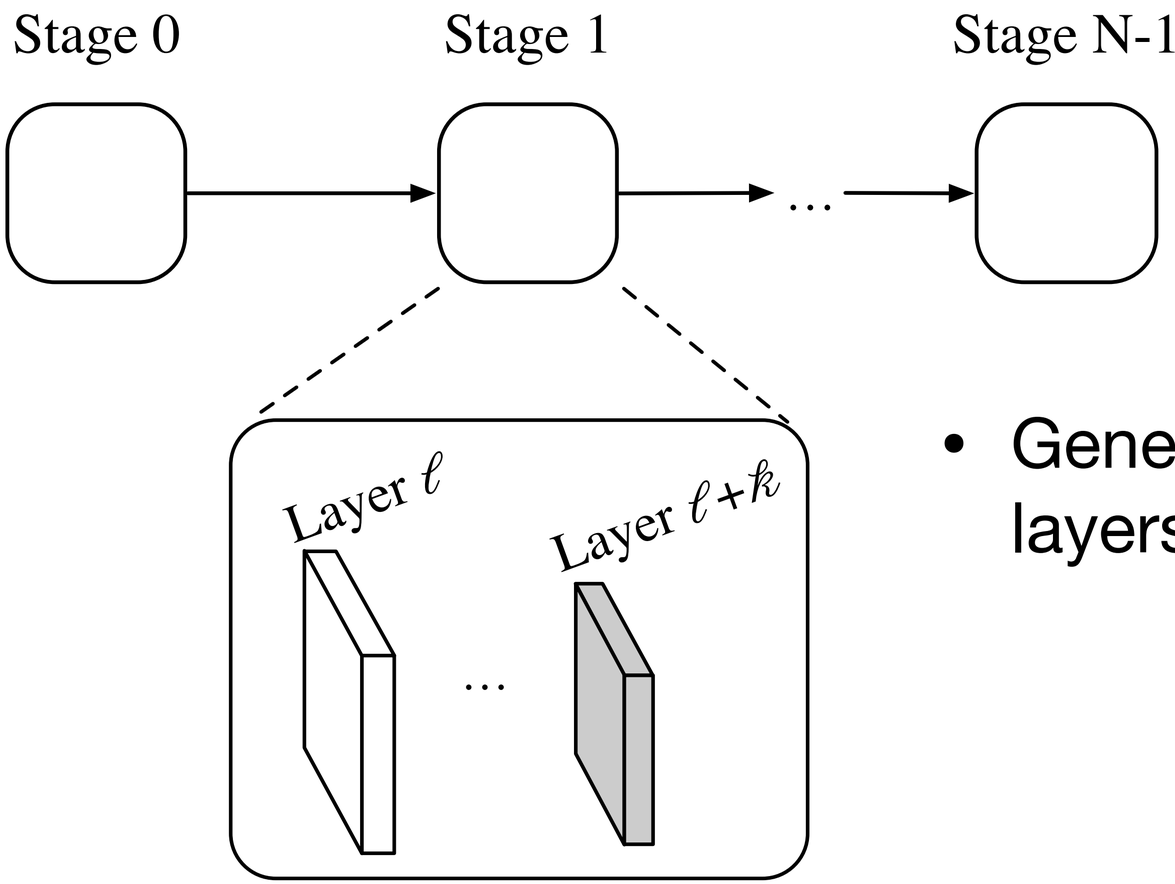


Multi-camera Composite Images

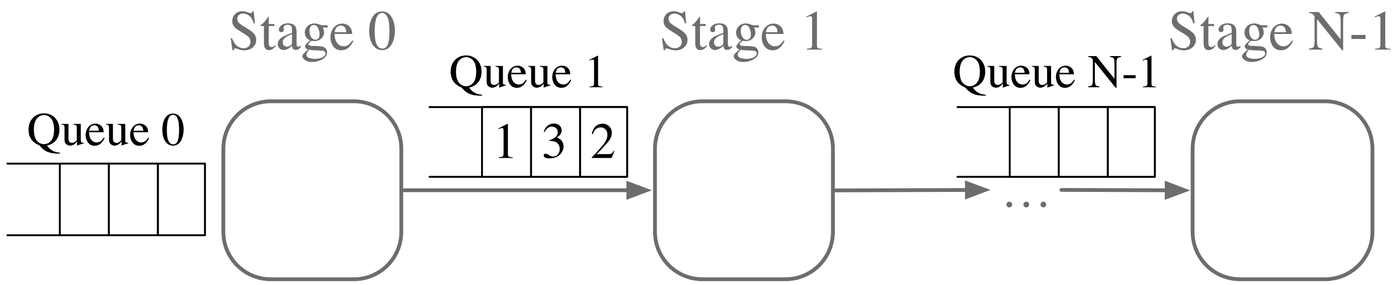
- CNN models are graphs of **layers**.
- Processing of images can be **independent**, e.g., object detection.

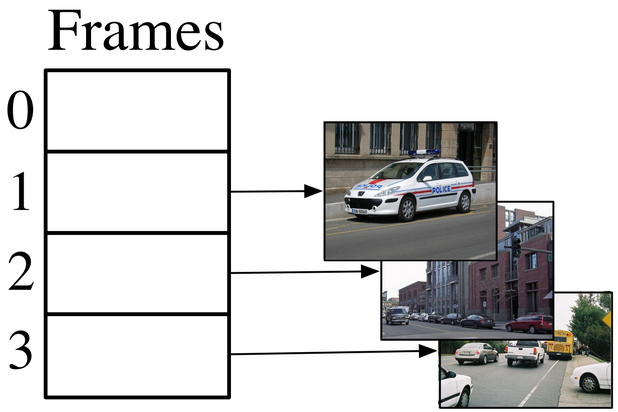


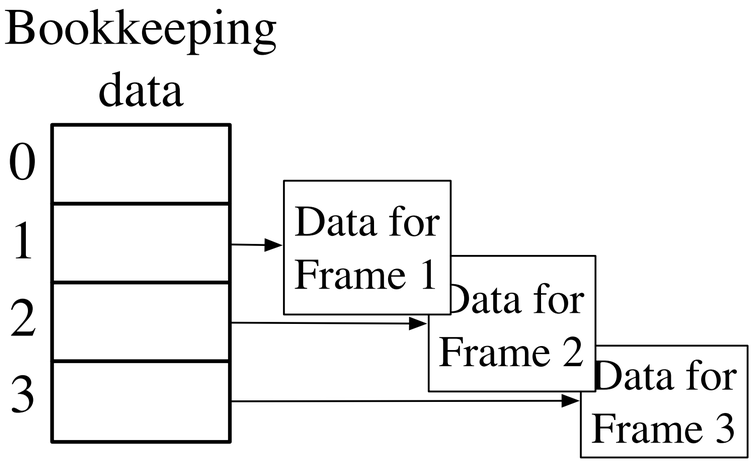
We enable ***parallel execution*** for CNN frameworks and ***shared CNN*** for multiple cameras.



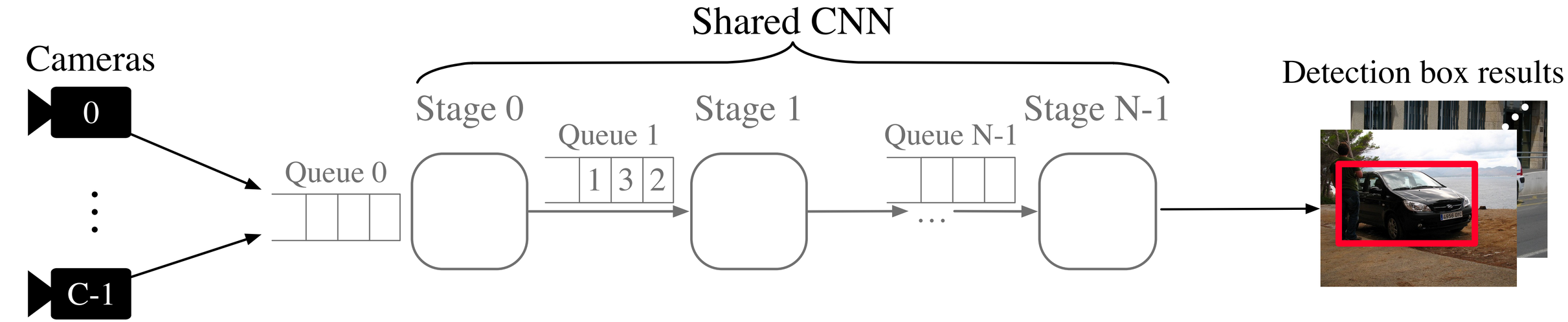
- Generalize concept of layers into **stages**

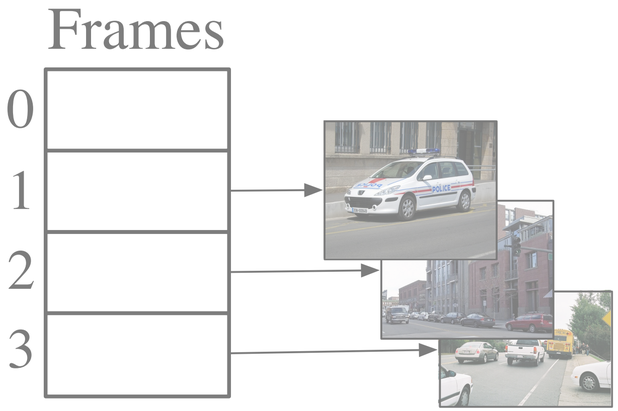


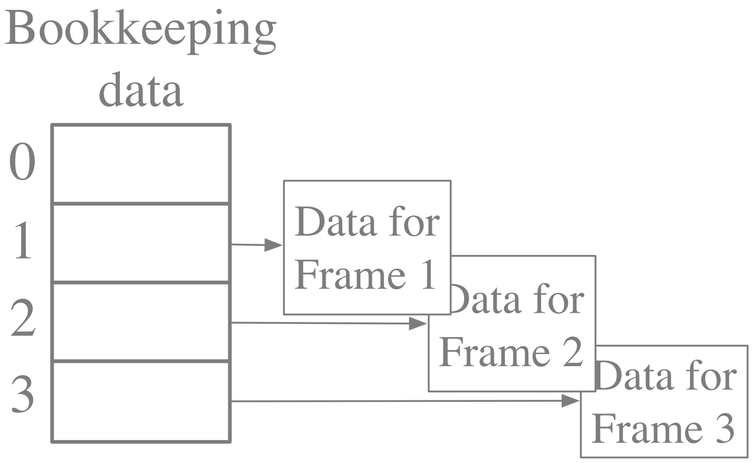




- Generalize concept of layers into **stages**
- Communicate data between stages using **PGM^RT** (a processing graph management tool)







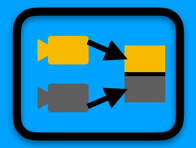
- Generalize concept of layers into **stages**
- Communicate data between stages using **PGM^{RT}** (a processing graph management tool)
- **Share CNN** among multiple cameras

Different Execution Methods

Part I:

Parallel Execution

Part II:



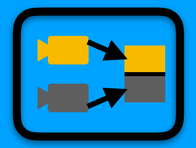
Multi-camera Composite Images

Different Execution Methods

Part I:

Parallel Execution

Part II:



Multi-camera Composite Images

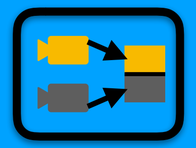
SERIAL	***private CNN*** in one process

Different Execution Methods

Part I:

Parallel Execution

Part II:



Multi-camera Composite Images

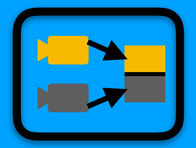
SERIAL	***private CNN*** in one process
PIPELINE	***shared CNN*** that has ***one*** thread per stage

Different Execution Methods

Part I:

Parallel Execution

Part II:



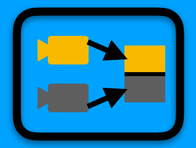
Multi-camera Composite Images

SERIAL	***private CNN*** in one process
PIPELINE	***shared CNN*** that has ***one*** thread per stage
PARALLEL	***shared CNN*** that has ***multiple*** threads per stage

Part I:

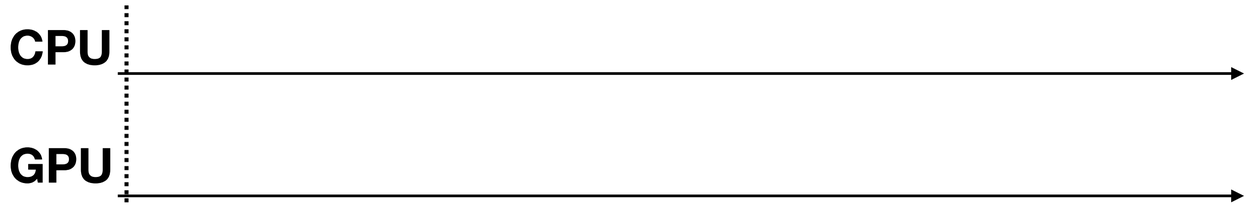
Parallel Execution

Part II:

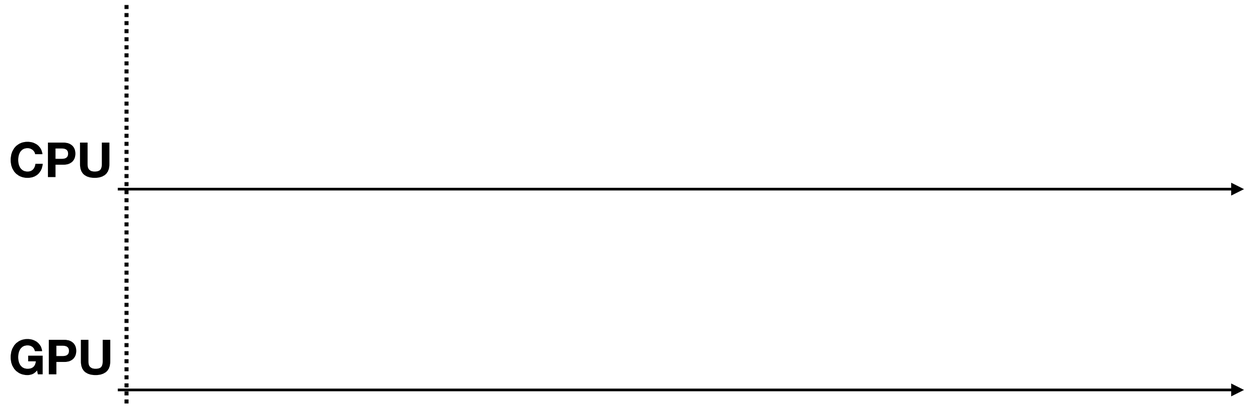


Multi-camera Composite Images

SERIAL



PIPELINE

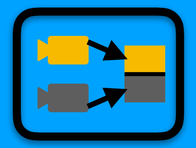


PARALLEL

Part I:

