# Chytrá kamera pro sledování dopravy

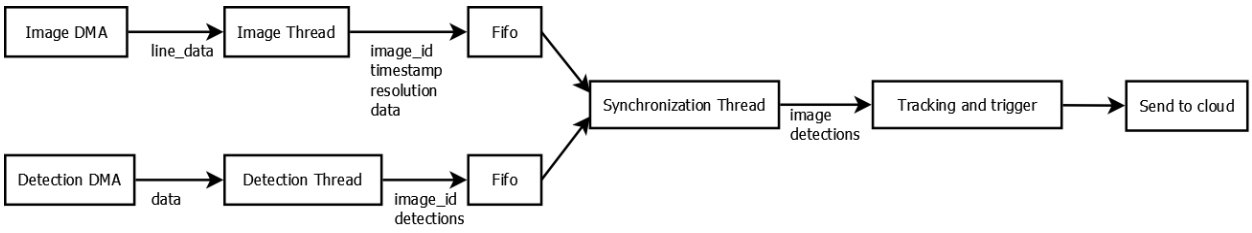
Funkční vzorek TH04010144-V1

Roman Juránek, Petr Musil, Pavel Zemčík
*Fakulta informačních technologií VUT v Brně*
*Božetěchova 2, 61200 Brno*
Petr Honec
*Camea, spol. s.r.o*
*Karásek 2290/1m, 621 00 Brno*

Tento funkční vzorek byl vytvořen s finanční podporou TA ČR

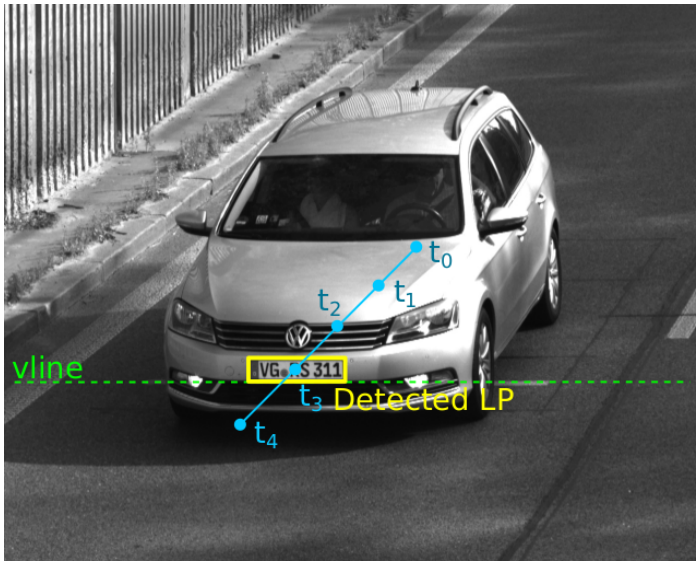
## Popis funkčního vzorku

Funkční vzorek integruje do jednoho celku - chytré kamery - technologie vyvinuté na FIT VUT. Jedná se zejména o FPGA komponenty - HDR snímání obrazu a detektor objektů. Dále pak obsahuje softwarovou část - zpracování dat ze snímače obrazu a zpracování detekcí objektů. Poslední částí jsou modely strojového učení pro detekci RZ vozidel, použité v FPGA. Vše je integrováno do kamery vyrobené v CAMEA spol. s r.o.




Obrázek 1: Blokový diagram toků dat v kameře. Obrazový tok z FPGA (v bloku *Image thread*) je synchronizován s detekcemi objektů provedenými rovněž na FPGA (čtenými blokem *Detection thread*). Data jsou zpracována v bloku "*tracking and trigger*" a při detekci události jsou vybraná data poslána na server.

Experimentální kamera pořizuje multiexpoziční snímky (sekvence 3 snímků s různými expozičními časy) a detekuje v nich RZ. Detekované značky jsou sledované, a pokud trajektorie značky překročí virtuální čáru (linie definovaná uživatelem, která je specifická pro sledovanou scénu), je vytvořena událost, která je poslána na datový server. Událost obsahuje zejména fotku vozidla a čas překročení čáry, ale i další metadata. Blokový diagram toku dat v kameře je naznačen na Obrázku 1, ilustrace dat z kamery je pak na Obrázku 2.




Obrázek 2: Ilustrace obrazu z kamery s naznačenými metadaty - detekcí značky, její trajektorií a virtuální čárou, jejíž překročení vyvolalo událost.

