**Detecteren van de dispenser:**

Voor het detecteren van de dispenser hebben we even moeten nadenken. Je kan dit op verschillende manieren proberen. Je kan een systeem training om een specifieke dispenser te herkennen. Alleen is dit niet zo efficiënt voor onze opdrachtgever. Telkens als de UZ Leuven een nieuwe dispenser zou gebruiken moet het systeem opnieuw getraind worden. Van dit idee stapte we dus al snel af.

Ons volgende idee was om met een QR-code de dispenser te detecteren. In het begin leek dit een goede oplossing. We hadden een python programma geschreven voor de Google Coral om, met behulp van Pyzbar (een extensie van python), QR-codes te kunnen detecteren. Vanaf hier begon het testen. We zorgden er eerst voor dat het systeem alleen de QR-code van de dispenser detecteerde en dit verliep ook vlot om te maken. Nu zaten we alleen nog met het probleem van afstand. We hadden dit op voorhand getest met een onze gsm’s. Hiermee konden we toch van vrij ver een kleine QR-code scannen. Alleen had de Google Coral het hier toch iets moeilijker mee. Aangezien de camera in de UZ Leuven op toch een vrij hoge en verre afstand van de dispenser zou hangen, besloten we deze na te bootsen. Bij het testen op deze situatie, konden we al snel concluderen dat we een zeer grote QR-code nodig hadden. Ook zou de detectie van de QR-code zeer afhankelijk zijn van in welke hoek deze naar de camera staat. We hebben ook de beelden van de camera omgezet met grayscalen. Dit zou het detecteren voor de camera makkelijker moeten maken. Te vergeefs heeft dit ook niet geholpen. Het idee van de QR-code detectie is dus moeilijker dan gedacht.

Het derde idee was om met een uniek logo te werken. Iets wat opvalt in het beeld van de camera. Opencv bied methodes tot het detecteren van specifieke images in een template, dus ons logo in het camera beeld. Wij hebben volgende methodes getest: TM\_CCOEFF, TM\_CCOEFF\_NORMED, TM\_CCORR, TM\_CCORR\_NORMED, TM\_SQDIFF, TM\_SQDIFF\_NORMED. Deze methodes bleken uiteindelijk niet performant genoeg te zijn. TM\_CCOEFF en TM\_CCOEFF\_NORMED vonden het logo bijna altijd. Alleen trokken deze de detectie kader rond het logo veel te groot en waren dus niet nauwkeurig genoeg. De andere methodes vonden het logo bijna nooit. Dit idee had ook weer problemen met in welke hoek het logo naar de camera staat.

Het vierde idee was om met behulp van tekstrecognition de dispenser te detecteren. Aangezien de vorige zo moeilijk verliepen en na een korte meeting met de klant hadden we toch besloten om met een statische detectie zone te werken. De bedoeling is dat bij de installatie de gebruiker één keer een cirkel trekt op het camerabeeld. Als een gedetecteerd persoon in deze cirkel staat voor een voldoende aantal tijd, dan zal deze geteld worden als iemand die zijn handen heeft ontsmet. Deze laatste aanpak werkt zeer goed op ons systeem en we hebben vandaaruit het systeem verder opgebouwd.

**Detecteren van mensen:**

Na wat zoeken hebben we uiteindelijk een opensource broncode (https://www.pyimagesearch.com/2018/08/13/opencv-people-counter/) gevonden voor het detecteren en tracken van mensen. Deze broncode gaf ons een goede start om er onze eigen versie van te maken, zo kon het al mensen detecteren, een id geven om te tracken doorheen de opeenvolgende frames (camerabeelden) en de gedetecteerde mensen tellen. In het begin telde deze code mensen die van boven naar onder en van onder naar boven wandelden in het beeld. Aangezien wij alleen de inkom moeten filmen hebben wij de code aangepast, zodat het alleen mensen die van onder naar boven wandelen telt. Ook hebben we een zone gemaakt waarin de gedetecteerde mensen worden geteld. Zo worden mensen die al geteld zijn geweest en van id zijn verandert niet opnieuw geteld, zolang ze uit deze zone staan.