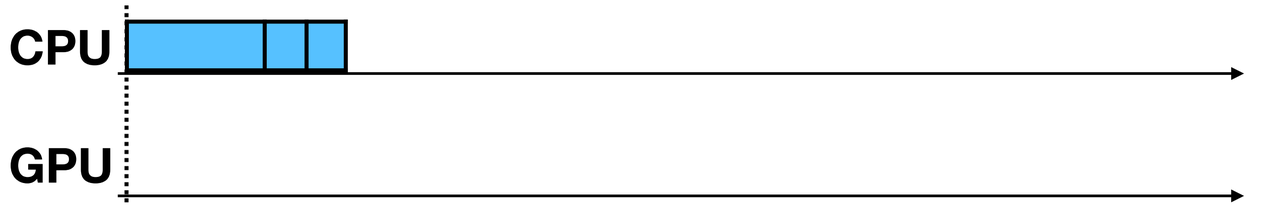
Parallel Execution

Part II:

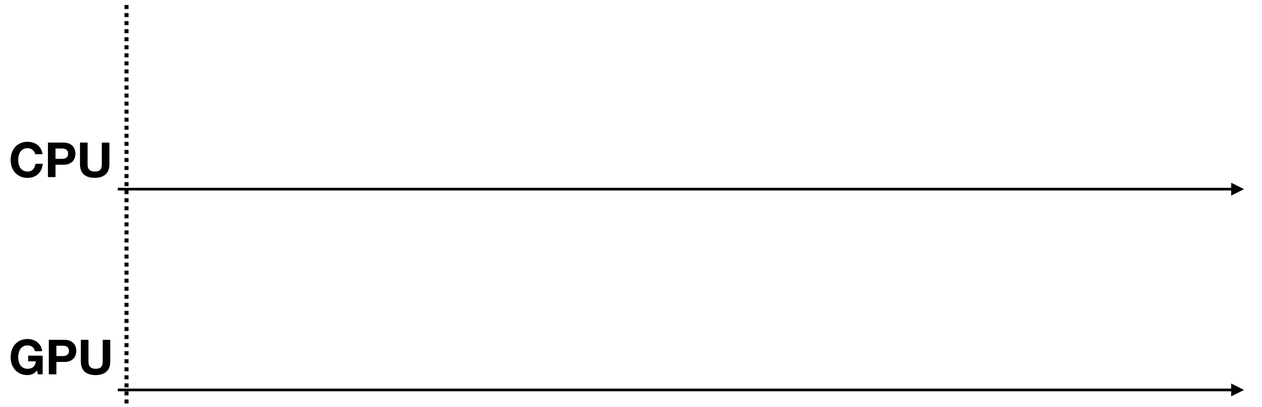


Multi-camera Composite Images

SERIAL



PIPELINE

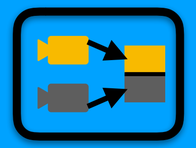


PARALLEL

Part I:

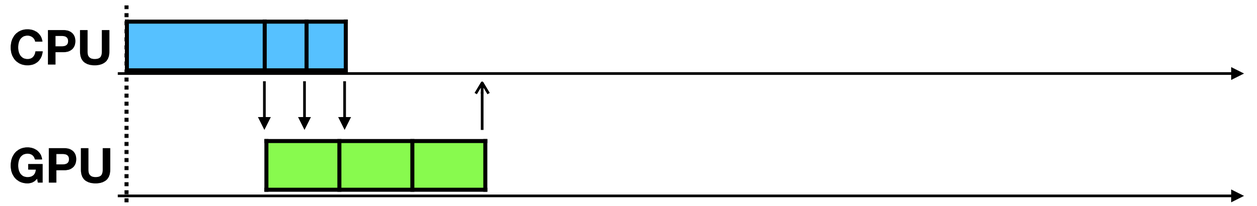
Parallel Execution

Part II:

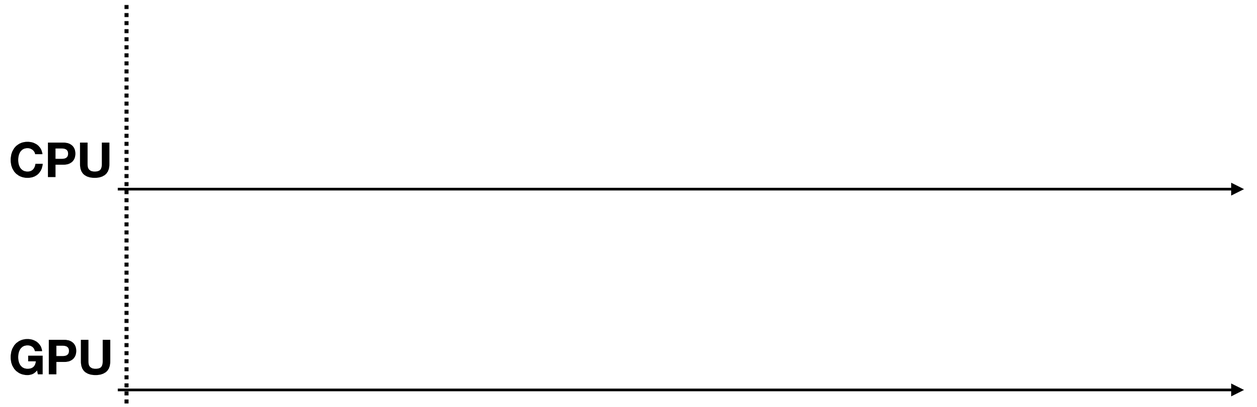


Multi-camera Composite Images

SERIAL



PIPELINE

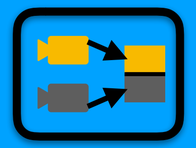


PARALLEL

Part I:

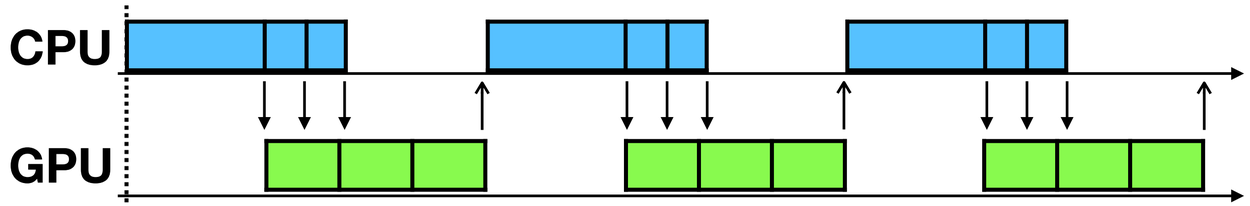
Parallel Execution

Part II:

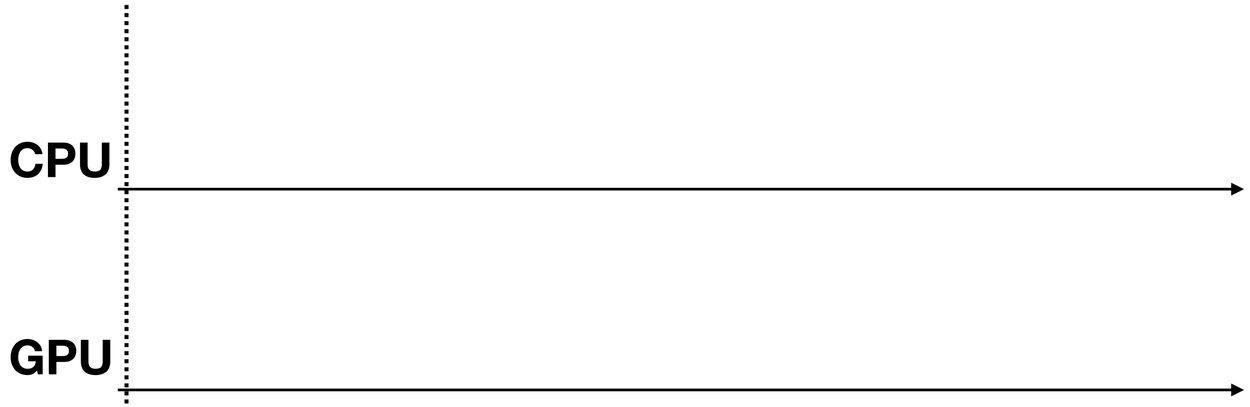


Multi-camera Composite Images

SERIAL



PIPELINE

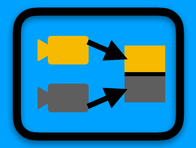


PARALLEL

Part I:

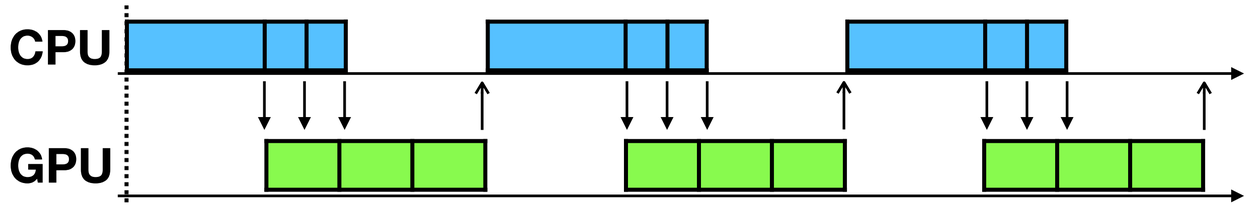
Parallel Execution

Part II:

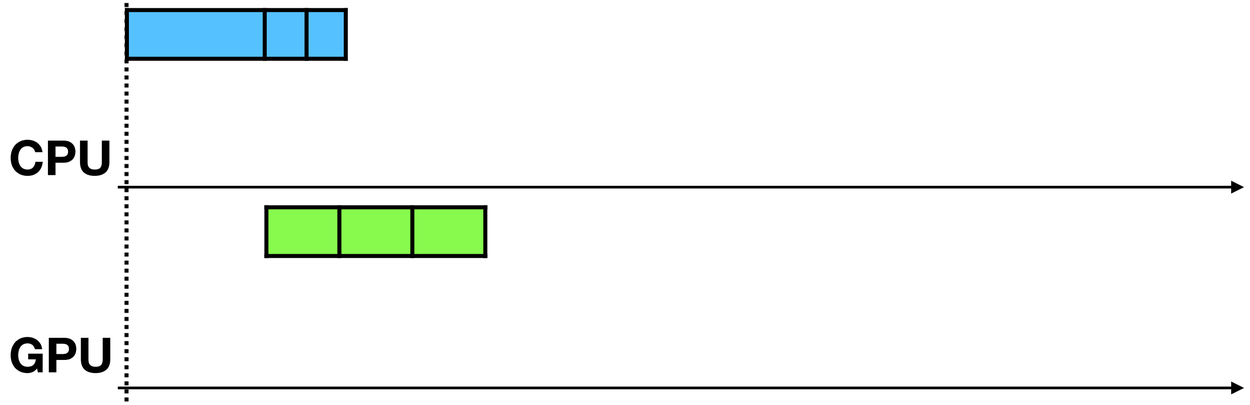


Multi-camera Composite Images

SERIAL



PIPELINE

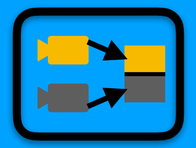


PARALLEL

Part I:

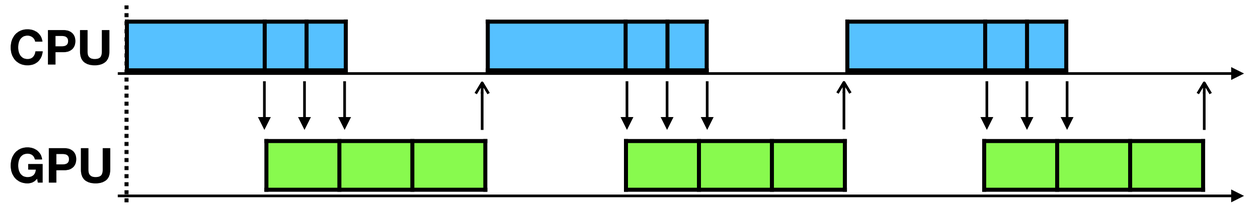
Parallel Execution

Part II:

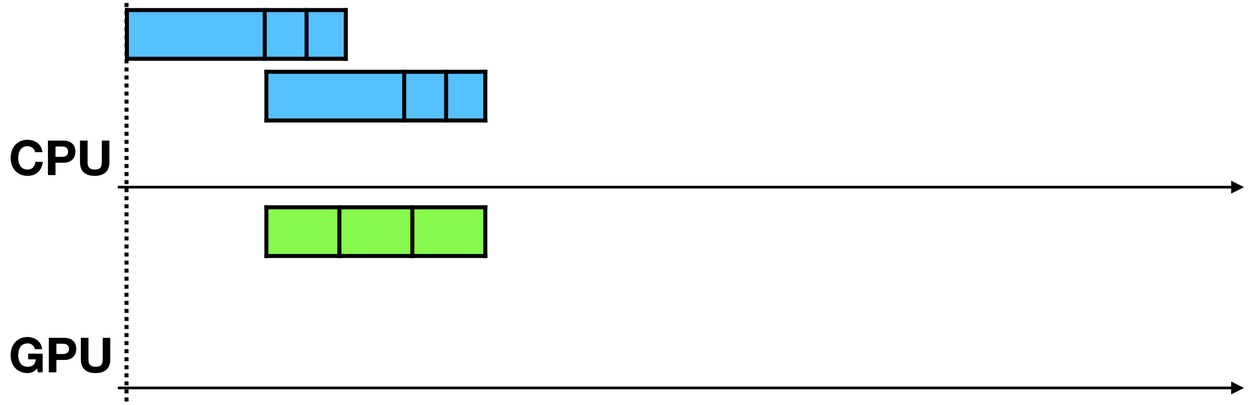


Multi-camera Composite Images

SERIAL



PIPELINE

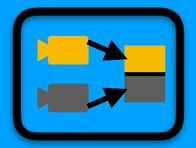


PARALLEL

Part I:

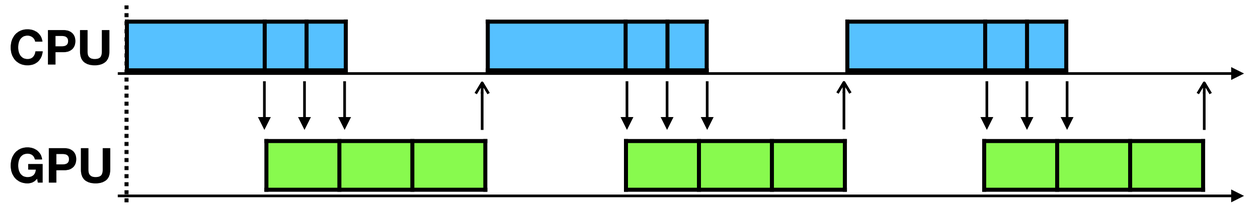
Parallel Execution

Part II:

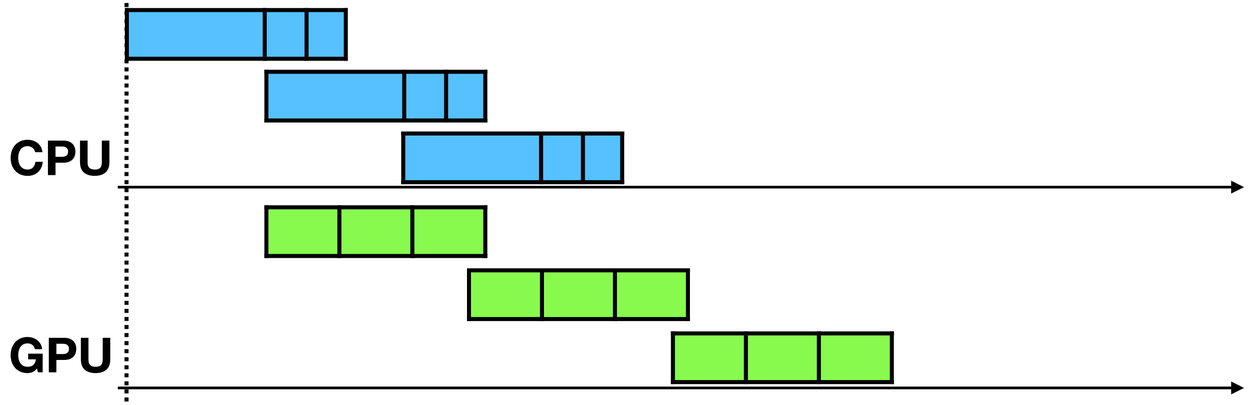


Multi-camera Composite Images

SERIAL



PIPELINE

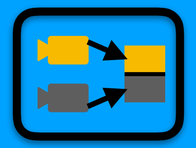


PARALLEL

Part I:

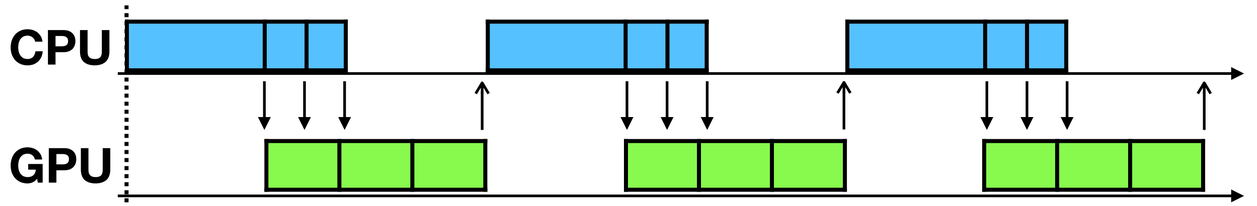
Parallel Execution

Part II:

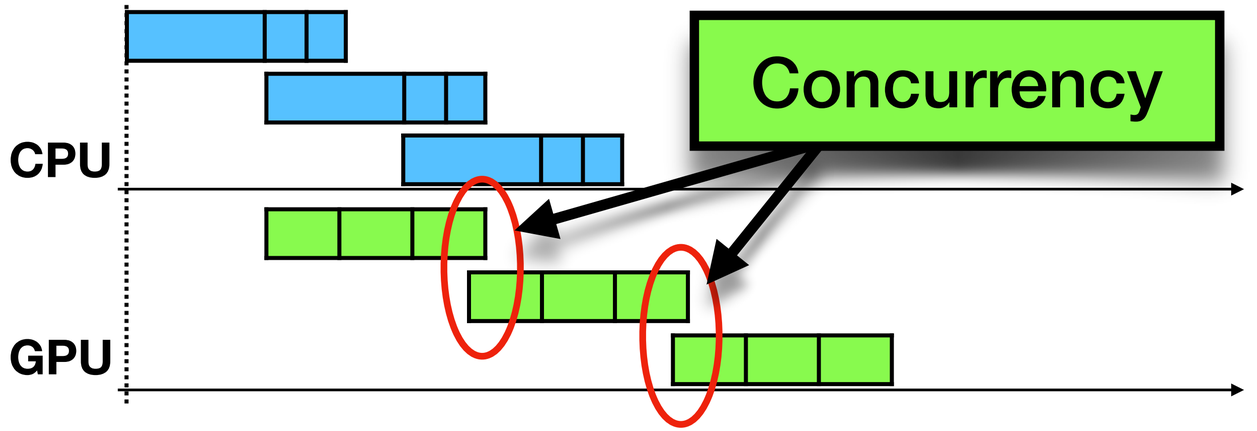


Multi-camera Composite Images

SERIAL



PIPELINE

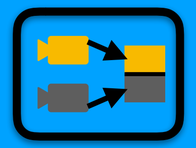


PARALLEL

Part I:

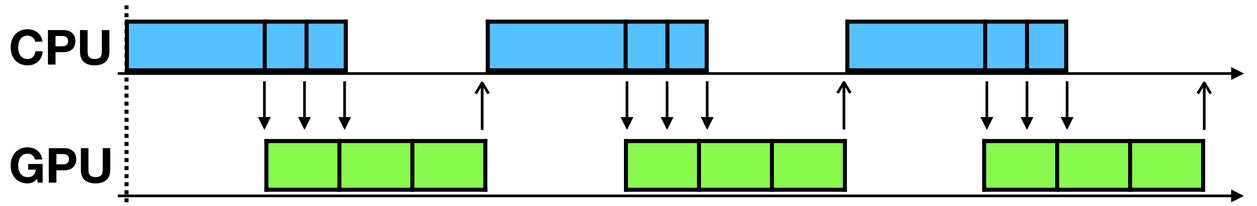
Parallel Execution

Part II:

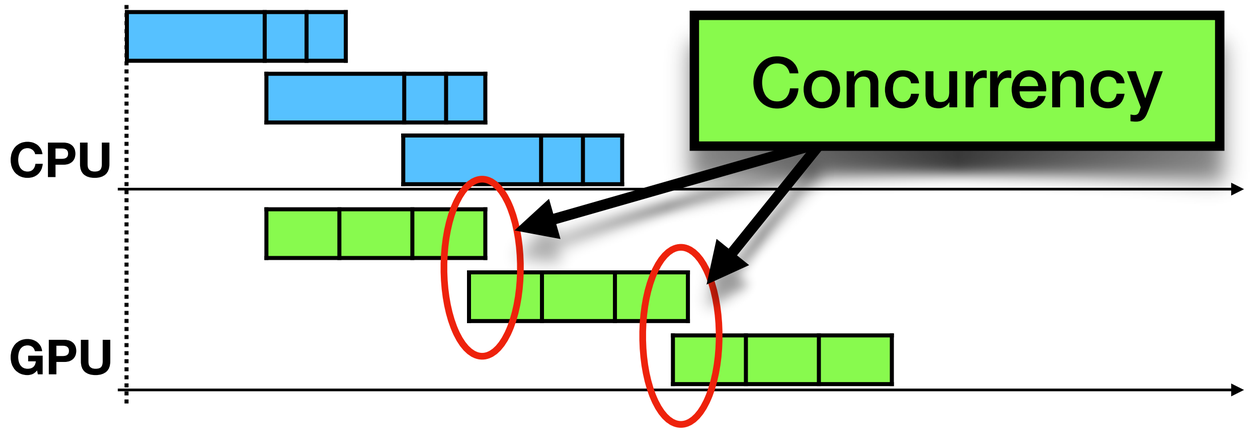


Multi-camera Composite Images

SERIAL



PIPELINE

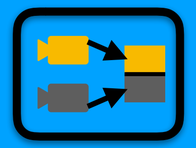


PARALLEL

Part I:

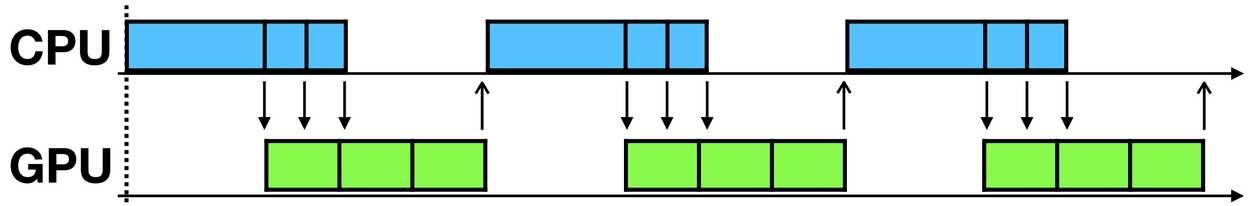
Parallel Execution

Part II:

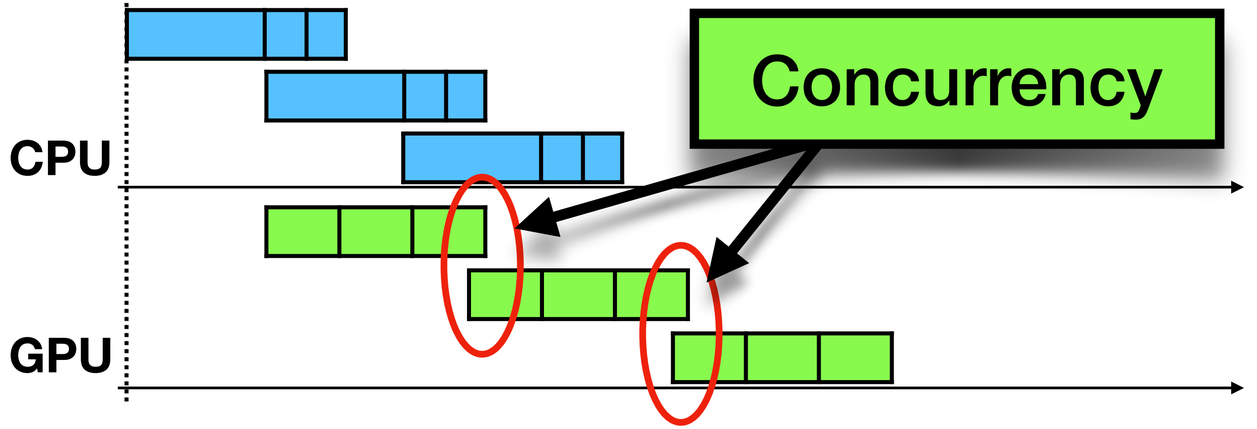


Multi-camera Composite Images

SERIAL



PIPELINE

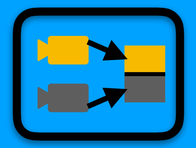


PARALLEL

Part I:

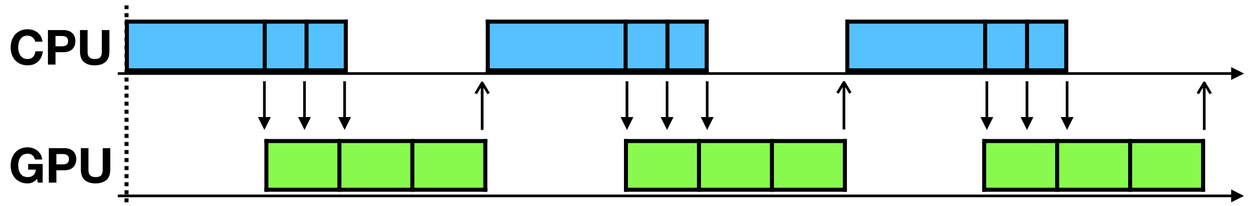
Parallel Execution

Part II:

