

Goedkope embedded hardware voor computervisie

07 april 2021, 02:00

Dankzij de enorme vooruitgang in computervisie en de steeds grotere beschikbaarheid van goedkope computercapaciteit de voorbije jaren, kon makkelijk worden voldaan aan de vraag naar slimme oplossingen op kleine singleboardcomputers. Steeds vaker worden computervisiealgoritmen ingezet voor de visie op een verscheidenheid van perifere gebruikstoepassingen: drones, beveiligingscamera's, mobiele applicaties, retailanalyses enz.

We ontwikkelden onlangs een proof of concept voor een bedrijf, met de bedoeling bepaalde objecten real time te detecteren.

Proof of concept

In de opstelling van het proof of concept legt een camera continu beelden vast en transfereert deze naar een minicomputer, hier een Raspberry Pi 4, die ze dan verwerkt met behulp van de bibliotheek Python OpenCV. We begonnen met het detecteren van legoblokken van een bepaalde kleur, bijvoorbeeld rood.

Als hoofdverwerkingseenheid gebruikten we de Raspberry Pi 4, een goedkoop, lichtgewicht opensource hardwareplatform in creditcardformaat, met een krachtige CPU en tal van andere nuttige functionaliteiten om als een volwaardige computer te werken. Perfect dus voor mobiele en andere applicaties waarbij de grootte, het gewicht en de kosten van kapitaal belang zijn. Voor de

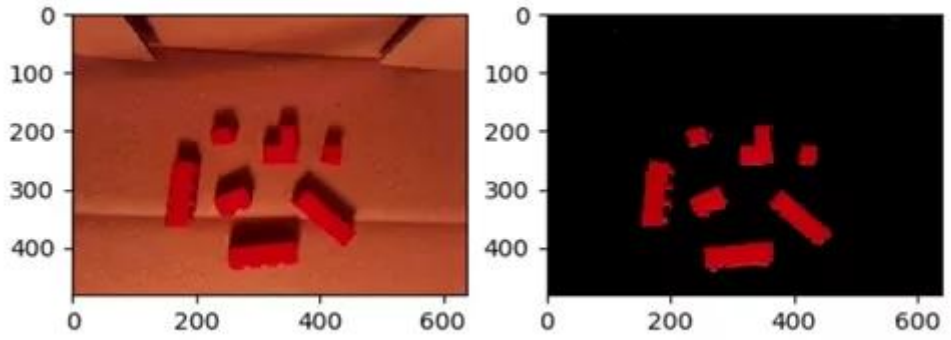
beeldopname in deze opstelling werd de Raspberry Pi Camera v2 gebruikt, een goedkope en hoogkwalitatieve 8-megapixel Sony beeldsensor die statische beelden van 3280x2464 pixels en ook 1080p videobeelden kan vastleggen.

Deze opstelling voldeed aan de vereiste om continu beelden op te nemen en ze te verwerken aan 100 fps (beelden per seconde). Met de Pi camera konden we zeer eenvoudig de videobeelden aan 100 fps opnemen. En ook de beeldverwerking kon makkelijk aan de correcte snelheid worden uitgevoerd. Evalueert men de snelheid van beide processen afzonderlijk, dan zou men kunnen concluderen dat de hardware in staat moet zijn om beide processen samen aan de vereiste snelheid uit te voeren. Maar toen we ook de legoblokken in elk opgenomen beeld moesten uitsegmenteren, slaagde onze opstelling er aanvankelijk niet in de verwerkingssnelheid van 100 fps te halen; in de plaats daarvan zakte de snelheid naar 30 fps.

Profilering en multithreading

In de meeste realtimesystemen zijn prestaties essentieel. Programmeurs weten hoe moeilijk en tijdrovend het is de oorzaken van slechte prestaties te achterhalen. Als de hoofdoorzaak eenmaal is gevonden, ligt de oplossing vaak voor de hand. **Profilering** is het monitoren van een toepassing op verschillende uitvoeringsniveaus om te achterhalen waar de resources worden gebruikt. Hiertoe stelt men onder meer vast wanneer een methode wordt uitgevoerd en hoe lang de methode erover doet om een gegeven taak te volbrengen. Na de profilering van onze toepassing om uit te maken hoe lang elke functie duurt, voerden we een **multithreading** uit. Deze methode verbetert de uitvoeringsprestaties van een proces aan de hand van 'concurrency' of het tegelijkertijd uitvoeren van meerdere taken (threads), los van elkaar binnen dat programma. De hoofdtaak was het opnemen van beelden aan 100 fps, terwijl een tweede taak erin bestond de opgenomen beelden te verwerken aan 180 fps. Na het synchroniseren van beide taken, slaagde onze opstelling er uiteindelijk in zowel op te nemen als te verwerken aan 100 fps.

[illegible] Raspberry Pi:



Afrondend, geven we graag de volgende twee stellingen mee:

1. Onderschat nooit de mogelijkheden van dergelijke goedkope systemen voor slimme realtimetoepassingen.
2. Men kan een hoge efficiëntie bereiken aan de hand van klassieke programmeertechnieken, zonder blind te investeren in dure hardware.

Kampt u met gelijkaardige problemen? Neem dan contact met ons op!

(Bron beeld bovenaan: https://www.dreamstime.com)

Auteurs