Činnost kamery je základem aplikací jako počítání a kategorizace vozidel nebo měření úsekové rychlosti. Obvykle jsou tyto úlohy řešené jednoduššími algoritmy a v kooperaci kamery s průmyslovým počítačem (v blízkosti kamery aby se zabránilo velkým datovým tokům přes síť). Experimentální kamera ukazuje, že je možné, s komponentami vyvinutými na FIT VUT, tyto úlohy řešit přímo v kameře a tak 1/ zjednodušit řešení, 2/ snížit jeho cenu a 3/ snížit spotřebu energie. Navíc, v případné kombinaci s dalšími senzory (jako například radar), otevírá možnosti k využití v dalších úlohách jako například měření okamžité rychlosti vozidel.

## Hardware kamery

Kamera je postavena na SoC Xilinx Zynq, který obsahuje FPGA a ARM procesor s obrazovým senzorem OnSemi Python 5000 s maximálním rozlišením 5 Mpx. FPGA zajišťuje komunikaci SoC se senzorem, snímání obrazu a detekci objektů. Sensor je připojen přes rychlé LVDS rozhraní, které umožňuje přenést až 120 FullHD snímků za sekundu (FPS). Obrazový senzor umožňuje přesné řízení expozice snímků pomocí externích signálů. Toho je využito při multiexpozičním snímání, kdy jsou snímky v sekvenci pořízeny s minimálními rozestupy což napomáhá k redukci artefaktů při skládání HDR obrazu.

**Sensor** OnSemi Python 5000.
**Platforma** Xilinx Zynq Z7020 - dvoujádrový ARM v7 s 1GB RAM

**Komponenty** využité v kameře jsou ***ACF Core*** [4] a ***HDR Core*** [5]. ACF Core zajišťuje detekci objektů pomocí klasifikátoru. Detektor je implementován v FPGA pouze s využitím vnitřní paměti čipu a zpracovává obraz již při vyčítání ze senzoru. Jeho výkon je dostatečný pro vyhodnocení několika modelů současně. HDR Core zajišťuje řízení expozice, časování snímače, skládání a tone mapping HDR obrazu. Obě komponenty byly vyvinuty na FIT VUT v rámci výzkumných projektů.

Obrázek 3: Experimentální kamera při pořizování dat.

## Software kamery

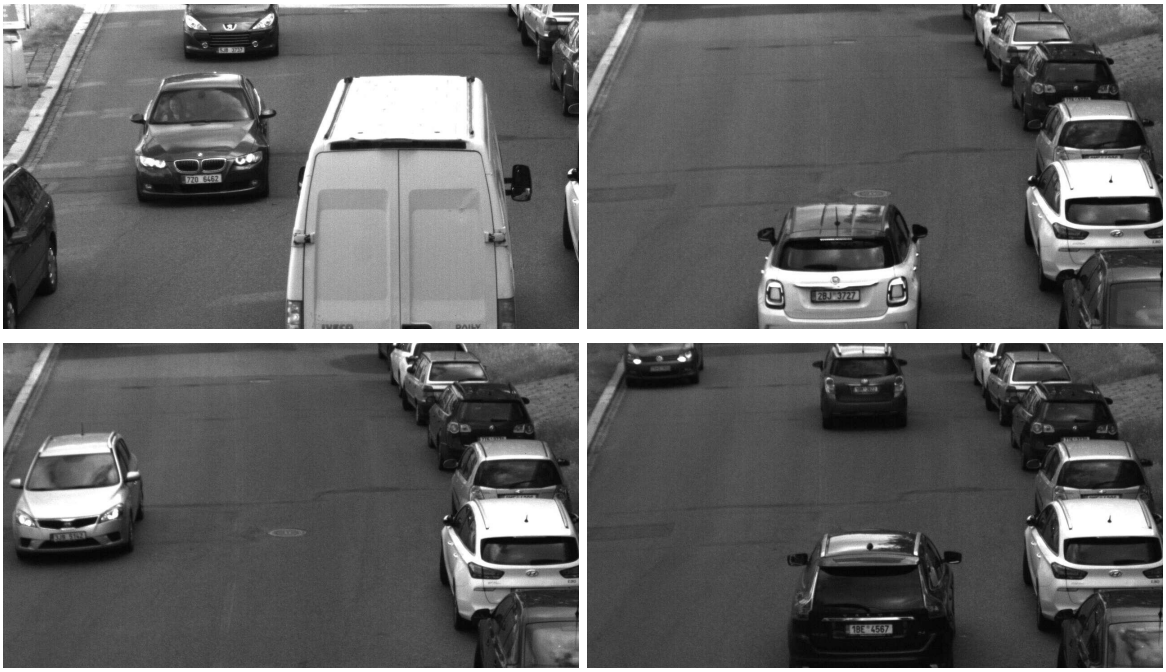
**Operační systém** - Linux openSUSE Tumbleweed

**Ovladače -** DMA driver pro rychlý transfer dat z FPGA do paměti ARM procesoru. Využívá IP jádro AXI DMA, které řeší kopírování dat z FPGA bez zatížení procesoru. Jádro má na vstupu Axi Stream rozhraní. DMA driver nastavuje konfiguraci tohoto jádra, připravuje adresový prostor pro kopírování dat s scatter-gather listy a řeší ztráty dat, přetečení a podobně. V operačním systému se tváří jako soubor za kterého je možno číst data standardním blokovým způsobem. Slouží pro transfer obrazových dat a výsledků detekce do paměti procesoru pro další zpracování.

**Obslužný software** zajišťuje konfiguraci zařízení, zpracování obrazových dat a výsledku detektoru z FPGA, výběr vhodných snímků pro další zpracování a jejich odesílání na server. Pro zajištění maximální výkonnosti je přijímání obrazových dat, výsledků detektoru a jejich vzájemná synchronizace implementován v separátních vláknech (*Image thread, Detection thread, Synchronization thread)* jak je znázorněno na obrázku 1. Image Thread čte obrazová data po řádcích z dma a rekonstruuje z nich obrazové pole. Detekuje ztracené řádky a vyplní obrázek do správné velikosti. Počet ztracených řádků je logován, ale je to obecně velmi vzácné. Detection Thread čte data z detekčního IP a transformuje je do správného formátu. Detekční IP vrací v případě detekce pozici, identifikaci měřítka, skóre a identifikaci klasifikátoru. Výsledky detekce jsou promítnuty do obrazu jako ohraničující obdélník. Synchronizační vlákno se

používá k párování správných detekcí a obrázků. Jakékoli chybějící detekce nebo obrázky jsou logovány.
Tato část aplikační pipeline je pevná, misto pro uživatelskou aplikaci je na výstupu synchronizačního vlákna. Podle měření zatěžuje tato pevná část pipeline procesor ARM jedním procentem jednoho jádra pro jeden snímek FullHd za sekundu (60 FPS = 60 % zatížení jednoho jádra procesoru).
V případě tohoto funkčního vzorku je uživatelská aplikace detekce automobilů podle RZ pro účely úsekového měření rychlosti. Detekované RZ jsou sledovány (tracking) přes více snímků s využitím Kalmanova filtru [1] pro odhad následující pozice objektu a algoritmu pro asociaci detekcí stejného objektu [2]. Jsou seskupeny detekce pro multiexpoziční snímky. Pro další zpracování do cloudu je vybrána fotografie objektu, který překročí virtuální čáru v obraze definovanou v konfiguraci. Pozice této virtuální čáry obvykle kopíruje reálnou čáru nakreslenou na silnici a slouží k přesnému měření času překročení pro úsekovou rychlost. Ve standardním provozu je takto vybráno pro odeslání méně než jeden snímek za sekundu pro jednoproudovou silnici.
Pro komunikaci s cloudem je využita knihovna ZeroMQ [3]. Jsou odesílány snímky ve formátu JPEG s připojenými údaji o snímku (id snímku, id kamery, timestamp, detekce) ve textovém formátu JSON. V případě multiexpozičního snímání mohou být (podle nastavení) odeslány všechny snímky v sekvenci. Komprimace jednoho FullHD snímku za sekundu zvýší zatížení jednoho jádra procesoru přibližně o 6%. Komprimované snímky FullHD obrazu mají přibližně 100 KB. Zpracování 20 tříexpozičních snímku (celkem 60 snímků) za sekundu s odesíláním průměrně 0.5 snímku za sekundu na cloud zatěžuje jedno jádro procesoru průměrně na 70%.
**Data modelů** pro detekci RZ jsou konfigurace detektoru v serializované v univerzálním formátu Protocol Buffers. Modely je možné měnit za běhu aplikace, to usnadňuje aktualizaci zařízení a dovoluje například použít jiný klasifikátor na den a noc.

## Ukázky z provozu



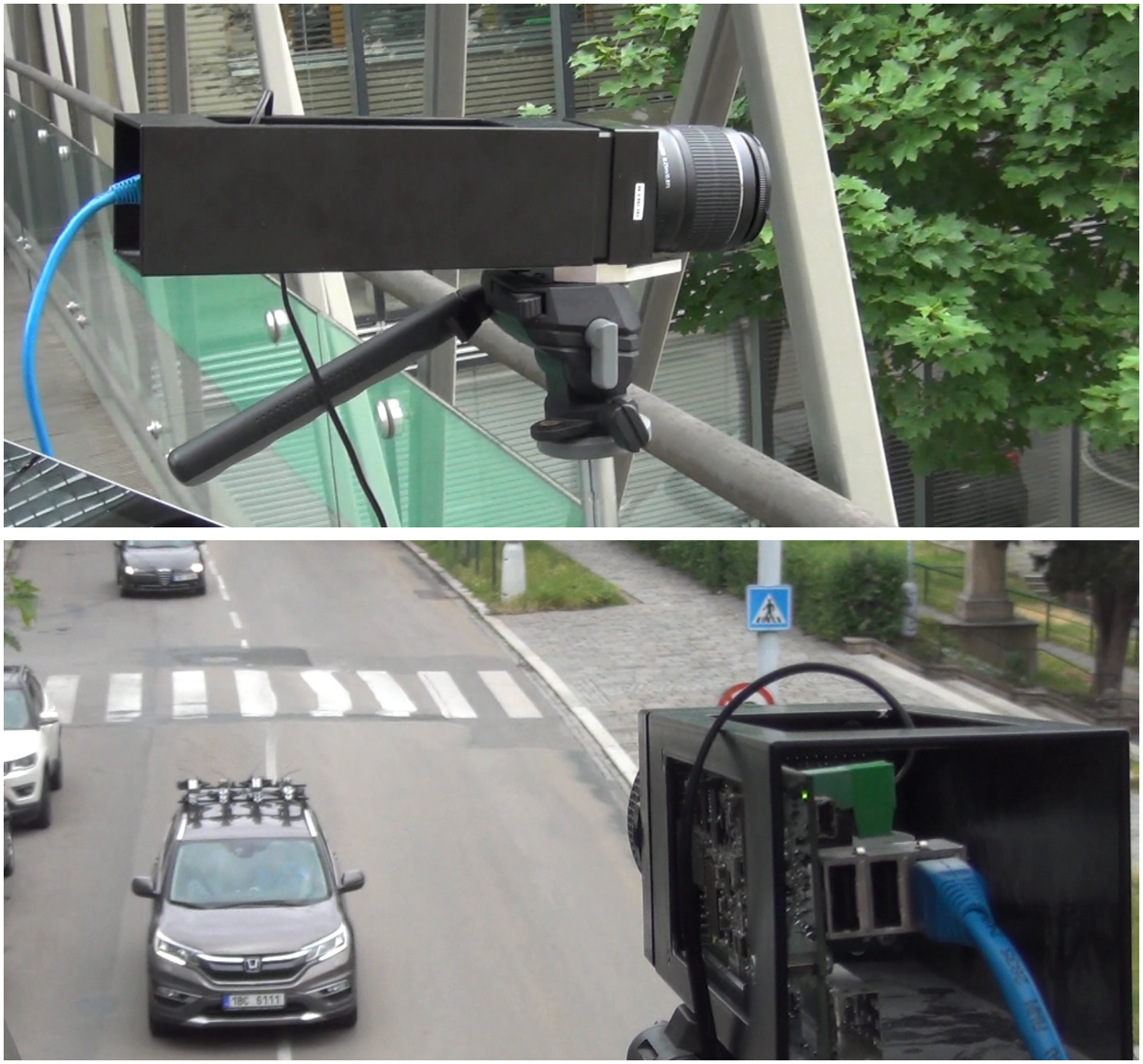
Obrázek 4: Ukázka dat pořízených funkčním vzorkem - fotografie vozidel projíždějících definovanou oblastí. Ka každé fotce jsou navíc uložená metadata - trajektorie vozidla, časová značka atd.

## Použitá data a softwarové nástroje

Data pro trénování modelů strojového učení dodala CAMEA spol. s r.o. Jednalo se řádově o desítky tisíc fotografií s automatickými anotacemi rohů RZ. Tato data byla použita pro vytvoření detektorů pomocí *WaldBoost package pro Python*[1], které jsou kompatibilní s architekturou ACF Core použitou v FPGA kamery.

[1] https://github.com/RomanJuranek/waldboost

# Fotodokumentace



Obrázek 5: Funkční vzorek při pořizování dat.

# Reference


1. Kalman filter, https://en.wikipedia.org/wiki/Kalman_filter
2. Hungarian algorithm, https://en.wikipedia.org/wiki/Hungarian_algorithm
3. ZeroMQ, https://zeromq.org
4. Musil et al: Cascaded stripe memory engines for multi-scale object detection in FPGA, IEEE TCSVT, 2020
5. Musil, M., Nosko, S., Zemcik, P.: De-ghosted HDR video acquisition for embedded systems. JRTIP, 2020