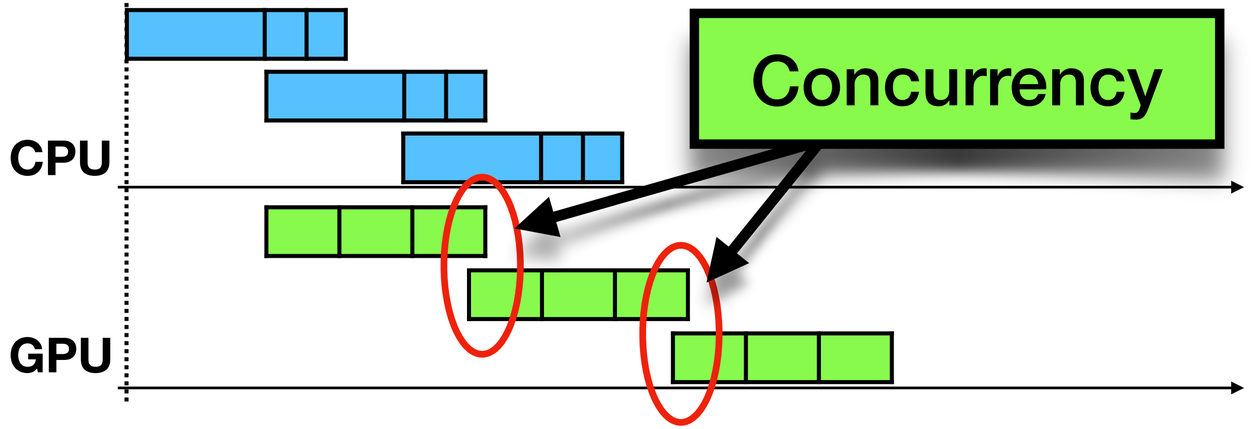


Multi-camera Composite Images

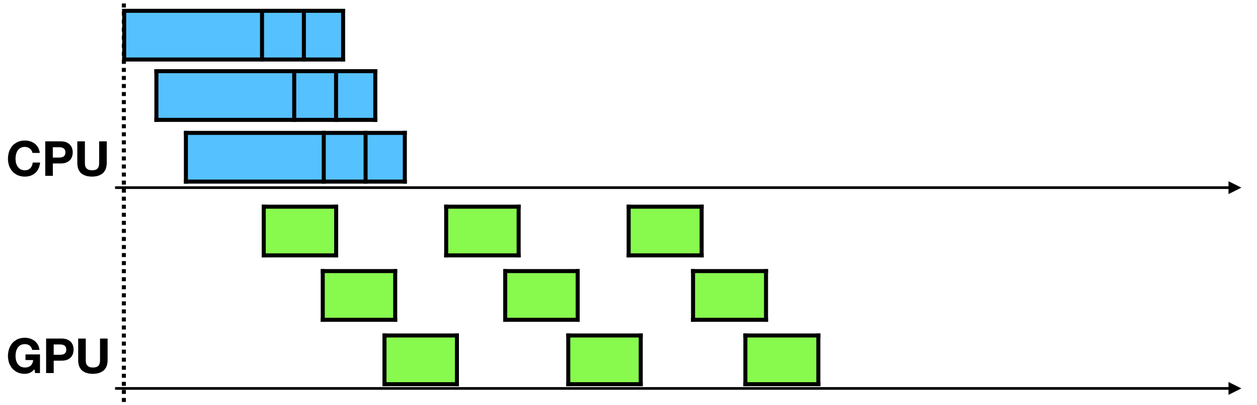
SERIAL



PIPELINE



PARALLEL



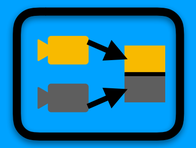
Evaluation

- We compared **latency** and **throughput** between
  - **SERIAL**
  - **SERIAL x6**
  - **PIPELINE**
  - **PARALLEL**
- With CNN model **Tiny YOLOv2** on CNN framework **Darknet**
- On hardware platform: **NVIDIA Drive PX 2.**

Part I:

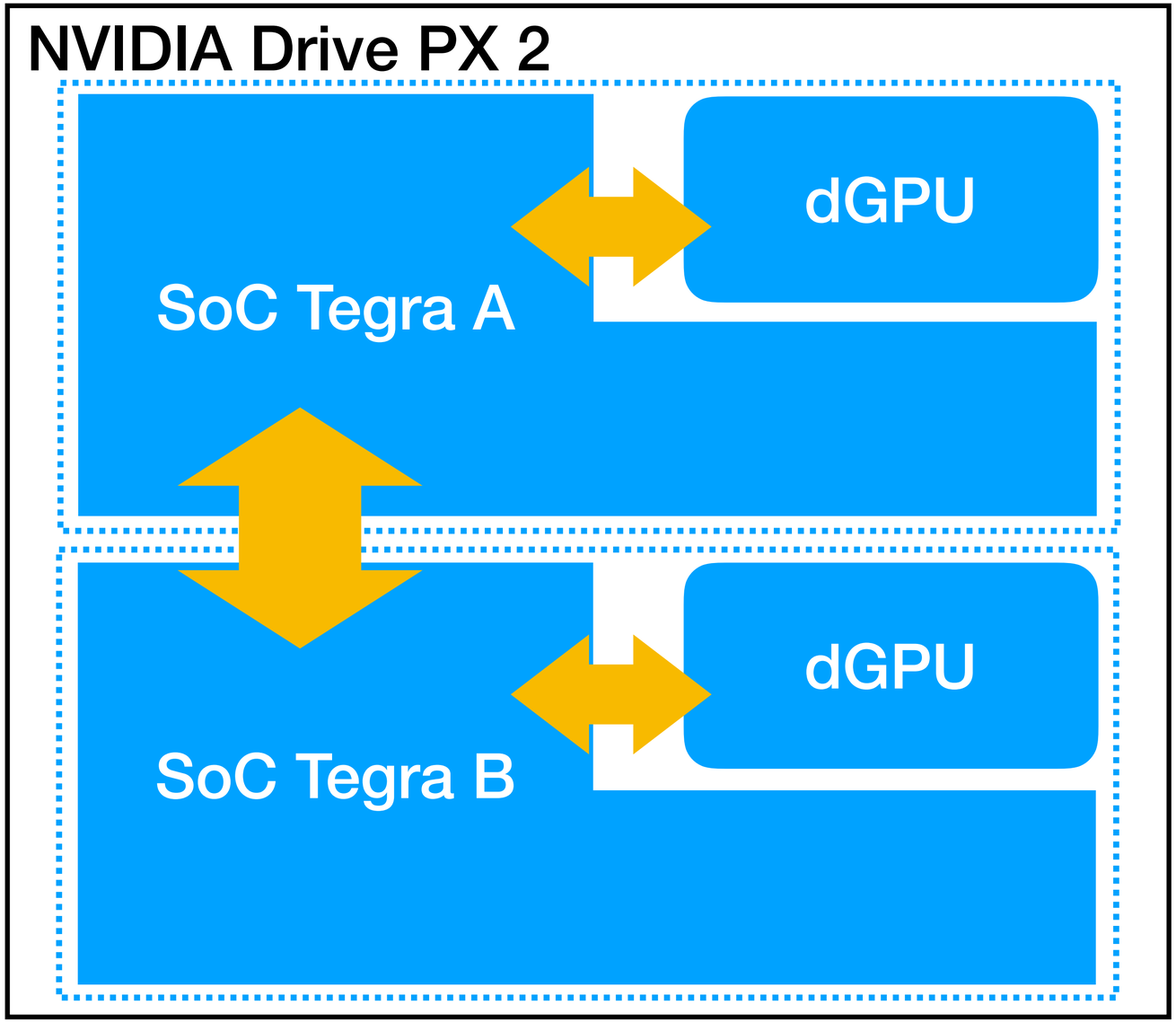
Parallel Execution

Part II:



Multi-camera Composite Images

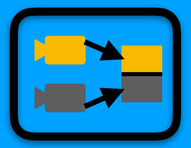
Evaluation (Hardware)



Part I:

Parallel Execution

Part II:



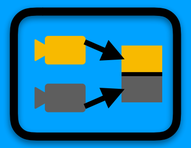
Multi-camera Composite Images

Evaluation (Hardware)

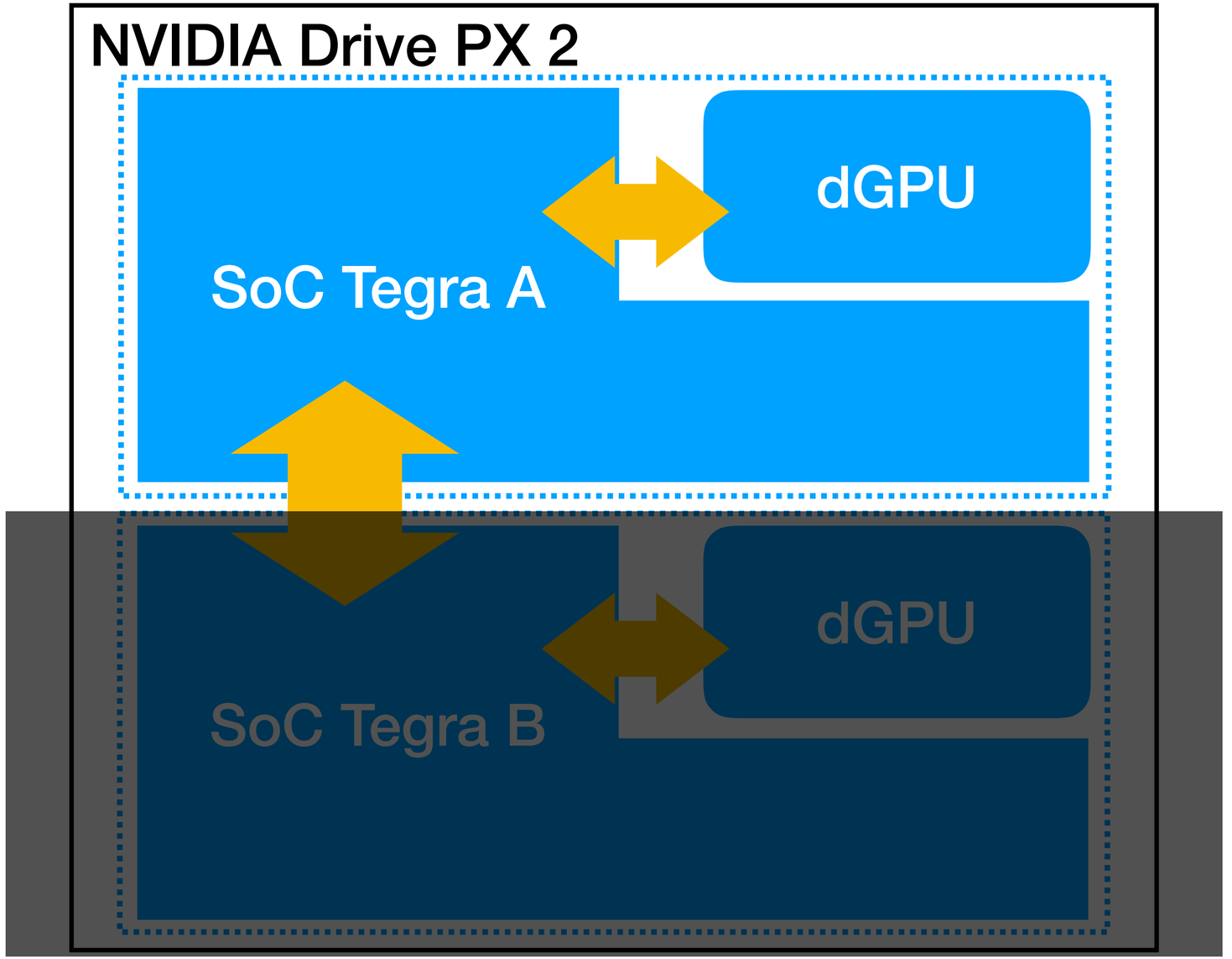
Part I:

Parallel Execution

Part II:



Multi-camera Composite Images

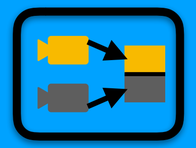


Evaluation Results

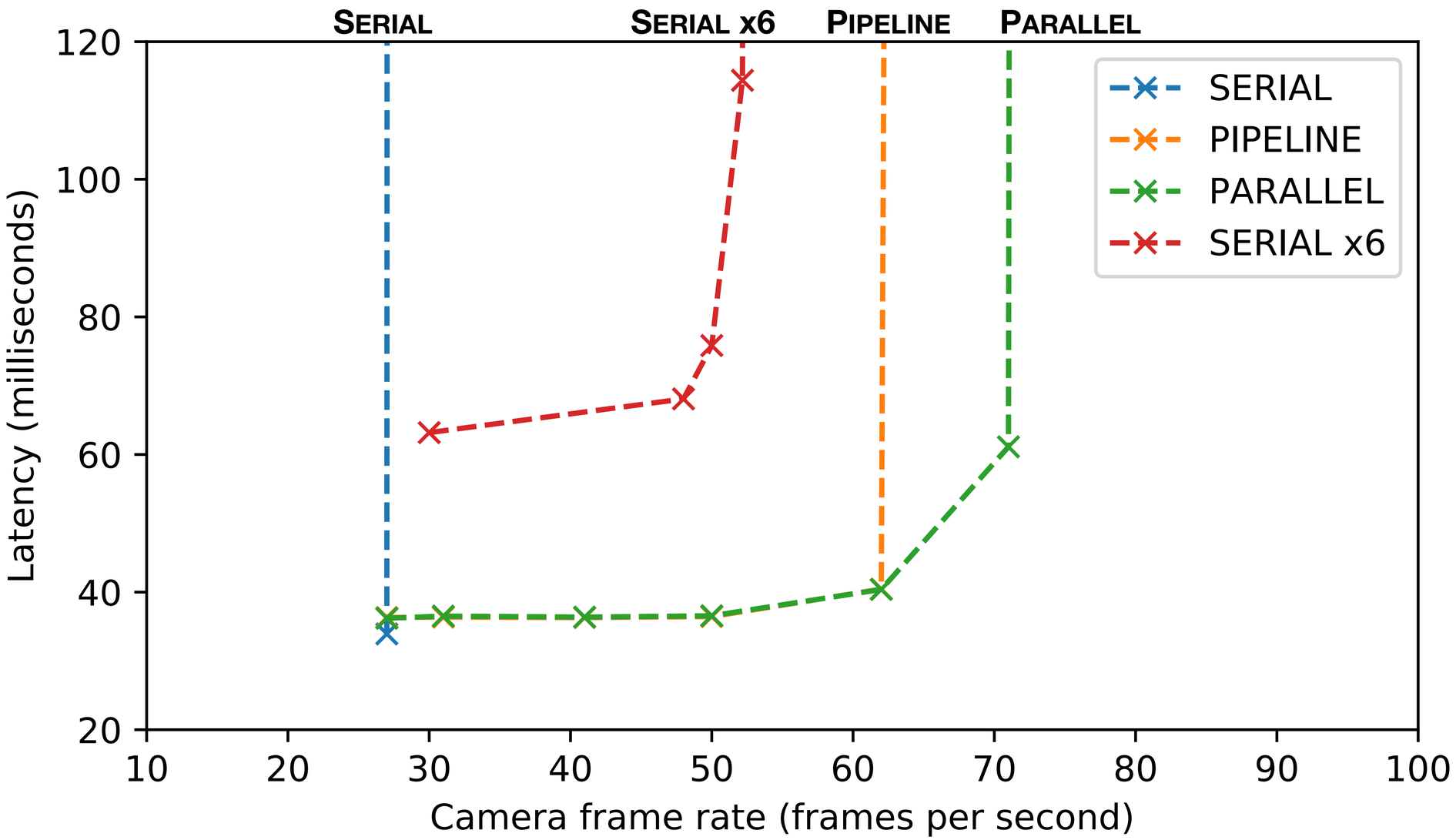
Part I:

Parallel Execution

Part II:



Multi-camera Composite Images

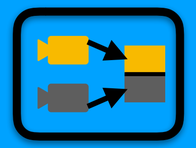


Evaluation Results

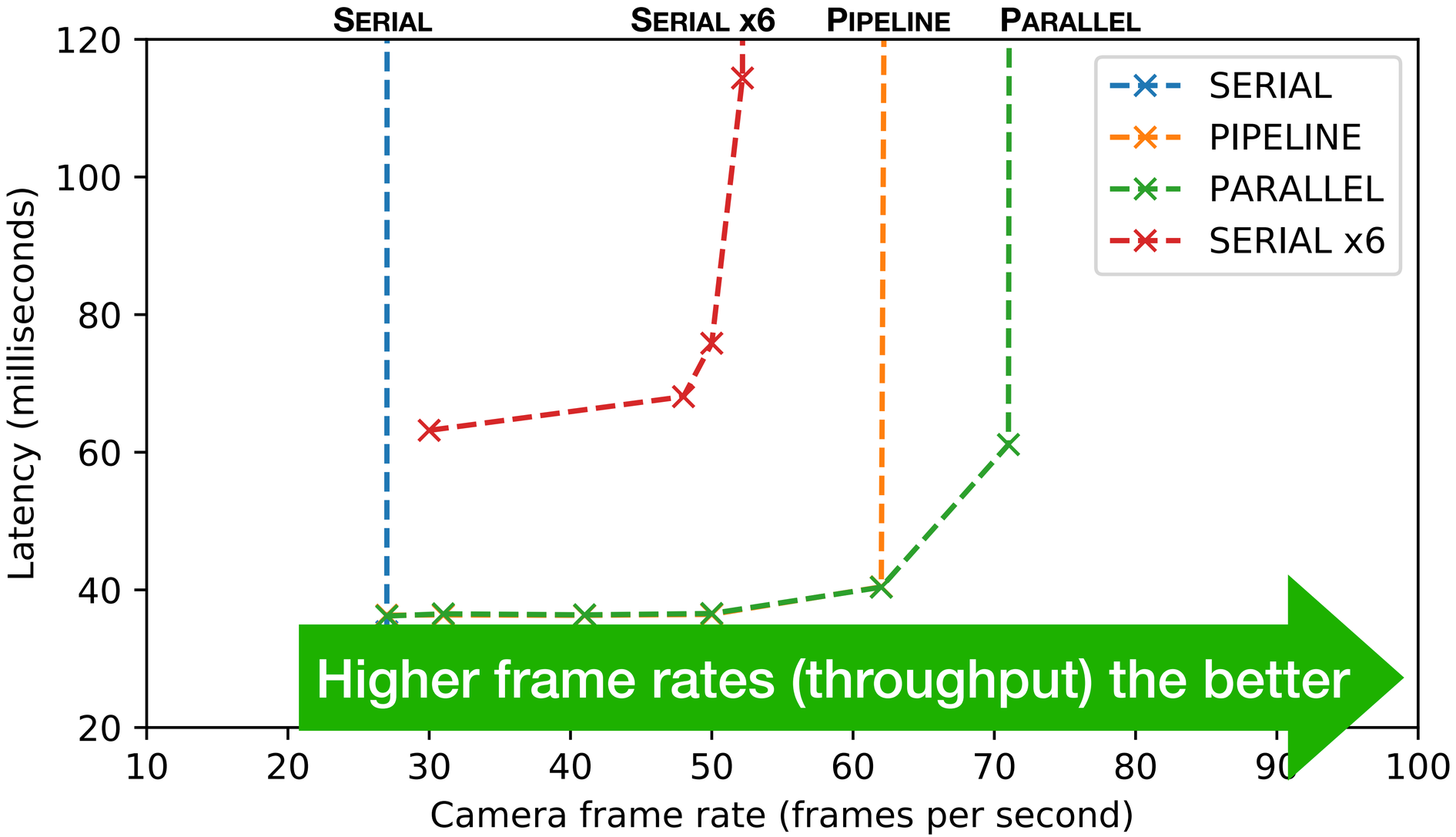
Part I:

Parallel Execution

Part II:



Multi-camera Composite Images

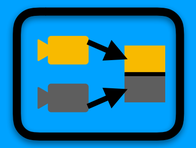


Evaluation Results

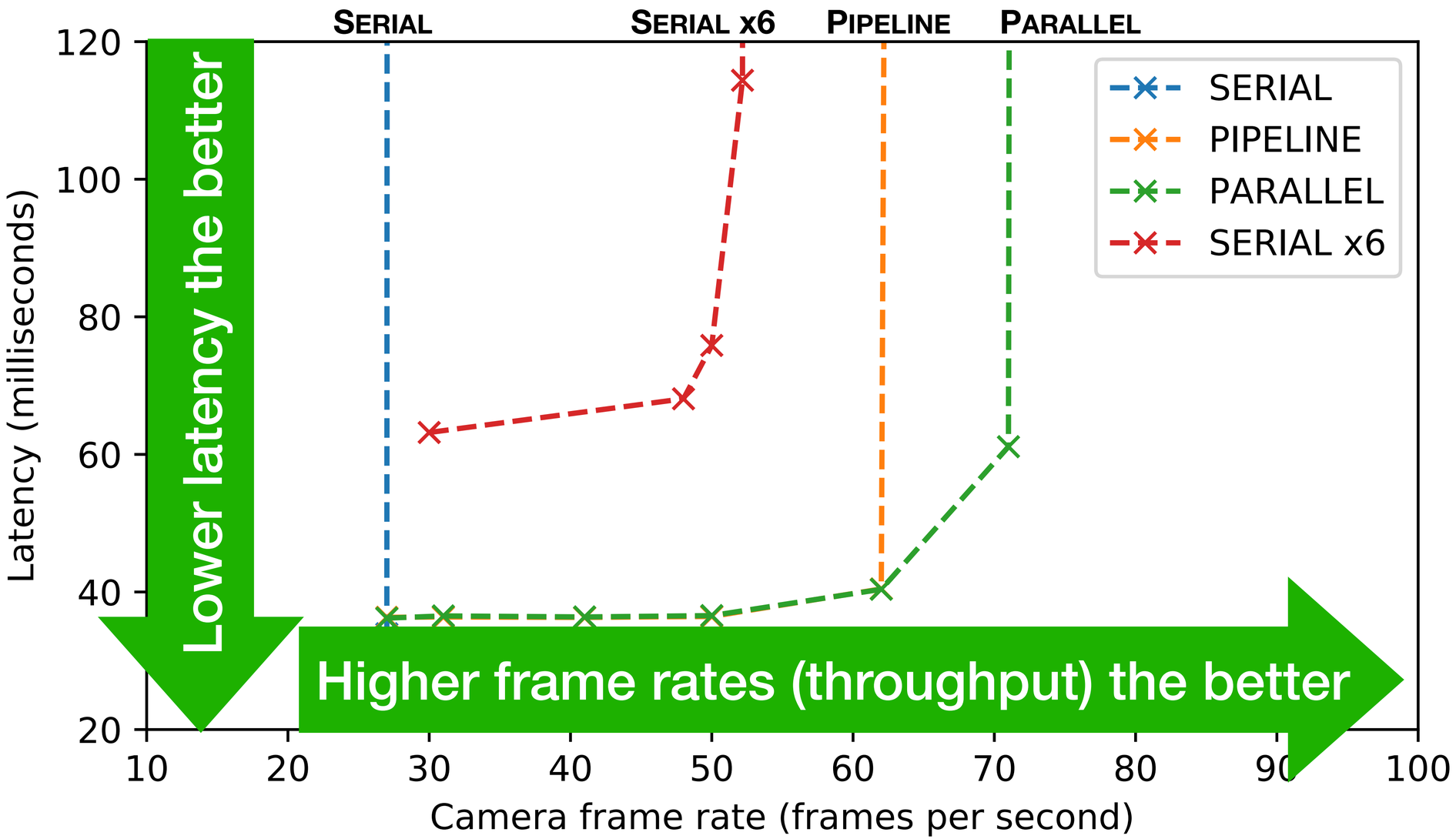
Part I:

Parallel Execution

Part II:



Multi-camera Composite Images

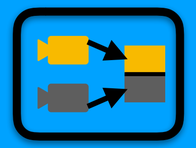


Evaluation Results

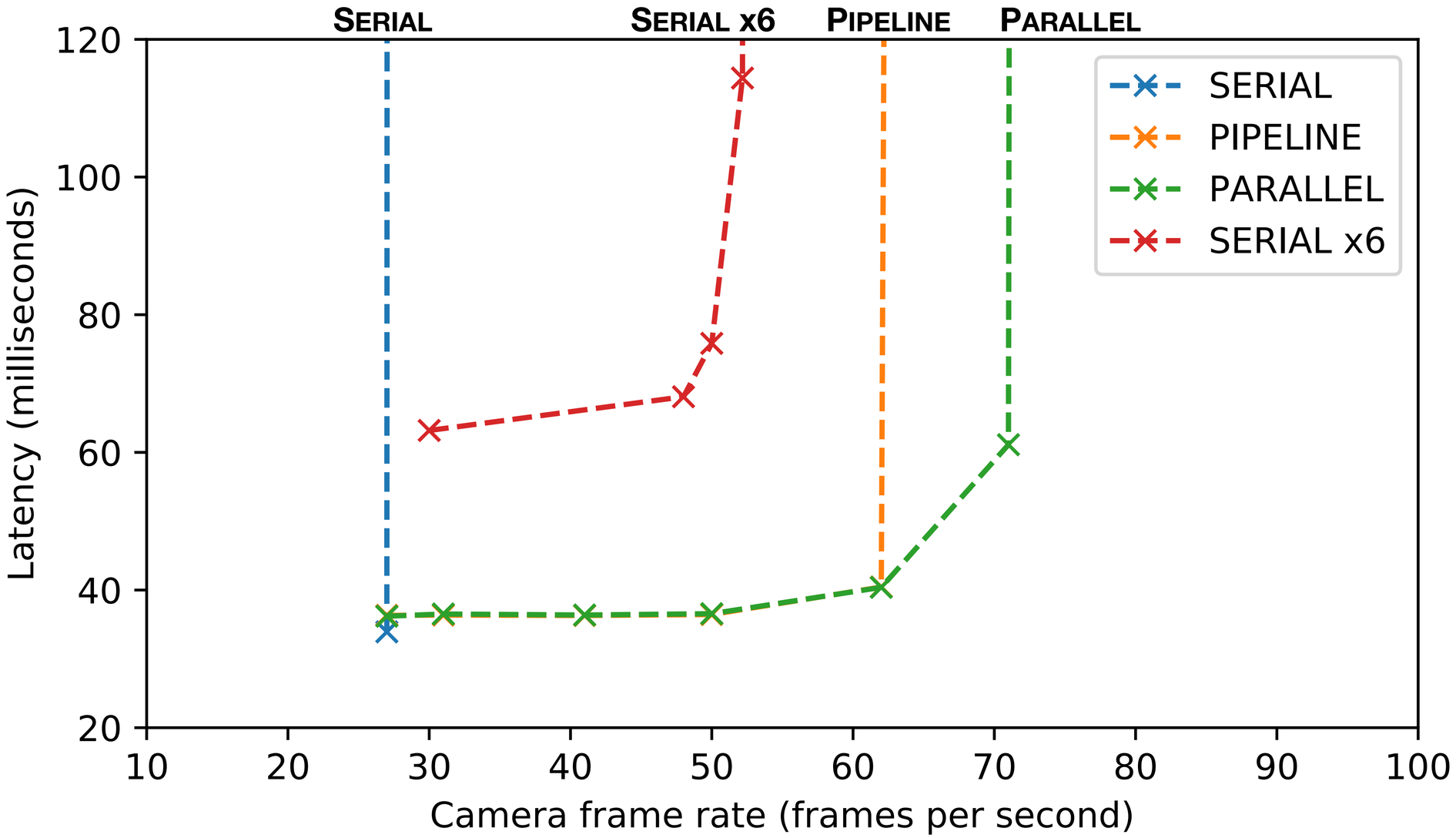
Part I:

Parallel Execution

Part II:



Multi-camera Composite Images

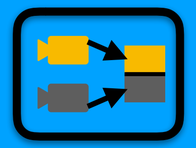


Evaluation Results

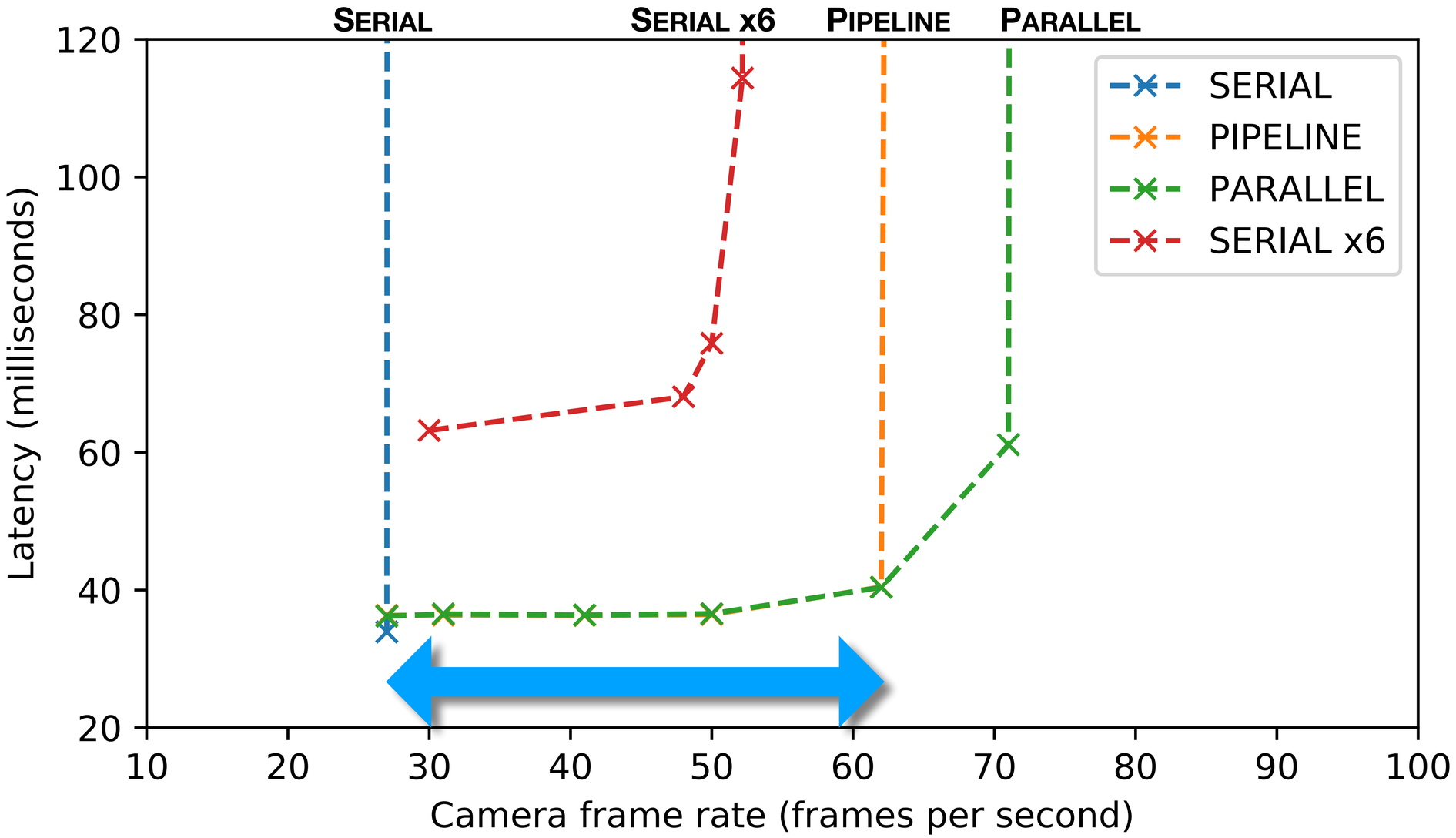
Part I:

Parallel Execution

Part II:



Multi-camera Composite Images

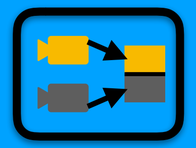


Evaluation Results

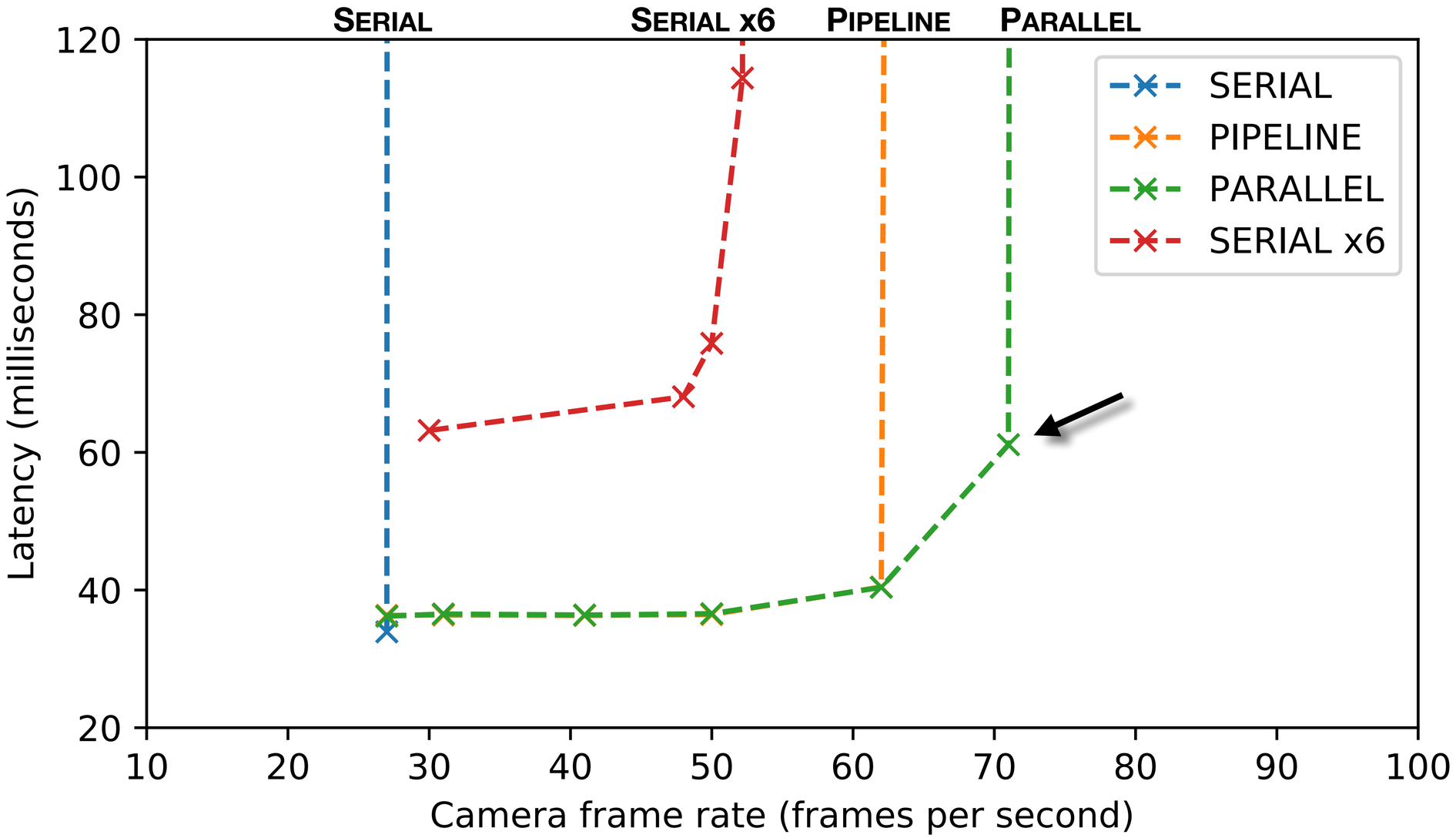
Part I:

Parallel Execution

Part II:



Multi-camera Composite Images

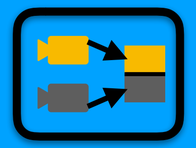


Evaluation Results

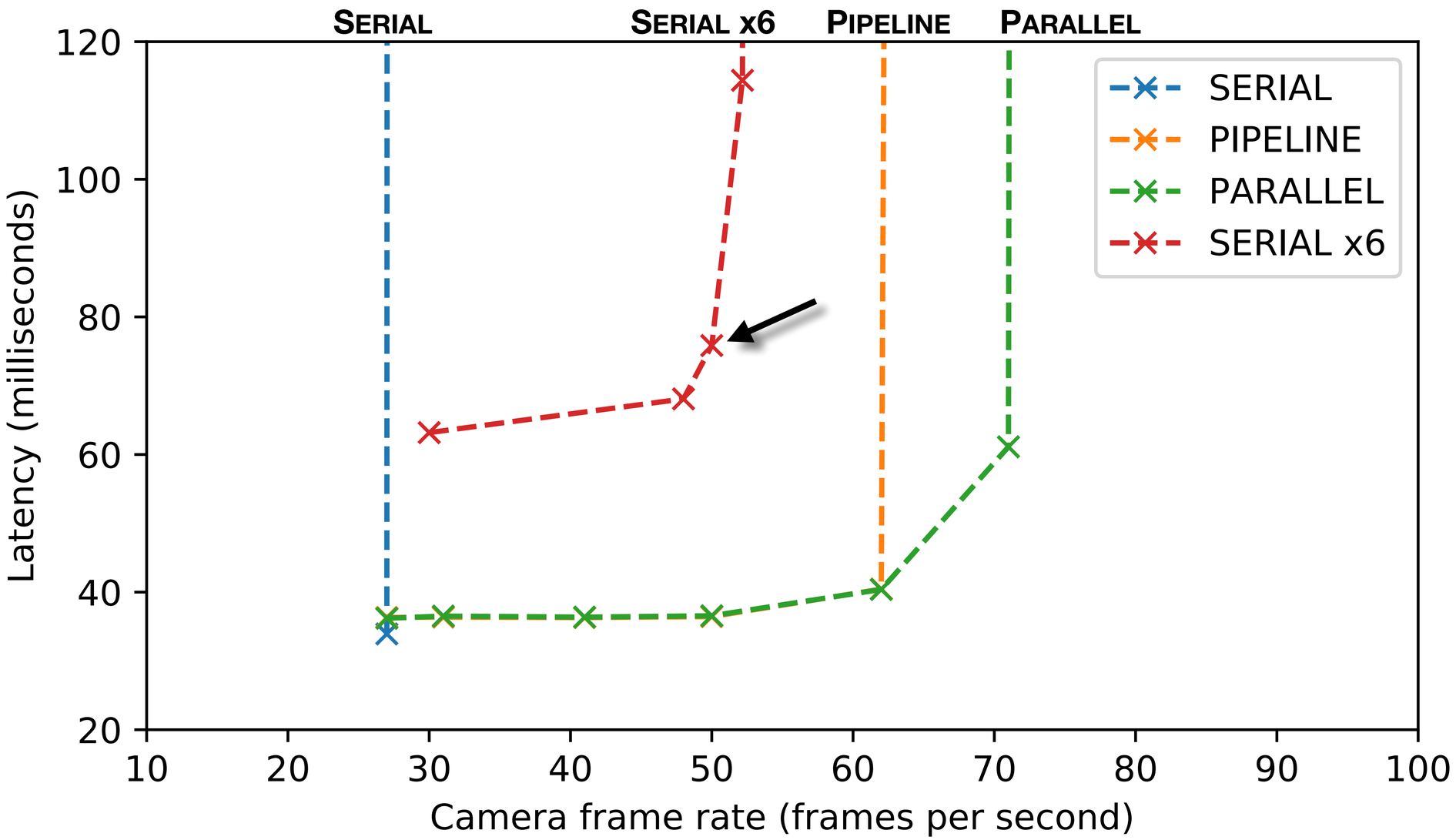
Part I:

Parallel Execution

Part II:



Multi-camera Composite Images

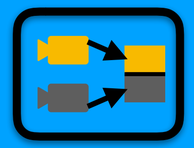


Evaluation Results (cont.)

Part I:

Parallel Execution

Part II:



Multi-camera Composite Images

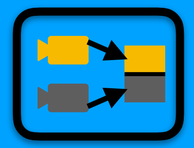
	CPUs (%)	Memory (MB)
SERIAL	92	774
SERIAL X6	536	4,644
PIPELINE (Single thread per stage)	219	1,132
PARALLEL (10 threads per stage)	239	1,136

Evaluation Results (cont.)

Part I:

Parallel Execution

Part II:



Multi-camera Composite Images

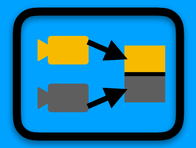
	CPUs (%)	Memory (MB)
SERIAL	92	774
SERIAL X6	**536**	**4,644**
PIPELINE (Single thread per stage)	219	1,132
PARALLEL (10 threads per stage)	239	1,136

Evaluation Results (cont.)

Part I:

Parallel Execution

Part II:



Multi-camera Composite Images

	CPUs (%)	Memory (MB)
SERIAL	**92**	**774**
PIPELINE (Single thread per stage)	**219**	**1,132**
PARALLEL (10 threads per stage)	239	1,136

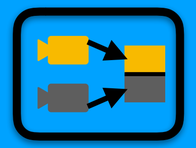
CPU and memory overheads are acceptable.

Evaluation Results (cont.)

Part I:

Parallel Execution

Part II:



Multi-camera Composite Images

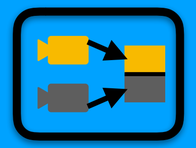
	CPUs (%)	Memory (MB)
PIPELINE (Single thread per stage)	219	1,132
PARALLEL (10 threads per stage)	239	1,136

Enabling intra-stage parallelism takes slight overheads.

Part I:

Parallel Execution

Part II:



Multi-camera Composite Images

Part I: Parallel Execution

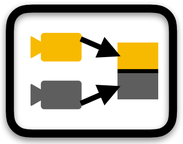
- Pipeline and Parallel improve throughput from **28 FPS to 71 FPS**
- With **acceptable overheads** and
- **No accuracy loss**

Multi-camera Composite Images

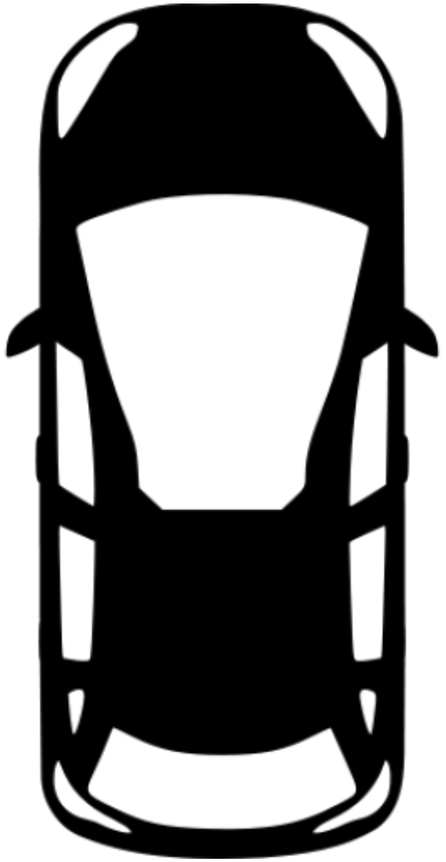
Part I:

Parallel Execution

Part II:



Multi-camera Composite Images

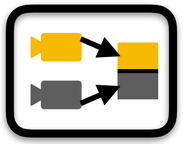


Multi-camera Composite Images

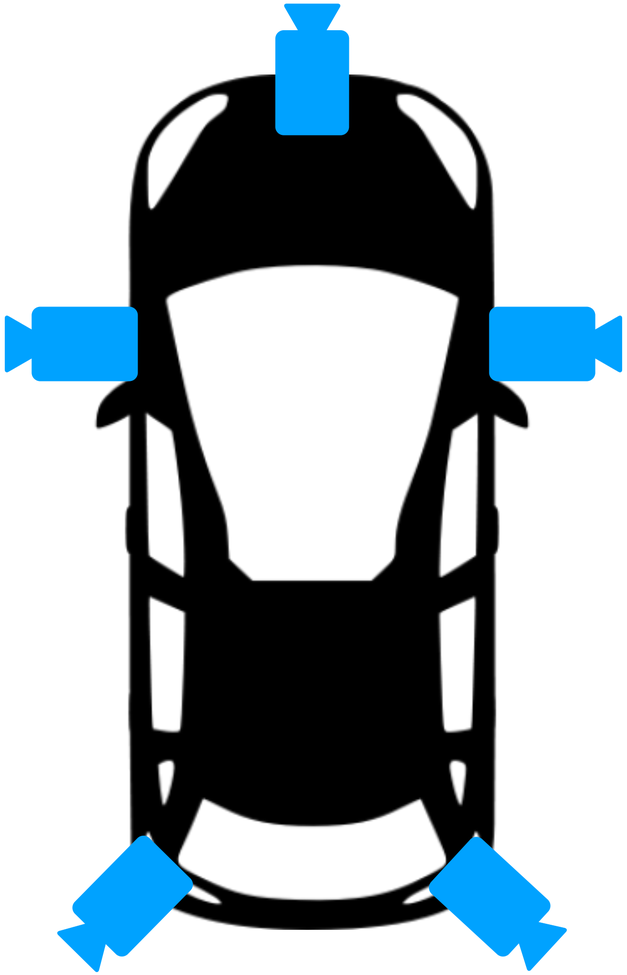
Part I:

Parallel Execution

Part II:



Multi-camera Composite Images

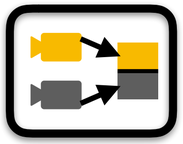


Multi-camera Composite Images

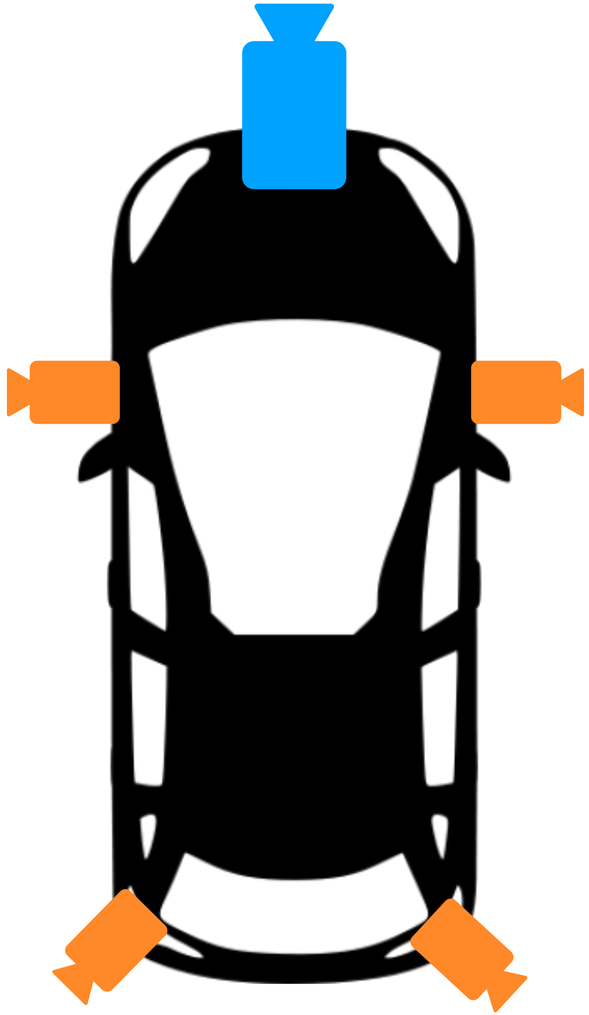
Part I:

Parallel Execution

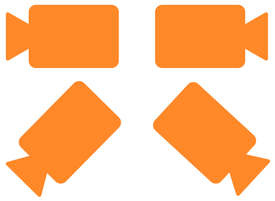
Part II:



Multi-camera Composite Images



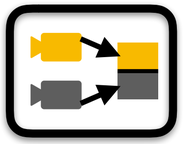
Multi-camera Composite Images



Part I:

Parallel Execution

Part II:



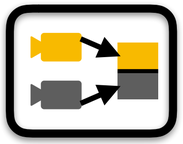
Multi-camera Composite Images

Multi-camera Composite Images

Part I:

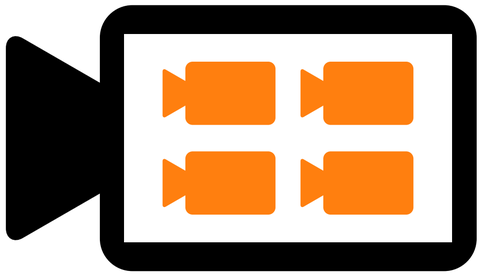
Parallel Execution

Part II:



Multi-camera Composite Images

Virtual Camera

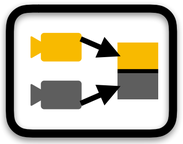


Multi-camera Composite Images

Part I:

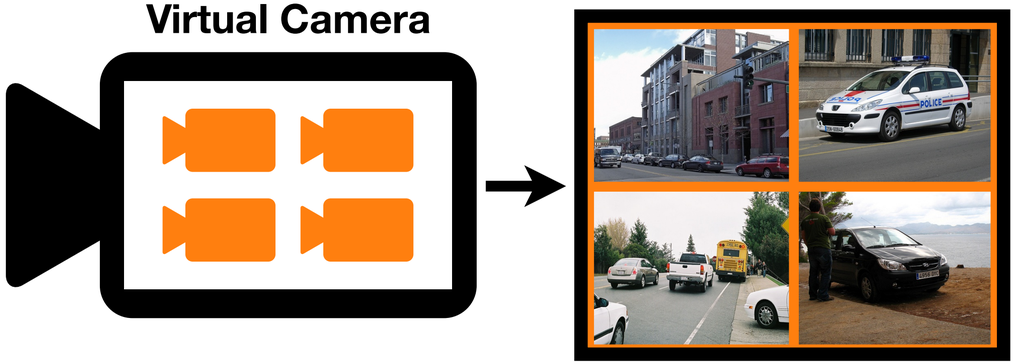
Parallel Execution

Part II:



Multi-camera Composite Images

Virtual Camera



Images are from PASCAL dataset

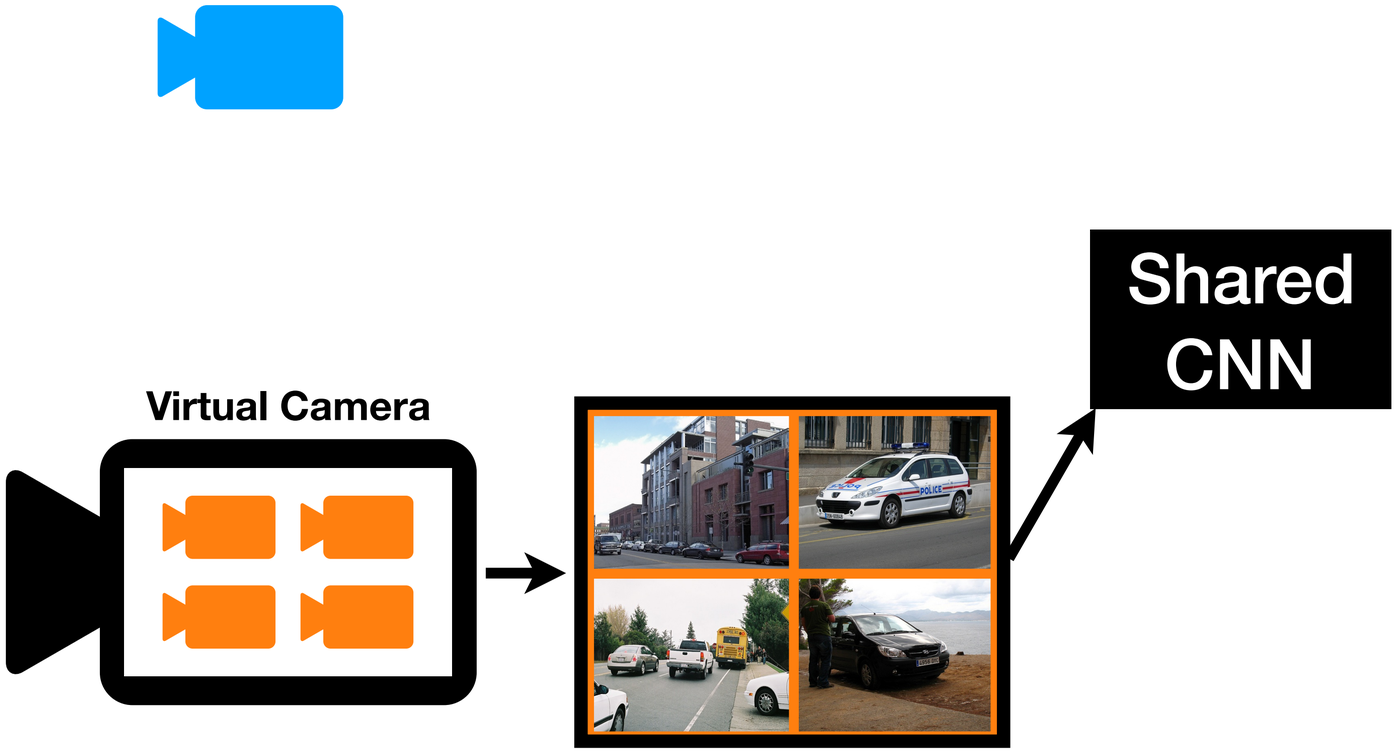
Multi-camera Composite Images

Part I:

Parallel Execution

Part II:

Multi-camera Composite Images



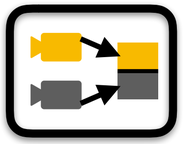
Images are from PASCAL dataset

Multi-camera Composite Images

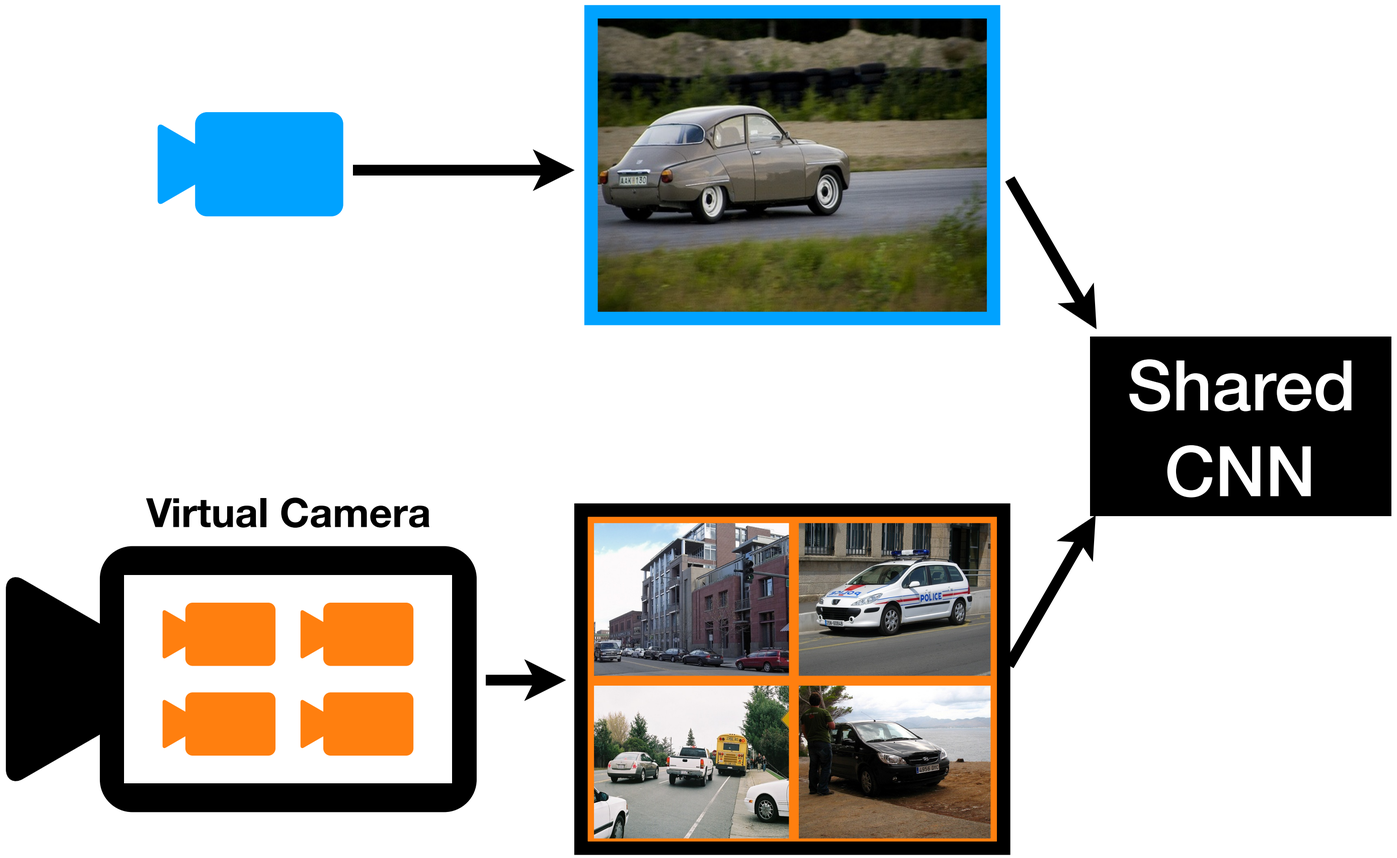
Part I:

Parallel Execution

Part II:



Multi-camera Composite Images



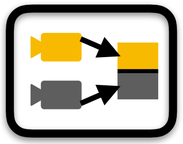
Images are from PASCAL dataset

Multi-camera Composite Images

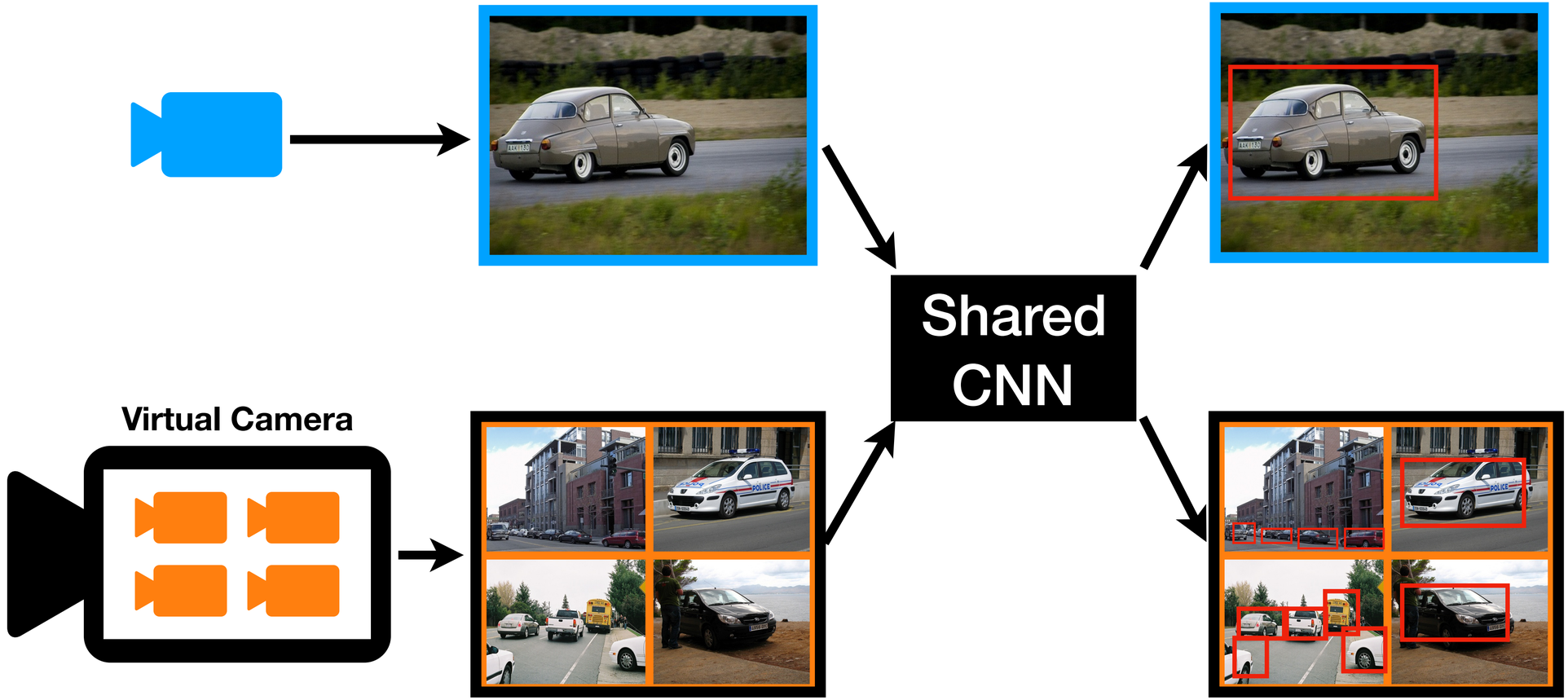
Part I:

Parallel Execution

Part II:



Multi-camera Composite Images



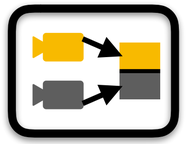
Images are from PASCAL dataset

Evaluation

Part I:

Parallel Execution

Part II:



Multi-camera Composite Images

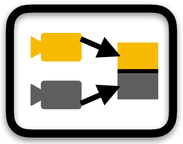
- We compared **latency**, **throughput** and **accuracy** between
  - **Full-size** images
  - **Four-camera composite** images

Evaluation Results

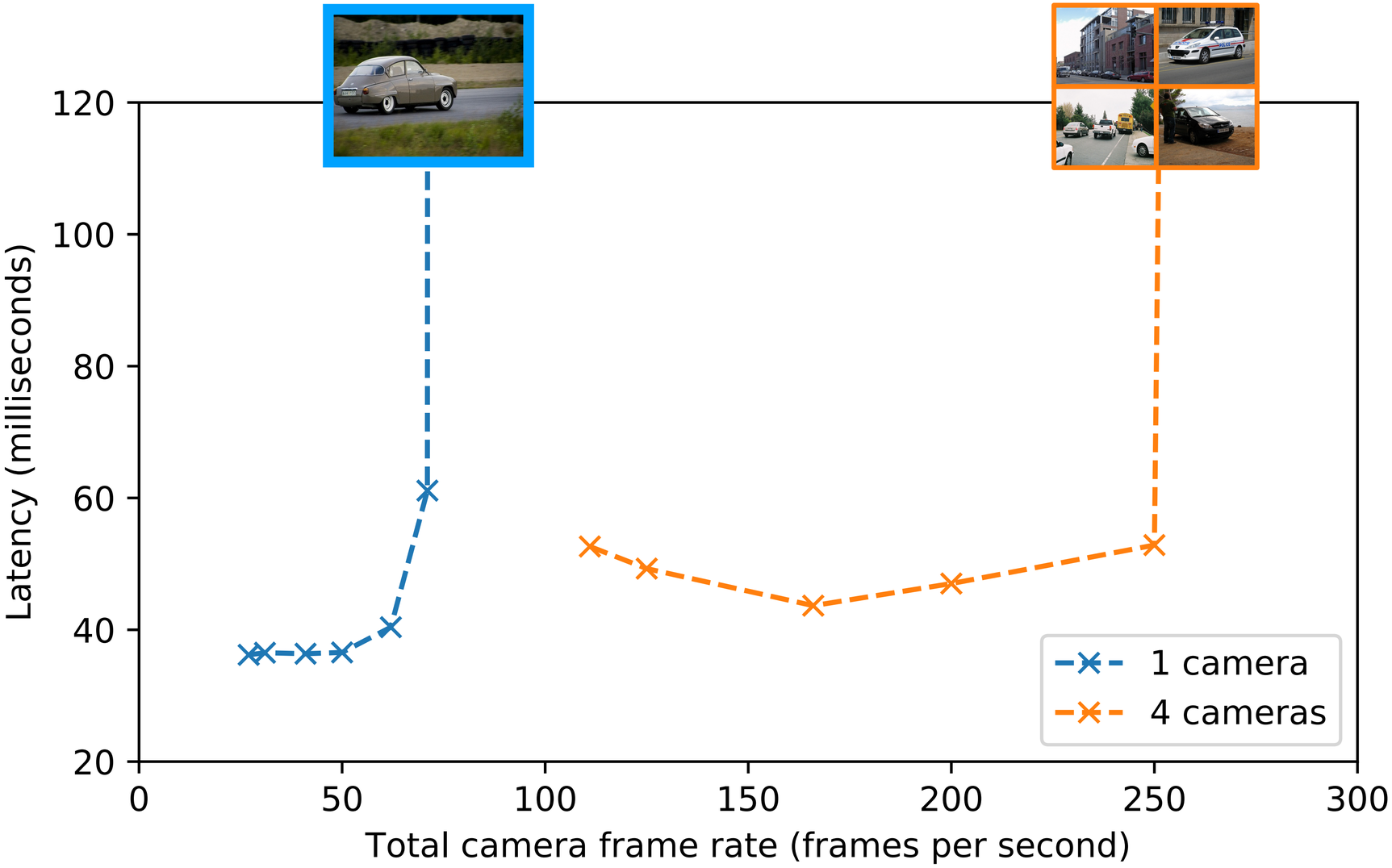
Part I:

Parallel Execution

Part II:



Multi-camera Composite Images



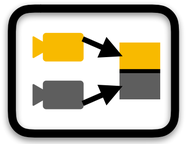
Images are from PASCAL dataset

Evaluation Results (cont.)

Part I:

Parallel Execution

Part II:



Multi-camera Composite Images

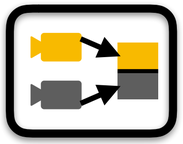
Images are from PASCAL dataset

Evaluation Results (cont.)

Part I:

Parallel Execution

Part II:



Multi-camera Composite Images

Table 1: accuracy (mAPs) of object classes relevant to autonomous driving

Full-size Test

Composite Test

Classes: bicycle, bus, car, motorbike, train, bird, person, cat, cow, dog, horse, sheep

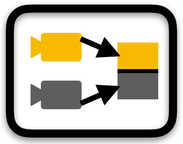
Images are from PASCAL dataset

Evaluation Results (cont.)

Part I:

Parallel Execution

Part II:



Multi-camera Composite Images

Table 1: accuracy (mAPs) of object classes relevant to autonomous driving

	 Full-size Test	 Composite Test
Original YOLO	63.66	44.91

Classes: bicycle, bus, car, motorbike, train, bird, person, cat, cow, dog, horse, sheep

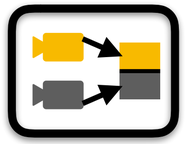
Images are from PASCAL dataset

Evaluation Results (cont.)

Part I:

Parallel Execution

Part II:



Multi-camera Composite Images

Table 1: accuracy (mAPs) of object classes relevant to autonomous driving

	 Full-size Test	 Composite Test
Original YOLO	63.66	44.91
Retrained YOLO	66.20	56.21

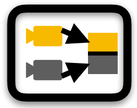
Classes: bicycle, bus, car, motorbike, train, bird, person, cat, cow, dog, horse, sheep

Images are from PASCAL dataset

Conclusions

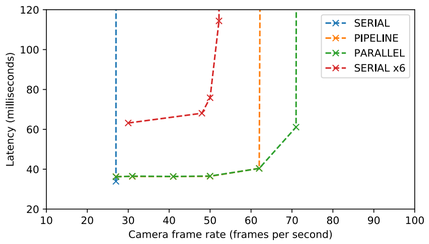
- We presented an industrial study that addresses the challenge of supporting multiple cameras.

Parallel execution



Multi-camera composite image

- **Evaluation results** showed
  - Significant throughput improvements
  - No accuracy loss with parallel execution
  - ~7.4% accuracy drop with multi-camera composite image (but 4-fold throughput improvement!)



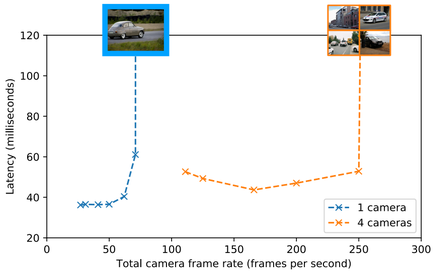


Table 1: mAPs of object classes relevant to autonomous driving

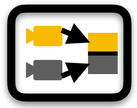
	Full-size Test	Composite Test
Original YOLO	**63.66**	44.91
Retrained YOLO	66.20	**56.21**

Images are from PASCAL dataset

Conclusions

- We presented an industrial study that addresses the challenge of supporting multiple cameras.

Parallel execution

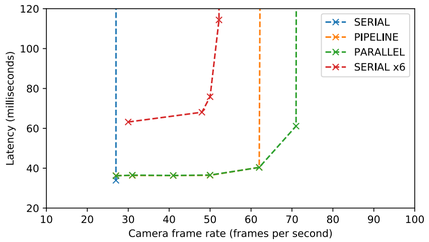


Multi-camera composite image

Other considerations in the paper:

- Configurable stages
- Multi-GPU execution

- **Evaluation results** showed
  - Significant throughput improvements
  - No accuracy loss with parallel execution
  - ~7.4% accuracy drop with multi-camera composite image (but 4-fold throughput improvement!)



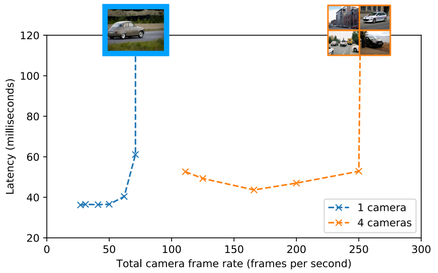


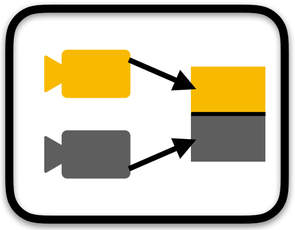
Table 1: mAPs of object classes relevant to autonomous driving

	Full-size Test	Composite Test
Original YOLO	**63.66**	44.91
Retrained YOLO	66.20	**56.21**

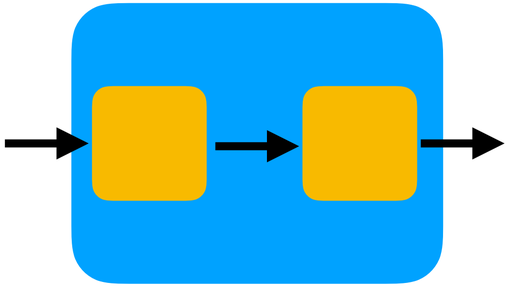
Images are from PASCAL dataset

Future Work

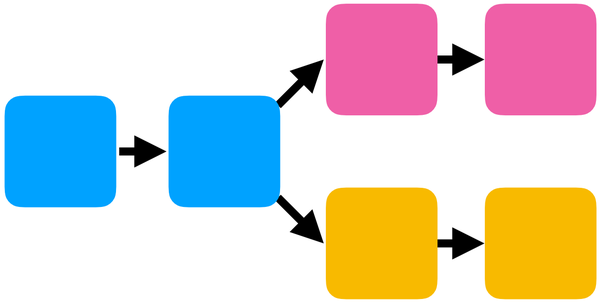
- Dynamically apply composite-image technique with criticality change



- Finer granularity of stages



- Dynamically share CNN among multiple models



Thank you!

Re-thinking CNN Frameworks for Time-Sensitive Autonomous-Driving Applications: Addressing an Industrial Challenge

Ming Yang[1], Shige Wang[2], Joshua Bakita[1], Thanh Vu[1], F. Donelson Smith[1], James H. Anderson[1], and Jan-Michael Frahm[1]

[1]The University of North Carolina at Chapel Hill
[2]General Motors Research

THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL