VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

DIPLOMOVÁ PRÁCE

Brno, 2021 Bc. Přemysl Till

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY
DEPARTMENT OF CONTROL AND INSTRUMENTATION

NÁSTROJE PRO POČÍTÁNÍ A MONITOROVÁNÍ OSOB
PEOPLE COUNTING AND MONITOR TOOLS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Přemysl Till**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Soběslav Valach**
SUPERVISOR

BRNO 2021

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní program **Kybernetika, automatizace a měření**

Ústav automatizace a měřicí techniky

Student: Bc. Přemysl Till ***ID:*** 192828
Ročník: 2 ***Akademický rok:*** 2020/21

NÁZEV TÉMATU:

Nástroje pro počítání a monitorování osob

POKYNY PRO VYPRACOVÁNÍ:

1) Seznamte se s radary založenými na principu FMCW.
2) Prostudujte možnosti využití těchto radarů v oblasti zpracování a vyhodnocení pohybu osob.
3) Navrhněte vhodnou implementaci detekce osob pro moduly od společnosti Texas Instruments v řadách IWR a AWR.
4) Vytvořte aplikační softwarové vybavení pro obvody A/IWR a aplikaci pro vizualizaci a komunikaci s PC zaměřenou na různé techniky počítání osob. Předpokládá se nastavování zón pro počítání a osob a jejich toku zónou – vektor pohybu.

DOPORUČENÁ LITERATURA:

Rao S; Texas Instruments: Introduction to mmwave Sensing: FMCW Radars

Termín zadání: 8.2.2021 ***Termín odevzdání:*** 17.5.2021

Vedoucí práce: Ing. Soběslav Valach

doc. Ing. Petr Fiedler, Ph.D.
předseda rady studijního programu

ABSTRAKT

Práce se zabývá využitím mmWave radaru pro sledování osob a monitorování jejich pohybu mezi stanovenými zónami zájmu. Teoretická část práce shrnuje fyzikální podstatu použité technologie a zabývá se algoritmy, které umožňují využít ji ke sledování pohybu lidí. V části praktické je navržen konkrétní algoritmus pro využití radaru ke sledování fronty zákazníků a přítomnosti obsluhy v obchodech, což umožňuje automatizovat správu pokladen a sbírat data v souladu s direktivou GDPR. Následně byla vyvinuta vizualizace pro platformu Windows, která umožňuje komunikovat s radarem, nastavovat jeho parametry, zobrazit tok lidí v reálném čase a dále zpracovávat získaná data.

KLÍČOVÁ SLOVA

Radary s milimetrovou vlnou, mmWave, počítání osob, Group Tracker, monitorování prodejny

ABSTRACT

The paper details the usage of mmWave radars to track people and monitor their movement through predefined zones of interest. The theoretical part describes the physical nature of the technology and then describes algorithms which can be used to monitor using it to monitor the movement of people. In the practical part, I have developed a concrete algorithm which can be used to monitor customer queues and cash registers in shops and inform the cashiers when their presence is needed, as well as gather impersonal GDPR-compliant data about the customer's habits. Afterwards, I have developed a visualization for the Windows platform, which can be used to communicate with the radar, manage its configuration, visualize the events in real time and perform further analysis of the measured data.

KEYWORDS

Millimeter wave radars, mmWave, people counting, Group Tracker, shop monitoring

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Bc. Přemysl Till
VUT ID autora:	192828
Typ práce:	Diplomová práce
Akademický rok:	2020/21
Téma závěrečné práce:	Nástroje pro počítání a monitorování osob

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora*

*Autor podepisuje pouze v tištěné verzi.

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu Ing. Soběslavu Valachovi, za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci. Dále bych rád poděkoval Ing. Miroslavu Slováčkovi za neocenitelnou pomoc při ladění a testování aplikace v reálném provozu.

Obsah

Seznam obrázků

Úvod

Tato práce se zabývá využitím technologie *millimeter wave (mmWave)*, tj. radarového měření pomocí elektromagnetického vlnění s krátkou vlnovou délkou, pro monitorování pohybu osob. Cílem práce je vytvoření aplikace, která by umožňovala sledovat osoby na základě naměřených dat, zobrazovat je v reálném čase, ukládat a dále statisticky zpracovávat. Primárním využitím je monitorování prodejen, aplikace by měla umožnit automatizaci správy pokladen (automatické přivolání obsluhy, pokud je pokladna zavřená a přichází zákazník), sběr informací o pohybu zákazníků, jejich zobrazení a analýzu, na jejímž základě by bylo možné optimalizovat prodej. Za tímto účelem bývala běžně používána řešení založená na kamerovém sledování, nicméně ta jsou v současné době poměrně problematická. Data zaznamenaná kamerou umožňují identifikaci konkrétních zákazníků, pročež jsou ve sporu s evropskou směrnicí GDPR [1]. Oproti tomu data nasbíraná radarovým měřením jsou zcela anonymní a žádné osobní údaje z nich vyčíst nelze – jde tak o vhodného kandidáta pro nahrazení existujících systémů.

První dvě kapitoly práce tvoří teoretický úvod. Prvním krokem bylo seznámení se s fungováním samotných mmWave radarů, což nám umožňuje lépe pochopit možnosti i omezení zvolené technologie. Na začátku rozboru proto stručně popisuji použitou metodu, její fyzikální podstatu a kvalitativní parametry měření, kterých zvolený radar dosahuje. Dalším krokem pak byla analýza existujících algoritmů, které lze při vývoji aplikace využít. Za základ, na kterém je aplikace vybudována, jsem zvolil algoritmus *Group Tracker*, jehož implementace je součástí firmware radaru, zároveň ale krátce zmiňuji i jiné metody, které by bylo možné využít. Jde o algoritmus postavený na bázi Kalmanova filtru, který ale na rozdíl od většiny monitorovacích algoritmů nerozlišuje zvlášť funkce pro prostorové (*clustering*) a časové (*tracking*) sledování objektů, ale spojuje obě funkce dohromady, což mu umožňuje dosáhnout lepších výsledků, než jeho předchůdci [2]. Dále se věnuji také možnosti vylepšení výsledků tohoto algoritmu pomocí stavového automatu, který by bral v potaz zamýšlené využití a prostorové rozložení prodejny.

Praktická část práce pak sestává jednak z navržení, případně optimalizace použitých algoritmů, jednak z vytvoření aplikace pro PC, která by umožňovala konfigurovat a ovládat připojený radar, vizualizovat jeho výstupy, analyzovat data a ukládat je pro pozdější zpracování. Cílovou platformou jsou Microsoft Windows, jelikož jde o zdaleka nejrozšířenější operační systém, je tedy pro obsluhu nejsrozumitelnější (a pokud již na prodejně PC je, velmi pravděpodobně využívá právě tento OS). K vývoji aplikace jsem zvolil jazyk C# a jeho framework *Windows Presentation Foundation* (odůvodnění této volby viz úvod kapitoly Vizualizace). Umožňuje komunikaci s radarem přes sériovou linku, nastavení parametrů měření, zobrazení

dat v reálném čase i ukládání pro pozdější přehrání a zároveň generuje statistické informace, jako je časový vývoj obsazenosti zón zájmu či prostorové teplotní mapy. Závěrečná kapitola tohoto textu může zároveň posloužit jako určitá forma návodu k ovládání této aplikace.

1 Radary s milimetrovou vlnou

Radary s milimetrovou vlnou (mmWave) používají k detekci objektů elektromagnetický signál s vlnovou délkou v řádu milimetrů. Tomu odpovídá vysílání na frekvencích v řádech desítek GHz, v pásmu tzv. extrémně vysokých frekvencí (EHF). Vlnění se odráží od objektů a ze zachycených odrazů lze určit jejich polohu, velikost i rychlost. Díky krátké vlnové délce má měření velmi vysoké rozlišení, radar vysílající na frekvenci 60 GHz (tj. o vlnové délce 5 mm) dokáže detekovat posun o zlomek milimetru [3]. Navíc, jelikož potřebná velikost antény je přímo úměrná vlnové délce, použití mmWave technologie umožňuje použití velmi malých součástek a usnadňuje vytváření integrovaných řešení.

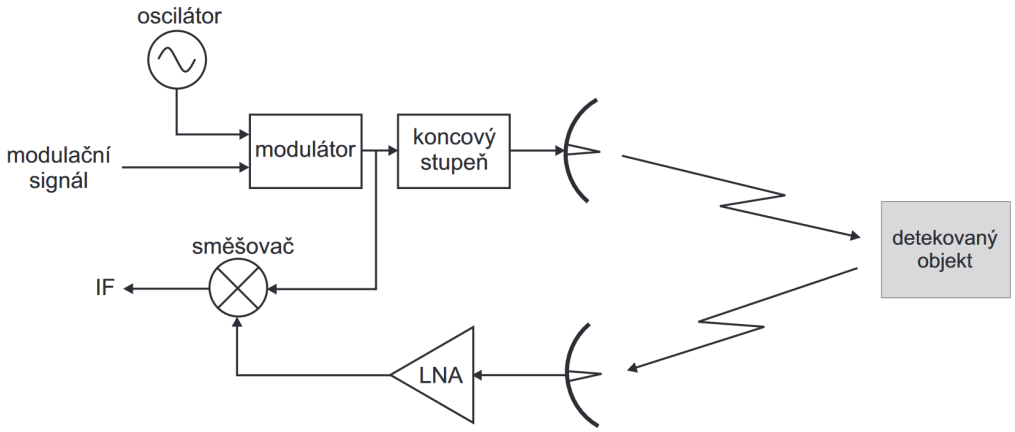
1.1 Radary s kontinuální vlnou

Použité radary mmWave využívají měření pomocí tzv. kontinuální vlny (*Continuous Wave – CW*). Zatímco při měření pulzním je vyslán vždy jeden elektromagnetický pulz, zachycen jeho odraz a na základě prodlevy určena vzdálenost objektu, radary s kontinuální vlnou vysílají i přijímají signál nepřetržitě. Výhodou oproti pulzním radarům je absence mrtvé zóny na začátku měření (způsobené u nich přepínáním mezi vysíláním a přijímáním signálu) a závislosti maximální jednoznačně změřitelné vzdálenosti na rozlišení v časové oblasti (u FMCW je tato vzdálenost závislá na periodě modulačního signálu). Aby ale ze zachyceného odrazu bylo možné rozeznat vzdálenost (případně rychlost) objektu, je nutné přidat další informace (jinak by nebylo možné zjistit, která část vysílaného signálu se vrátila, a tedy ani kdy a kde k odrazu došlo). V praxi se používají snímače založené na Dopplerově jevu a na frekvenční modulaci.

Radary na bázi Dopplerova jevu využívají změny frekvence signálů při odrazu od pohybujícího se objektu. Detekují také frekvenční změny v odraženém signálu, na základě kterých určují rychlost zachycených objektů. Nedokáží ale určit jejich polohu ani rozeznat nehybné objekty – pro využití v praktické části této práce jsou tedy nevhodné.

1.2 Radary s frekvenční modulací

Radary využívající tzv. frekvenčně modulovanou kontinuální vlnu (*Frequency Modulated CW – FMCW*) plynule mění frekvenci vysílaného signálu. Pomocí frekvence zachyceného odrazu tak dokáží určit, která část signálu byla odražena (a tedy jak dlouhá byla prodleva a kde se objekt nachází). Blokové schéma takového radaru zachycuje Obr. 1.1.



Obr. 1.1: Blokové schéma FMCW radaru [4].

Vysílaný signál vzniká v modulátoru z harmonického signálu a předdefinovaného modulačního signálu. Jako modulační signál je nejčastěji používán pilovitý průběh, který zajišťuje lineární závislost mezi změnou frekvence a vzdáleností. Perioda a amplituda modulačního signálu pak určuje průběh frekvenční charakteristiky, jak ukazuje graf 1.2. Frekvence periodicky stoupá od výchozí hodnoty f_i po cílovou f_f. Sklon charakteristiky S_m je určen poměrem šířky pásma (B_m) ku periodě modulačního signálu (T_m).

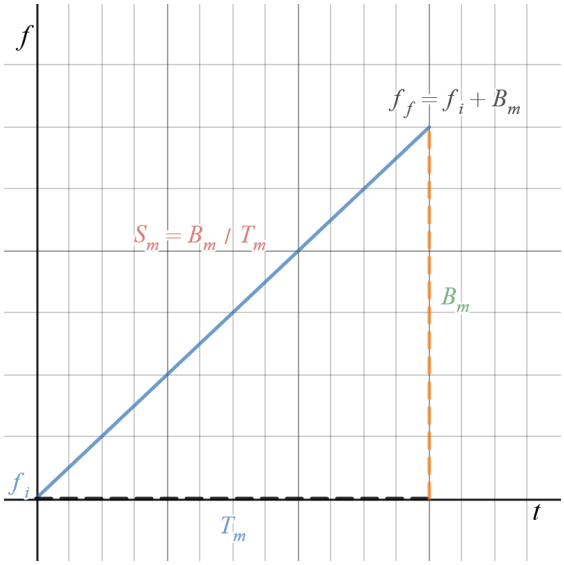
Modulovaný signál je následně odeslán vysílačem. Zachycené odrazy z přijímače jsou ve směšovači kombinovány s vysílaným signálem, čímž získáme tzv. mezifrekvenci (*Intermediate Frequency – IF*). Z toho jsme schopni určit polohu objektu, jak ukáže následující kapitola.

1.2.1 Výpočet vzdálenosti

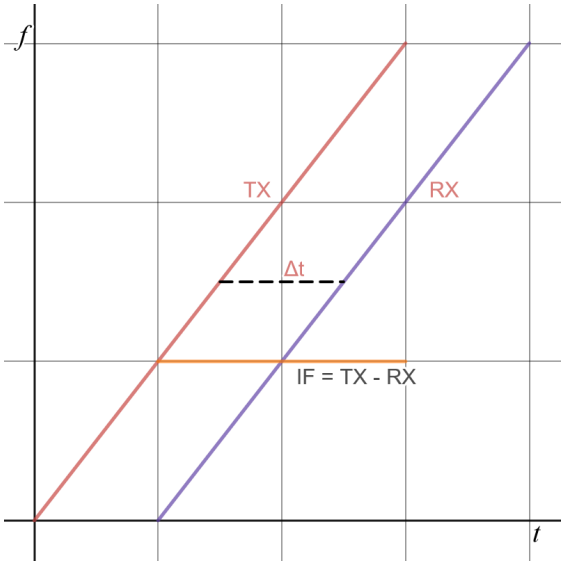
Obrázek 1.3 ukazuje srovnání frekvenční charakteristiky vysílaného (TX) signálu s odraženým (RX). Je patrné, že přijatý signál je posunutým obrazem vysílaného, posun je navíc přímo úměrný vzdálenosti objektu dle rovnice

$$\Delta t = \frac{2d}{c} \tag{1.1}$$

kde: Δt . . . . . posun signálů
d . . . . . vzdálenost objektu
c . . . . . rychlost světla



Obr. 1.2: Jedna perioda modulovaného signálu.



Obr. 1.3: Frekvenční charakteristika vysílaného (TX) a odraženého (RX) signálu.

Jelikož vzdálenost obou signálů je neměnná, frekvence rozdílového signálu IF je také konstantní.[1] Signál IF tak můžeme popsat následující rovnicí, ze které lze vyčíst vzdálenost objektu:[2,3]

$$\mathrm{IF} = A * \sin\left(2\pi f_0 t + \phi_0\right) \tag{1.2}$$

$$f_0 = \frac{2d * S_m}{c} \tag{1.3}$$

$$\phi_0 = 2\pi f_i \Delta t = \frac{4\pi d}{\lambda} \tag{1.4}$$

kde: S_m ..... sklon modulačního signálu
d ..... vzdálenost objektu
c ..... rychlost světla
f_i ..... frekvence nosného signálu
λ ..... vlnová délka nosného signálu
Δt ..... časový posun RX vůči TX

Situace je velmi podobná při detekci několika objektů současně. Mezifrekvenční signál pak bude složen z několika harmonických složek, z nichž každá svou frekvencí odpovídá jednomu ze zachycených objektů (a tedy jednomu z přijatých signálů RX). Pomocí Fourierovy transformace můžeme získat spektrální obraz mezifrekvenčního signálu, ze kterého opět vyčteme vzdálenosti jednotlivých objektů dle rovnice 1.2.

Abychom dokázali rozlišit mezi několika objekty v podobné vzdálenosti, musíme být schopni odlišit odpovídající vrcholy ve Fourierově transformaci. K tomu potřebujeme pozorovací okno, jehož perioda je menší než vzdálenost obou vrcholů. Naše pozorovací okno je dáno délkou modulačního signálu, platí tedy

$$\Delta f > \frac{1}{T_m} \tag{1.5}$$

kde: Δf ..... vzdálenost objektů ve frekvenční oblasti
T_m ..... perioda modulačního signálu

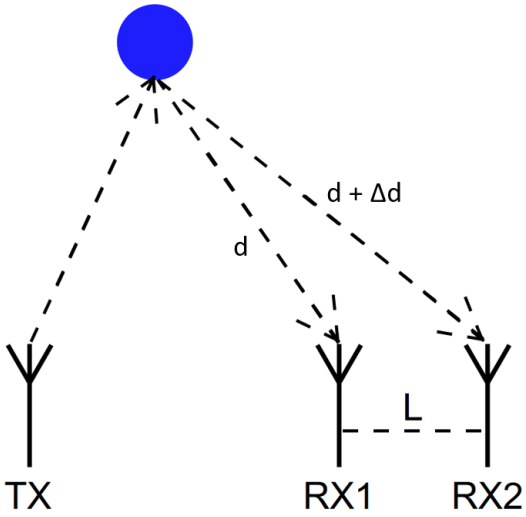
Z této rovnice můžeme aplikací rovnice 1.3 a definice S_m získat

$$\Delta d > \frac{c}{2 S_m T_m} = \frac{c}{2 B_m} \tag{1.6}$$

[1] Jeho definičním oborem je ale pouze oblast, kde jsou definovány signály TX i RX, viz Obr. 1.3

[2] Rovnice 1.3 nebere v potaz změny ve frekvenci způsobené Dopplerovým jevem. Jde o drobnou odchylku, která je v praxi kompenzována po spočtení rychlosti objektu (viz Výpočet rychlosti).

[3] Rovnice 1.4 je pouze aproximací platnou pro dostatečně nízké hodnoty S_m a d [3]. V dalších výpočtech ji ale počítáme především pro malé hodnoty $\Delta\phi$ (a tedy Δd), pro které je aproximace poměrně přesná.



Obr. 1.4: Měření úhlu na základě dvojice přijímačů.

kde: Δd ..... vzdálenost objektů
S_m ..... sklon modulačního signálu ($\frac{B_m}{T_m}$)
B_m ..... šířka pásma

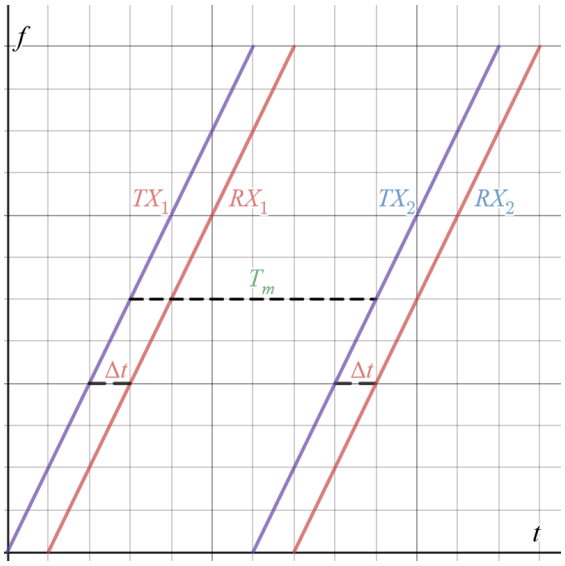
Je tedy zjevné, že rozlišovací schopnost je nepřímo úměrná použité šířce pásma. Například pro $B_m = 4$ GHz, se kterou pracuje radar použitý v praktické části práce, jsme schopni rozlišit objekty vzdálené od sebe přibližně $3,75$ cm.

1.2.2 Výpočet úhlu

Krom absolutní vzdálenosti objektů dokáže použitý FMCW radar určit i úhel, pod kterým se (vůči radaru) objekty nachází. Využívá k tomu kombinaci dvou či více přijímacích antén, jak ukazuje Obr. 1.4. Z výše uvedeného vzorce 1.4 vyplývá, že mezi odrazy zachycenými přijímači RX_1 a RX_2 vznikne fázový posun přímo úměrný hodnotě Δd. Aplikací trigonometrických pravidel na trojúhelník tvořený sledovaným objektem a anténami RX_1 a RX_2 navíc dokážeme určit, že pro úhel mezi osou radaru a sledovaným objektem θ platí $\Delta d = L * \sin(\theta)$. Pro úhel θ tak získáváme rovnici[4]

$$\theta = \arcsin \frac{\Delta\phi * \lambda}{2\pi L} \tag{1.7}$$

[4]Hodnota $\Delta\phi$ v rovnici 1.7 je závislá na hodnotě $\sin\theta$, rovnice tak obsahuje nelineární závislost. Lineární aproximace $\sin\theta \approx \theta$ platí pouze pro malé hodnoty θ, výpočet je tudíž přesnější pro předměty pod malými úhly [3]. To je důležité zejména při volbě vhodného umístění radaru v měřené oblasti – zóny zájmu by měly být co nejblíže k ose měření.



Obr. 1.5: Dvě periody modulovaného signálu.

kde: θ ..... úhel pozorovaného objektu vůči ose měření
$\Delta\phi$ ..... posun fáze mezi odrazy RX_1 a RX_2
λ ..... vlnová délka nosného signálu
L ..... vzdálenost přijímačů RX_1 a RX_2

Aby bylo měření úhlu jednoznačné, je nutné dodržet $|\Delta\phi| < \pi$. Dosazením do rovnice 1.7 získáme podmínku

$$\frac{2\pi L * \sin\theta}{\lambda} < \pi \tag{1.8}$$

$$\theta_{max} = \arcsin\frac{\pi * \lambda}{2\pi L} = \arcsin\frac{\lambda}{2L} \tag{1.9}$$

Funkce $\arcsin(x)$ má maximum v bodě $x = 1$, kdy dává výsledek $\arcsin(1) = \frac{\pi}{2}$. Maximální zorné pole má tak radar pro antény RX vzájemně vzdálené přesně o polovinu vlnové délky, a to 90° do obou směrů od osy měření.

1.2.3 Výpočet rychlosti

Pro určení rychlosti objektu lze využít dvě po sobě následující periody modulovaného signálu, které jsou od sebe vzdáleny T_m, jak ukazuje obrázek 1.5. Po provedení Fourierovy transformace dle minulé kapitoly pro oba signály získáme pro sledovaný objekt v obou charakteristikách signál RX o stejné frekvenci.[5] Odražené signály

[5] Aby se sledovaný předmět pohnul natolik, abychom v druhé Fourierově transformaci nepoznali jeho obraz, musel by se za jedinou periodu přemístit o vzdálenost řádově odpovídající alespoň

RX_1 a RX_2 ale budou mít odlišnou fázi – když v rovnici 1.4 dosadíme $\Delta d = v * T_m$, získáme pro rozdíl fází rovnici

$$\Delta\phi = \frac{4\pi v T_m}{\lambda} \tag{1.10}$$

kde: $\Delta\phi$ ..... posun fáze mezi odrazy RX_1 a RX_2
v ..... rychlost objektu
T_m ..... perioda modulačního signálu
λ ..... vlnová délka nosného signálu

Z této rovnice můžeme vyjádřit rychlost objektu jako

$$v = \frac{\Delta\phi * \lambda}{4\pi T_m} \tag{1.11}$$

Tato rovnice je jednoznačná pouze pro hodnoty $|\Delta\phi| < \pi$, rovnice 1.11 je tak vhodná pouze pro rychlosti menší než v_{max}, pro měření vyšších rychlostí je nezbytné použít radar s menší periodou.

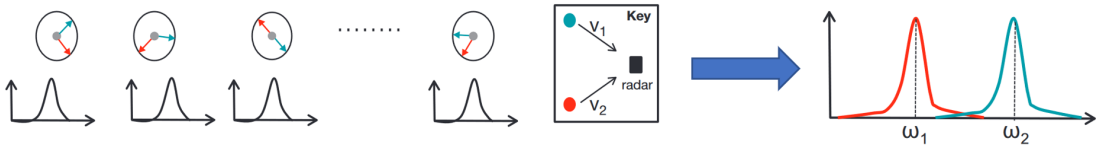
$$v_{max} = \frac{\pi * \lambda}{4\pi T_m} = \frac{\lambda}{4T_m} \tag{1.12}$$

Problematické je ale měření rychlosti s několika různými objekty ve stejné vzdálenosti. Jelikož všechny složky IF generované těmito objekty se při Fourierově transformaci sečtou (budou mít stejnou frekvenci, pouze odlišné fáze), výše uvedenou metodu nelze použít. Aby tento problém vyřešily, využívají FMCW radary tzv. okno, složené z N po sobě jdoucích period modulovaného signálu. Pro každý ze signálů ve zvoleném okně je následně spočítána Fourierova transformace, která pro dané ekvidistantní objekty vrací stejné frekvence, ale různé fázové posuny. Nad množinou takto získaných fázorů je následně provedena druhá Fourierova transformace (nazývaná Dopplerova), která oba objekty rozliší.

Celý postup zachycuje obrázek 1.6. V levé části jsou výsledky Fourierovy transformace pro jednotlivé periody obsažené v okně – všechny mají stejnou absolutní hodnotu, ale liší se jejich fáze (viz fázory nad jednotlivými grafy). Následuje aplikace Dopplerovy FFT na získané fázory, čímž získáme hodnoty ω pro jednotlivé objekty. Jejich rychlosti pak spočítáme obdobně jako v rci. 1.11:

$$v_i = \frac{\omega_i * \lambda}{4\pi T_m} \tag{1.13}$$

rozlišovací schopnosti radaru určené na konci předchozí podkapitoly. Aby k tomu došlo u radaru schopného rozlišit $\Delta d = 3,75$ cm používajícího $T_m = 40\ \mu s$, musel by se objekt pohybovat rychlostí v řádu stovek až tisíců m/s, což není pro zamyšlenou aplikaci relevantní. Měření s několika předměty v těsné blízkosti navíc naráží na závažnější problémy, zmíněné později v této kapitole.



Obr. 1.6: Výsledky FFT (vlevo) a následné Dopplerovy FFT (vpravo) [3].

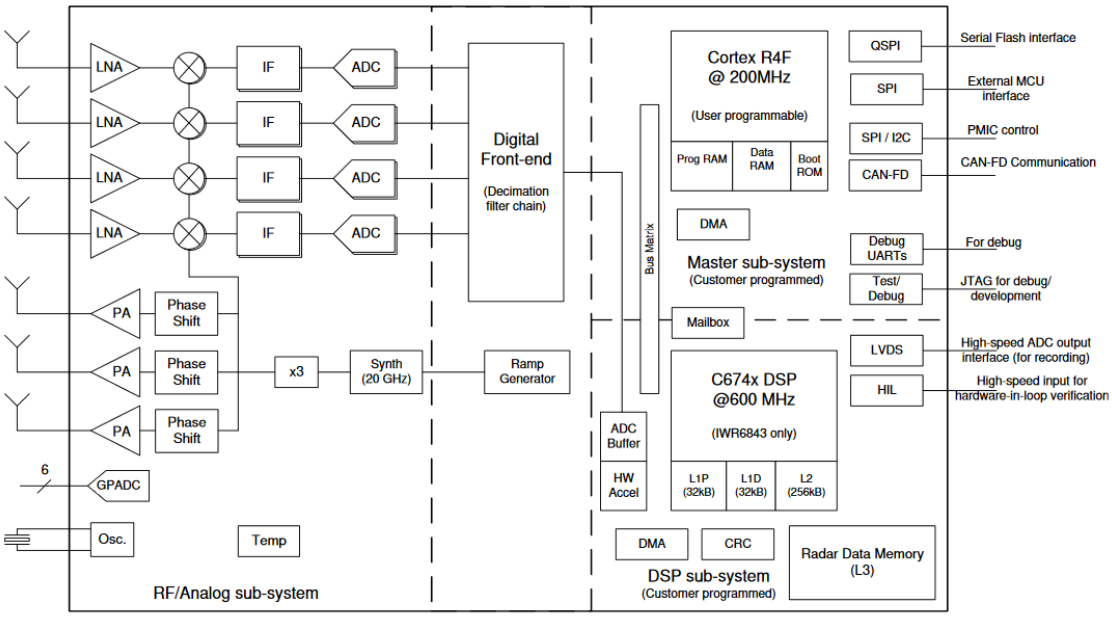
kde: v_i ..... rychlost objektu i
ω_i ..... výsledek Dopplerovy FFT pro objekt i
λ ..... vlnová délka nosného signálu
T_m ..... perioda modulačního signálu

1.3 Integrovaný senzor IWR6843

V praktické části práce byl využit přípravek IWR6843 firmy Texas Instruments. Ten obsahuje:

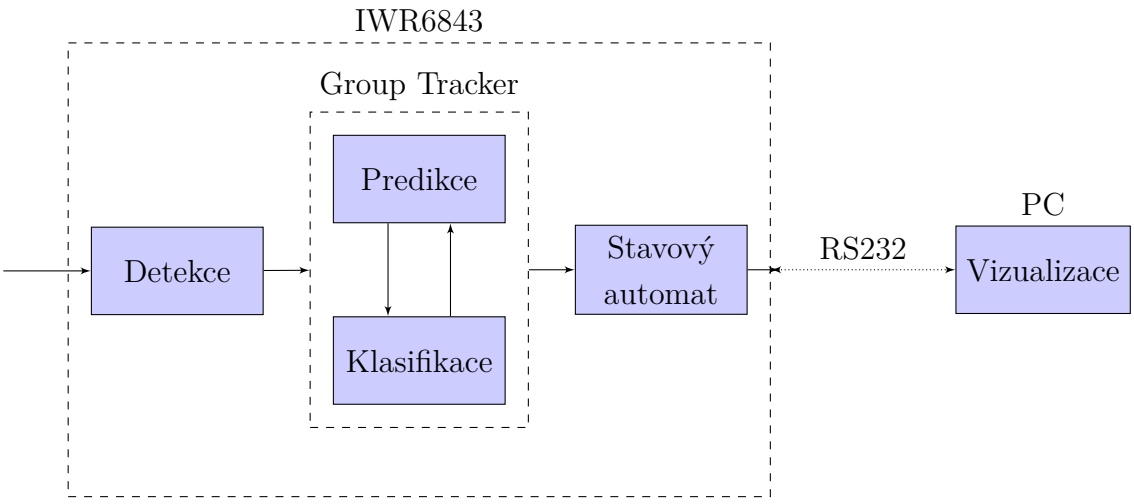
- integrovaný vysílač, přijímač, základní pásmo, fázový závěs,
- A/D převodník s rozlišením 12 bit,
- mikrokontroler architektury ARM s pamětí flash,
- modul DSP pro zpracování signálů,
- hardwarový akcelerátor pro FFT, filtraci a CFAR,
- paměť RAM o velikosti 1,75 MB rozdělenou do tří vrstev,
- oscilátor na frekvenci 40 MHz,
- rozhraní pro A/D převodník, SPI, UART, CAN, I2C, GPIO a LVDS.

Pro snímání je využit radar s frekvenčně modulovanou kontinuální vlnou, pracující v pásmu milimetrových vln. Jsou k dispozici tři vysílače a čtyři přijímače. Základní frekvence radaru f_i je 60 GHz, šířka pásma B_m 4 GHz [5].



Obr. 1.7: Blokové schéma IWR6843 [5].

Obr. 1.8: Radar IWR6843.



Obr. 2.1: Blokové schéma systému.

2 Zpracování dat a detekce osob

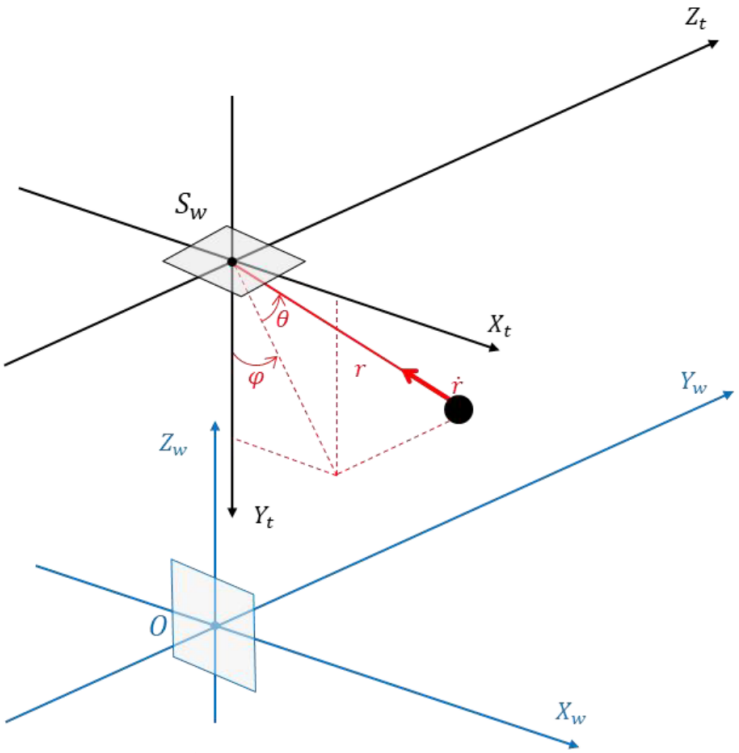
V této části práce popíši několik algoritmů, které lze využít při detekci osob na základě signálu získaného z mmWave radaru. Základní postup pro predikci a klasifikaci vychází z algoritmu *Group Tracker* [2], vyvinutého společností Texas Instruments. Nad tímto algoritmem je následně navržena nadstavba ve formě stavového automatu, který na základě předdefinovaných přechodů mezi zónami koriguje a vyhlazuje výsledky měření.

2.1 Popis systému

Celý systém pro detekci, vyhodnocení a vizualizaci se skládá z několika hlavních částí, jak ukazuje Obr. 2.1.

Naměřená data z radaru vstupují do modulu Detekce, který na základě odraženého signálu identifikuje jednotlivé body. Jde o nízkoúrovňové zpracování, jehož rozbor není součástí této práce, při implementaci byl použit firmware od výrobce radaru. Výstupem, který je předáván k dalšímu zpracování, je tzv. *Point Cloud* – množina bodů v „radarových souřadnicích“, tj. polárních souřadnicích se středem v senzoru (viz Obr. 2.2). Pro každý bod tak známe jeho vzdálenost, úhel svíraný s vertikální a horizontální osou radaru, sílu Dopplerova jevu a odstup signálu od šumu (*signal-noise ratio – SNR*). Poslední jmenovaná hodnota vyjadřuje, s jakou spolehlivostí byl daný bod změřen.

Detekované body jsou předány k dalšímu zpracování algoritmu *Group Tracker*. Ten ale pro snazší výpočty operuje v kartézských souřadnicích, nikoliv sférických,



Obr. 2.2: Souřadnicová soustava radaru [2].

které generuje radar. Pokud použijeme označení souřadnic zachycené na Obr. 2.2, můžeme na převod použít následující rovnici[1]

$$\begin{aligned} x &= r * \cos(\theta) * \sin(\varphi) \\ y &= r * \cos(\theta) * \cos(\varphi) \\ z &= r * \sin(\theta) \end{aligned} \tag{2.1}$$

Samotný algoritmus pro predikci a sledování objektů je postaven na bázi Kalmanova filtru, jehož fungováním se budu zabývat v následující podkapitole. Na základě prostorového rozložení bodů a jejich časového vývoje odhaduje počet a rozmístění cílů (osob), výstupem tohoto algoritmu je tzv. *Target List* – seznam identifikovaných cílů a jejich pozice, rychlost a zrychlení (již v kartézských souřadnicích), krom těchto

[1]Teoreticky můžeme rozlišovat dvě sady kartézských souřadnic – radarové kartézské souřadnice, kdy je v bodě (0, 0, 0) umístěn senzor radaru, a světové souřadnice, kdy je bod (0, 0, 0) projekcí radaru na úroveň země. Jelikož ale předpokládáme radar připevněný na stropě, paralelně se zemí, přepočet mezi oběma systémy je triviální – stačí přejmenovat osy, jak naznačují popisky grafu na Obr. 2.2, a následně upravit souřadnici z o výšku, ve které je radar upevněn. Výpočty v této kapitole pod veličinami x, y a z uvažují veličiny v souřadnicích radaru, v dalších kapitolách, zabývajících se vizualizací, už ale předpokládáme souřadnice světové (které jsou pro konfiguraci i obsluhu radaru intuitivnější).

veličin si každý cíl nese ještě kovarianční matici, která popisuje varianci jednotlivých parametrů.

Posledním stupněm zpracování signálu je stavový automat. Ten má předdefinované informace o zónách zájmu a jejich vstupních a výstupních oblastech a upravuje výsledky klasifikačního algoritmu na základě topologie reálného světa – působí jako brzda proti tomu, aby lidé „mizeli“ uprostřed zón, intuitivně vychází z předpokladu, že pokud někdo odejde z místnosti, pravděpodobně k tomu použije dveře. Tento automat ovládá výstupy ze systému, jako je spouštění alarmů pro jednotlivé zóny (viz kapitola Návrh stavového automatu).

Závěrečnou částí systému je vizualizace, určená pro PC systémy Windows. Ta přes sériovou linku přebírá informace ze všech tří úrovní firmwaru (detekované body, rozpoznané cíle i stavové informace), na základě kterých zobrazuje stav a dále analyzuje data. Tato část projektu už ale nespadá pod detekci osob, zabývá se jí až kapitola Vizualizace.

2.2 Kalmanův filtr

Vzhledem k tomu, že celý algoritmus se opírá fungování Kalmanova filtru, rád bych v následující podkapitole stručně shrnul, jak tento filtr funguje. Jde o metodu, která umožňuje na základě zašuměných měření a informací o stavu systému v čase t předpovídat stav v čase $t+1$ a následně jej upravit tak, aby vzal v potaz jak naši predikci, tak reálně naměřená data. Použitý algoritmus jej využívá ke sledování pohybujících se osob, jelikož nám umožňuje předpovědět jejich polohu a následně ji korigovat na základě nových signálů.

Kalmanův filtr vychází ze dvou základních informací: stavu vypočítaného na základě stavového popisu systému (rovnice 2.2) a získaného měřením. V každém kroku je předpovězen očekávaný stav systému (stav *a priori*), ten je porovnán s naměřenými daty a následně na základě spočítané variance upraven na tzv. stav *a posteriori*. Poměr mezi váhami *a priori* predikce a nového měření je vypočítán na základě kovariance obou metod, postup je tak vhodný i pro práci s ne zcela spolehlivými údaji (což data z radaru i odhady pohybu osob často jsou).

2.2.1 Predikce stavu

$$s'_k = A * s_{k-1} + B * u_{k-1} + w_k \tag{2.2}$$

kde: s_{k-1} ..... stav v čase $k-1$
u_{k-1} ..... vstup systému v čase $k-1$
s'_k ..... *a priori* odhad stavu v čase k
w_k ..... šum v čase k
A ..... matice systému
B ..... matice vazeb vstupu na stav

Předpokládejme nyní, že uvažovaným systémem je jeden z cílů, tj. sledovaná osoba. Stav systému jsou sledované veličiny – poloha, rychlost a zrychlení ve všech třech osách. Náš radar na stavové veličiny nijak nepůsobí, vektor u a matici B tak nemusíme uvažovat (negenerujeme do systému žádný vstup, který by ovlivňoval pohyb cíle). Matici A pak můžeme určit pomocí jednoduché fyzikální rovnice rovnoměrně zrychleného pohybu[2] (při predikci předpokládáme konstantní zrychlení, případné změny budou korigovány v aktualizačním kroku algoritmu). Stavový vektor a vypočítanou matici A zachycují rovnice 2.3 a 2.4.

$$s_k = \begin{bmatrix} x & y & z & \dot{x} & \dot{y} & \dot{z} & \ddot{x} & \ddot{y} & \ddot{z} \end{bmatrix}^T \tag{2.3}$$

$$A = \begin{bmatrix} 1 & 0 & 0 & t & 0 & 0 & \frac{t^2}{2} & 0 & 0 \\ 0 & 1 & 0 & 0 & t & 0 & 0 & \frac{t^2}{2} & 0 \\ 0 & 0 & 1 & 0 & 0 & t & 0 & 0 & \frac{t^2}{2} \\ 0 & 0 & 0 & 1 & 0 & 0 & t & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & t & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & t \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.4}$$

Ve stavové rovnici filtru nám tak zbývá poslední neznámý prvek – šum w_k. Ten představuje kovarianční matice velikosti 9x9, která vyjadřuje šum v odhadu jednotlivých parametrů – má tak zásadní vliv na fungování filtru (menší šum odhadu znamená, že při výpočtu *a posteriori* stavu dáme větší váhu predikci a menší měření, větší šum naopak posiluje naměřené hodnoty). K modelování je využit bílý šum s normálním rozložením, více informací k jeho odhadu viz například příloha 6.4.3 v [2].

[2] $x = x_0 + v_0 t + \frac{1}{2} a t^2 \mid v = v_0 + at \mid a = konst.$

2.2.2 Upravení odhadu

Z výpočtu popsaného v předchozí kapitole můžeme získat predikci stavových veličin s'_k. Zároveň máme k dispozici nově naměřené hodnoty m_k, které jsou vyjádřeny ve sférických souřadnicích radaru:

$$m_k = \begin{bmatrix} r & \varphi & \theta & \dot{r} \end{bmatrix}^T \tag{2.5}$$

Abychom ale mohli porovnat měřenou hodnotu s *a priori* stavem, musíme být schopni převést data z kartézských do sférických souřadnic. K tomu využijeme následující matici H, která vyjadřuje transformaci souřadnic:

$$H(s_k) = \begin{bmatrix} \sqrt{x^2+y^2+z^2} \\ \arctan(z, \sqrt{x^2+y^2}) \\ \frac{x\dot{x}+y\dot{y}+z\dot{z}}{\sqrt{x^2+y^2+z^2}} \end{bmatrix} \tag{2.6}$$

$$\arctan(a,b) = \begin{cases} \arctan(\frac{a}{b}) + \pi & \text{pro } b < 0 \\ \frac{\pi}{2} & \text{pro } b = 0 \\ \arctan(\frac{a}{b}) & \text{pro } b > 0 \end{cases} \tag{2.7}$$

Na základě znalosti *a priori* stavu a skutečných hodnot nyní můžeme spočítat residua jako rozdíl těchto dvou hodnot, tedy

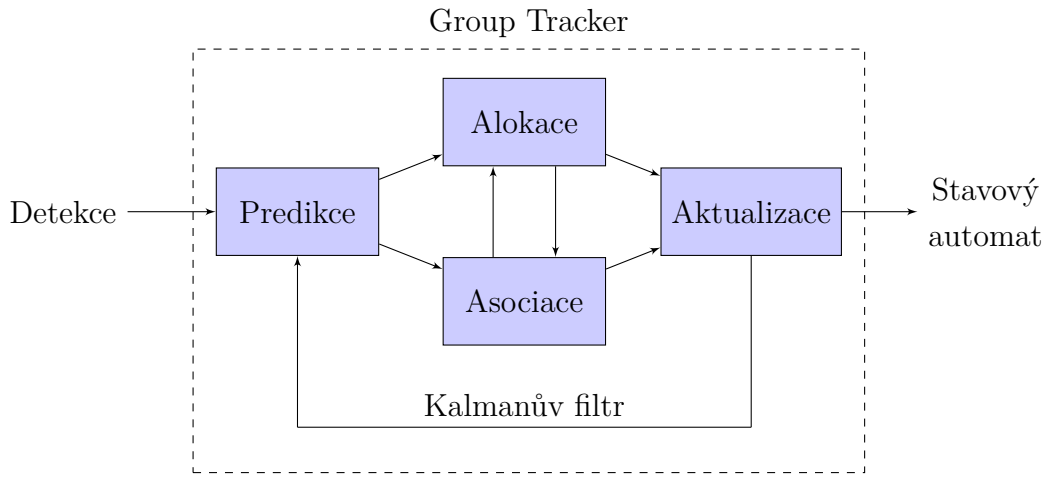
$$e_k = u_k - H(s'_k) \tag{2.8}$$

Zbývá nám určit, jestli máme více důvěřovat datům z výpočtu či z měření (tj. jak velkou část residua máme k *a priori* výsledkům přičíst). K tomu použijeme Kalmanův zisk K, vypočtený na základě residuální kovarianční matice C_k a *a priori* odhadu kovariance P'_k.

$$K_k = P'_k * J_H^T(s'_k) * C_k^{-1} \tag{2.9}$$

$$s_k = s'_k + K_k * e_k \tag{2.10}$$

Výpočet matic P a C je ovšem poměrně komplikovaný, jelikož transformace H je nelineární a při výpočtu kovariance je tak nutno použít tzv. rozšířený Kalmanův filtr, který využívá lineární aproximaci rozvojem do Taylorovy řady (odtud pochází také Jacobiho matice J_H v rovnici 2.9). Pro pochopení algoritmu však není konkrétní postup výpočtu klíčový, stačí nám vědět, že velikost aktualizace se odvíjí od spolehlivosti (kovariance) měřených a spočtených hodnot, jak ukazuje rovnice 2.10. V případě zájmu tak pro celý postup výpočtu doporučuji pročíst [2], kapitoly 1.2 a 2.5, případně i odvození Jacobiho matice v příloze 6.2 tamtéž.



Obr. 2.3: Blokové schéma sledovacího algoritmu.

2.3 Sledování

Klasické algoritmy pro sledování objektů obvykle používají dva moduly – jeden pro sledování bodů v prostoru (*clustering*), druhý v čase (*tracking*). Zvolený algoritmus *Group Tracker* ale kombinuje oba moduly v jeden [2].

Základ sledovacího algoritmu tvoří Kalmanův filtr, který byl popsán již v předchozí podkapitole (v diagramu bloky Predikce a Aktualizace). Tento filtr ovšem dokáže pracovat pouze s časovým vývojem cílů, výstupem radarového měření je ale množina bodů, z nichž je nutné cíle teprve poskládat. Proto jsou ke Kalmanovu filtru přidány ještě bloky Asociace a Alokace, které zajišťují přiřazování bodů ke sledovaným cílům a případně dynamické vytváření cílů nových.

2.3.1 Popis algoritmu

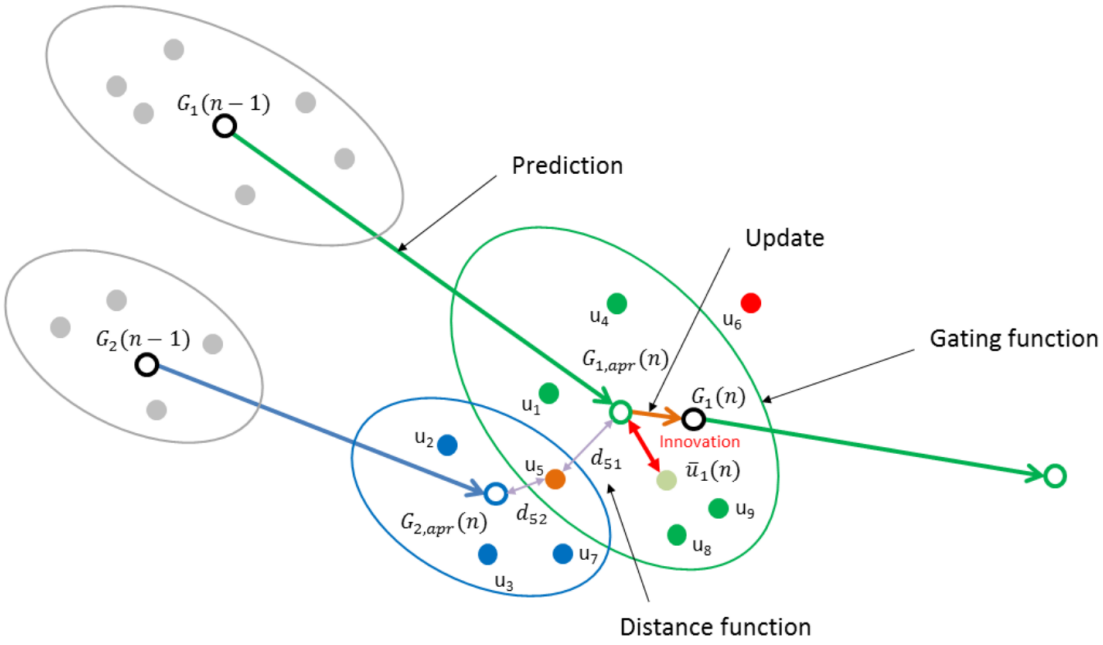
Algoritmus 1 Sledování cílů

Vstup: Sledované cíle v čase $k-1$.

Vstup: Naměřené hodnoty v čase k.

Výstup: Sledované cíle v čase k.

1: Predikce (Kalmanův filtr – *a priori* odhad cílů)
2: **while** ∃ nezpracované body **do**
3: Asociace (přiřazení všech bodů uvnitř hraničních funkcí k cílům)
4: Alokace (vytvoření nového cíle, pokud nám k tomu zbývají body)
5: **end while**
6: Aktualizace (Kalmanův filtr – upravený *a posteriori* odhad)



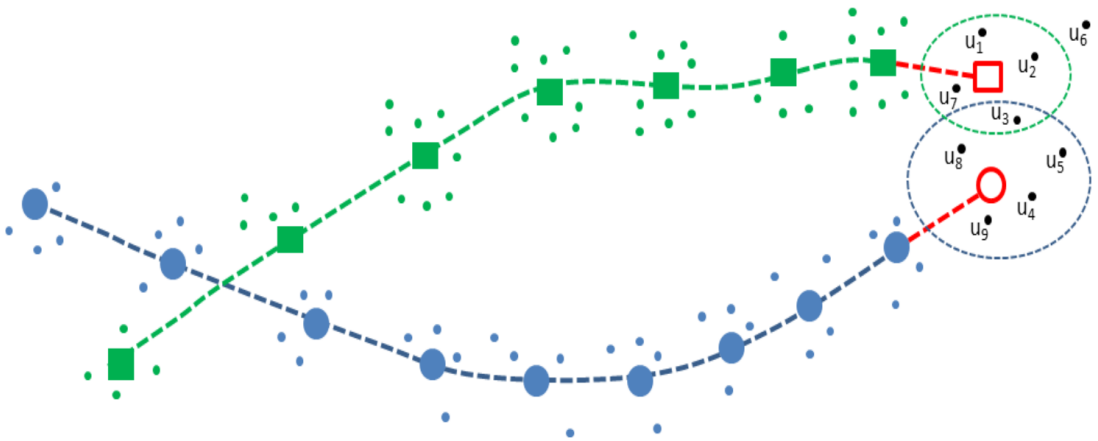
Obr. 2.4: Predikce, asociace a aktualizace sledování [2].

Prvním krokem každé iterace je **predikce**. Na základě výsledků z iterace minulé odhadneme, kam se za uplynulou dobu cíle posunuly – použijeme k tomu *a priori* odhad Kalmanova filtru popsaný v předchozí kapitole.

Následně musíme převést naměřené body na nové pozice cílů. Použijeme k tomu blok **asociace**, který na vstupu krom měřených bodů dostane i nové *a priori* odhady, každý cíl je přitom specifikován svým středem hmoty (G_i) a hraniční funkcí, což je elipsa se středem v G_i, která vychází z prostorového rozložení hmoty, odstupu signálu od šumu a kovariance měřených parametrů (pro grafické znázornění cílů i bodů viz Obr. 2.4). Při asociaci nejprve označíme všechny body, které jsou uvnitř hraniční funkce alespoň jednoho z cílů. Každý bod následně přiřadíme k tomu z cílů, v jejichž hranicích se nachází, k jehož středu má nejmenší vzdálenost (viz bod u_5).

Asociace ovšem nemusí přiřadit všechny body – viz např. u_6 ve schématu. Může to být tím, že jde pouze o falešný signál (několikanásobné odrazy, pohyb menších objektů, které nás nezajímají), ale může jít i o novou osobu, která právě vstoupila na scénu. K rozlišení těchto případů slouží blok **alokace**, který prochází nepřiřazené body a testuje, jestli by nějaká jejich podmnožina mohla definovat nový cíl.

Při alokaci si nejprve náhodně zvolíme jeden z nepřiřazených bodů jako nový střed hmoty. Následně procházíme volné body v jeho blízkosti a testujeme, jestli by mohly být součástí stejného cíle: pokud ano, přidáme je ke zvolenému bodu a přepočítáme nový střed hmoty, pokračujeme hledáním dalšího bodu. Pokud tímto způsobem dokážeme najít dostatek sdružených bodů na to, aby mohly vytvořit



Obr. 2.5: Ilustrace sledovacího algoritmu [2].

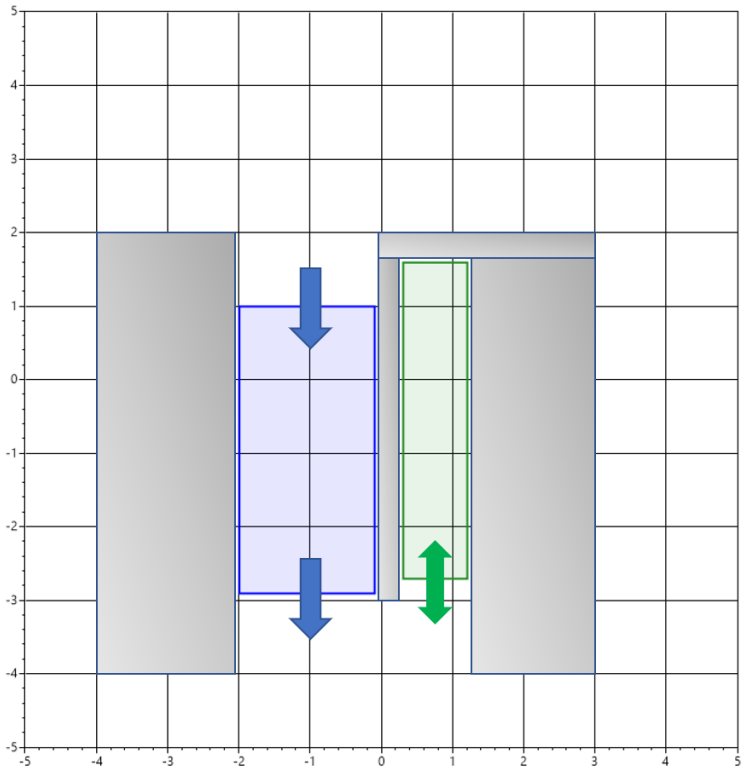
cíl, který by splňoval naše požadavky (vyjádřené pomocí prostorového rozložení a celkového SNR použitých bodů), přidáme jej do seznamu cílů a v bloku asociace přepočítáme nové přiřazení bodů. Pokud je hledání cíle neúspěšné, vyzkoušíme jiný z nepřiřazených signálů, pokud již další nemáme, alokovat nový cíl nelze a přejdeme k **aktualizaci**.

Posledním krokem algoritmu je syntéza predikovaných a měřených dat. K tomu použijeme opět Kalmanův filtr, který na základě odhadovaných (predikce) a měřených (na základě bodů přiřazených během asociace) veličin vytvoří nový, upravený odhad stavu i kovariance. Ten je výstupem algoritmu a zároveň vstupem pro predikci v následující iteraci.

2.4 Stavový automat

Poslední částí monitorovacího systému je stavový automat. Ten, na rozdíl od předchozích vrstev, může zohlednit i topologii prostoru (pomocí předkonfigurovaných zón, vytvořených při instalaci radaru). Dokáže tak vzít v potaz fyzikální podstatu sledovaných bodů – lidé se v místnosti nezjevují a nemizí náhodně, ale pravděpodobně do zón vstupují a opouštějí je v přesně definovaných místech. Zákazník nejspíše do obchodu vstupuje dveřmi, nikoliv podlahou, stejně tak pokladní nebude cestou k pokladně přeskakovat pokladní pás.

Druhou funkcí automatu je modelování určité setrvačnosti systému. Ta opět vyplývá z reality zamýšleného použití. Například při nasazení radaru pro přivolání pokladní ve chvíli, kdy nikdo neobsluhuje pokladny a ve frontě stojí zákazník, není ideální spouštět alarm okamžitě, jakmile k této situaci dojde. Může jít o chvilkový výpadek sledování (pokladní se nehýbe, splynula radaru s vybavením místnosti,



Obr. 2.6: Vliv topologie prostoru na pohyb osob.

během pár sekund ji opět zaznamenáme), také je možné, že obsluha stojí kousek od pokladny a během chvilky přijde sama. Nejspíše by ji nepotěšilo, kdyby se jí každých pár minut spouštěl alarm, aniž by k tomu měl pádný důvod. Vhodně sestaveným automatem tak můžeme ošetřit podobné situace, které při provozu předpokládáme, a optimalizovat tím běh algoritmu – více viz samotná implementace systému.

Využití topologie sledovaného prostor zachycuje Obr. 2.6. Neprůchodné objekty (zdi, regály, pokladní pás) jsou znázorněny šedě, modrá zóna představuje oblast pro zákazníky, zelená pro pokladní. Z nákresu je patrné, že pokud je například pokladní v horní části své zóny (na souřadnicích $(1, 1)$), její zmizení z výstupu sledovacího algoritmu bychom neměli interpretovat jako opuštění zóny.

Na topologii se ovšem nemůžeme spoléhat stoprocentně – je možné, že původní detekovaný signál byl falešný (odrazy, nezajímavé objekty), že nám sledování dané osoby vypadlo dostatečně dlouho na to, aby mezitím zónu opravdu opustila, aj. Proto jsou přechodné stavy ošetřené čítači – pokud nastane situace, kterou nepovažujeme za standardní (prodavačka zmizela do zdi), počkáme určitou dobu, jestli tuto situaci sledovací část programu sama nekoriguje, a až pokud k tomu ve stanoveném čase nedojde, vyhodnotíme změnu (pokladna je prázdná).

Obr. 2.7: Blokové schéma alternativního algoritmu.

2.5 Alternativní algoritmy

Postup sledování postavený na algoritmu *Group Tracker* samozřejmě není jediným možným. Při implementaci jsme sice ověřili, že naše požadavky splňuje dostatečně, i tak bych zde ale rád zmínil pár dalších možností, jak radarový signál monitorovat.

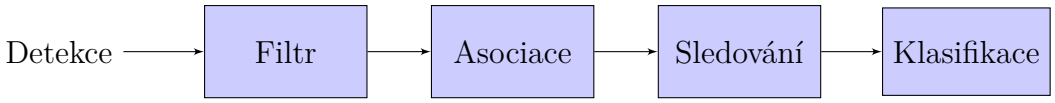
2.5.1 Rekurzivní Kalmanův filtr

Huang et al. ve svém výzkumu textitIndoor Detection and Tracking of People Using mmWave Sensor [6] navrhují systém, který je svou obecnou strukturou podobný výše popsanému algoritmu, nicméně některé kroky jsou v něm pozměněny, případně optimalizovány. Velmi obecné schéma algoritmu ukazuje Obr. 2.7.

První změnou je zavedení bloku pro filtrování statických objektů. K tomu je využita rychlá Fourierova transformace (FFT) na hodnotách vyčtených z přijímacích antén radaru, následuje pak odstranění stejnosměrné složky transformovaného signálu. To umožňuje odstranit ze záznamu nehybné objekty, které pravděpodobně představují vybavení místnosti a nejsou pro nás zajímavé.

Ke druhé zásadní změně dochází v bloku Sledování. Ten je sice opět postaven kolem Kalmanova filtru, využívá ovšem jinou z jeho variant. Jak jsem zmiňoval v kapitole pojednávající o tomto filtru, podstatný problém představuje nelineární závislost mezi naměřenými údaji a sledovaným stavem. Jelikož Kalmanův filtr je ve své podstatě lineárním systémem (pracuje na bázi metody nejmenších čtverců), s nelineární závislostí se nedokáže vypořádat. *Group Tracker* tento problém řeší využitím rozšířeného Kalmanova filtru, který lineárně aproximuje zkoumanou závislost, tím však zavádí do výpočtu aproximační odchylky a v některých případech může vést i k nestabilním stavům. Autoři tohoto algoritmu se proto rozhodli pro jinou variantu – Kalmanův filtr rekurzivní. Ten místo lineární aproximace používá k výpočtu zesílení K_k rekurzivní algoritmus, který se snaží minimalizovat odchylky. Výhodou je větší robustnost a snazší výpočet.

Autoři své řešení experimentálně porovnávají i s algoritmem *Group Tracker* (GT). Dle jejich výsledků dosahují podstatně lepších výsledků ve chvíli, když se v relativně malé oblasti pohybuje více osob (při pěti lidech uvádí spolehlivost 65 % pro svůj algoritmus a 45 % pro GT). Nicméně, vzhledem k tomu, že v našem po-



Obr. 2.8: Blokové schéma algoritmu pro klasifikaci.

užití řešíme primárně binární charakteristiky (v zóně je / není jeden či více lidí), případně agregované statistiky, přesnost sledování jednotlivců při plných prostorech pro nás nebyla natolik zásadní. Pro rozeznání samostatně stojícího člověka jsou ale výsledky prakticky identické (autoři uvádí 98 % pro svůj algoritmus, 96 % pro GT), rozhodl jsem se tedy zůstat u algoritmu GT, jehož podstatnou výhodou je, že základní implementaci již k firmwaru dodává výrobce, firma Texas Instruments.

2.5.2 Neuronové sítě

Další alternativou, kterou uvádí například Pegoraro et al. v práci *Multiperson Continuous Tracking and IdentificationFrom mm-Wave Micro-Doppler Signatures* [7], je využití strojového učení pro zlepšení výsledků. Celková struktura algoritmu je opět velmi podobná – začíná odstraněním statických objektů (tentokrát pomocí diskrétní Fourierovy transformace), následuje *clustering*, který rozdělí body mezi jednotlivé cíle a provede asociace, sledovací stupeň opět zajišťuje Kalmanův filtr. Novým přídavkem je ale blok poslední – klasifikace.

Tento blok umožňuje jednak rozdělení cílů do různých podkategorií, které může systém rozlišovat a případně statisticky analyzovat, jednak dlouhodobé sledování konkrétních jedinců (které může fungovat i tehdy, když daný jedinec nějakou dobu zmizí ze záběru). V rámci rozlišení podkategorií můžeme odhalit také falešná pozitiva, filtrovat signály od jiných objektů než lidí (které nás pravděpodobně nezajímají) a podobně. K tomu autoři navrhli poměrně složitou architekturu konvoluční neuronové sítě s enkodérem a dekodérem, která do značné míry připomíná systémy používané na automatickou klasifikaci obrazu.

Tento výzkum poskytuje velmi zajímavé výsledky, pravděpodobně by nám umožnil dosáhnout vyšší přesnosti při větším počtu souběžných cílů a poskytoval detailnější informace o jednotlivých zákaznících. Na druhou stranu, pro naše potřeby by šlo do značné míry o zbytečně silný nástroj – jak jsem zmiňoval již v předchozí podkapitole, zajímají nás především binární informace o obsazenosti zón, sledování konkrétních zákazníků také nevyužíváme.[3] Podstatnou nevýhodou tohoto systému

[3]Při sledování nás nezajímají informace o žádném konkrétním, jednotlivém zákazníkovi, ale pouze agregovaná data o celkovém toku osob. Navíc, jak bylo řečeno v úvodu práce, vzhledem k ochraně osobních údajů pro nás případná personalizace sledování nemá význam.

by pak byla výpočetní složitost – hluboká konvoluční síť by měla na výkon výrazně vyšší požadavky, než jednodušší algoritmy, a je tak otázkou, jestli by pro zpracování v reálném čase jednoduchý mikroprocesor vůbec stačil.

3 Návrh stavového automatu

V této části práce popíši návrh stavového automatu, který by modeloval zamýšlené primární využití vytvořeného programu – přivolání obsluhy v obchodě. Modelová situace odpovídá příkladu uvedenému v jedné ze sekcí předchozí kapitoly (sekce Stavový automat), jednoduché schéma monitorované oblasti viz Obr. 2.6 tamtéž.

Zásadní situace, kterou chceme ošetřit, je prázdná pokladna a neprázdná fronta. V té chvíli musíme spustit alarm (např. zvonek v odpočinkové místnosti pro zaměstnance, hlášení do sluchátek či podobně, samotná implementace závisí na charakteru a zařízení prodejny). Jak zóna pokladních, tak zóna fronty jsou definovány jako obdélníkové zájmové zóny, pro rozšíření systému (více pokladen, detekce dlouhé fronty atp.) či modelování složitější topologie (zóny nemožné obsáhnout jedním obdélníkem) lze poměrně jednoduše využít kombinaci většího počtu zón – je to ale rozšíření nad rámec tohoto základního algoritmu.

3.1 Použité symboly a označení

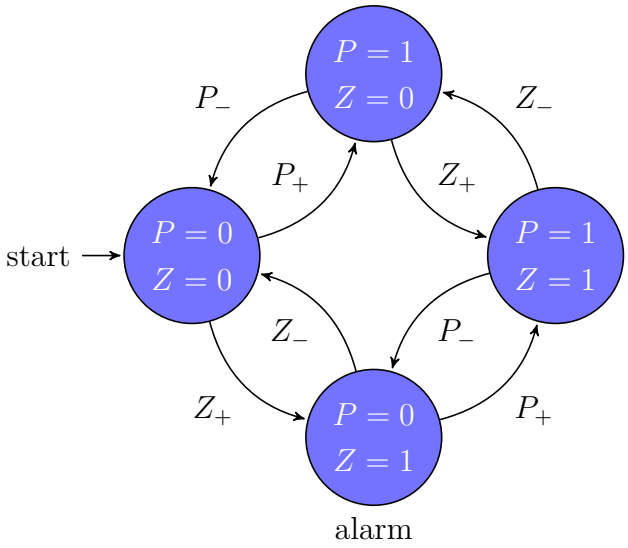
- $P = 0$ ($Z = 0$): Zóna pokladna (respektive zákazníci) je prázdná.
- $P = 1$ ($Z = 1$): Zóna pokladna (respektive zákazníci) je neprázdná.
- P_+ (Z_+): Do odpovídající zóny přišla první pokladní (respektive první zákazník).
- P_- (Z_-): Z odpovídající zóny odešla poslední pokladní (respektive poslední zákazník).
- T <N>s: Při vstupu do stavu je spuštěn časovač, nastavený na N sekund.[1]
- $T.DN$: Přechod je aktivní pouze po doběhnutí časovače.
- alarm: V daném stavu je aktivní mechanismus pro přivolání pokladní.

3.2 Postup návrhu

Nejjednodušší variantu stavového automatu zachycuje Obr. 3.1. Představuje automat o čtyřech stavech, které pokrývají možné kombinace obsazenosti jednotlivých zón. Ve stavu, kdy je u pokladny zákazník bez obsluhy, je spuštěn alarm.

Při testování tohoto modelu v reálném provozu jsme narazili na dva větší problémy. Zaprvé, radar není stoprocentně spolehlivý – čas od času došlo k tomu, že pokladní se ztratila ze záběru, i když v zóně stále byla. To vedlo k tomu, že alarm se poměrně pravidelně spouštěl i tehdy, když nebyl potřeba. Radar sice pokladní

[1] Uvedené časové údaje zhruba vychází z testování v provozu, i tak jsou ale primárně ilustrativní. Jejich přesné nastavení je možné provést na základě konkrétní situace v místě nasazení.



Obr. 3.1: Návrh jednoduchého stavového automatu.

po chvilce opět zaznamenal a alarm deaktivoval, opakované spouštění a vypínání alarmu ovšem bylo pro zaměstnance nepříjemné.

Druhým problémem bylo, že automat se spouštěl okamžitě ve chvíli, když se zákazník objevil. Prodavač přitom mohl stát kousek od pokladen, vyskladňovat zboží v nejbližším stojanu, všiml si jeho příchodu a je již na cestě – na událost tak reagoval sám, spuštění alarmu bylo opět nadbytečné.

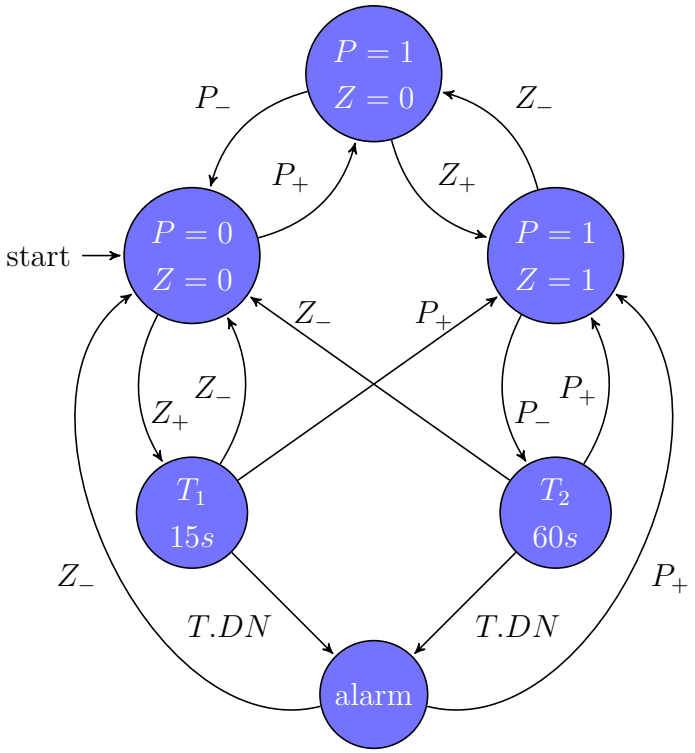
3.2.1 Automat s pamětí

Řešením těchto problémů byla druhá verze automatu, kterou zobrazuje Obr. 3.2. Ten před spuštěním alarmu nějakou dobu čeká, čímž se snaží předejít problémům jednoduchého automatu.

Stav T_1 představuje situaci, kdy zákazník přišel k prázdné pokladně. V tomto stavu používáme prodlevu poměrně krátkou, jelikož minimalizace čekání zákazníka je hlavním smyslem našeho programu. Krátká prodleva dává obsluze čas na vlastní reakci, bez upozornění, ale její prodloužení by vedlo k znatelnému zdržení ze strany zákazníka. Radši tak volíme kratší interval, který riskuje (relativně vzácná) zbytečná volání, než prodloužit čekání při každém legitimním spuštění.

Druhý časovač, T_2, představuje situaci, kdy obsluha zmizela ze zóny ve chvíli, když obsluhovala zákazníka. V této situaci je ale extrémně nepravděpodobné, že by prodavač opravdu zničehonic zákazníka nechal stát opuštěného u pokladny a odešel bezdůvodně pryč.

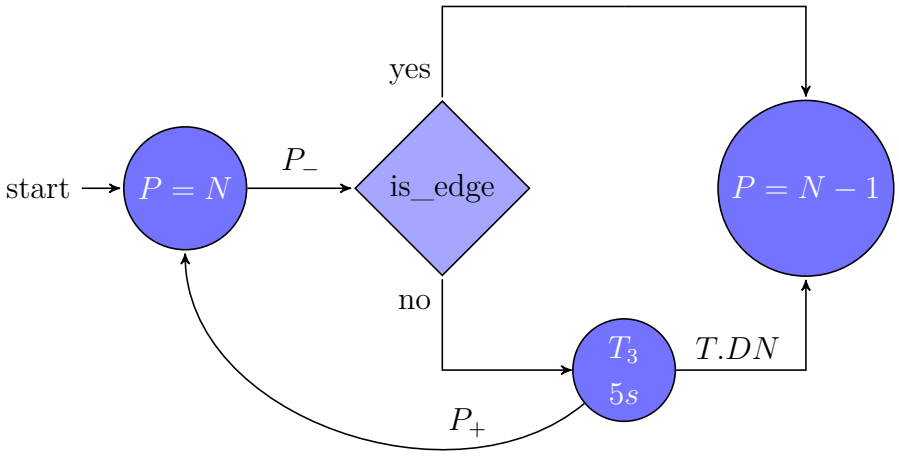
Předkládáme tedy dvě vysvětlení, která považujeme za výrazně pravděpodob-



Obr. 3.2: Návrh stavového automatu s pamětí.

nější. První možností je, že došlo k chybě ve sledovacím modulu. Pokladní zmizela ze záznamu například proto, že splynula s některým ze statických objektů (nejčastěji se samotnou pokladnou a nákupním pásem), nebo je v záběru velké množství falešných odrazů, které radar zmátly. Druhou možností je, že prodavač měl závažný důvod, proč zónu opustil – zákazník potřeboval s něčím pomoci, v obchodě došlo k nějakému problému, jenž vyžadoval jeho přítomnost, apod. Ani v jednom z těchto případů nechceme spouštět alarm, jelikož prodavač buď stále zákazníka obsluhuje, nebo má možnost náhradu přivolat manuálně. V tomto stavu je tedy časovač velmi dlouhý, aby systém dostal šanci napravit případné chyby či prodavač vyřešit vše potřebné.

Teoreticky by do systému mohlo být vhodné přidat ještě jeden časovač – na přechodu mezi stavy „alarm“ a „$P = 0$ $Z = 0$“. Odchod zákazníka od pokladny je poměrně zvláštním úkazem, který mohl být velmi snadno důsledkem chvilkového výpadku sledování. Přechodem do výchozího stavu ovšem dojde k tomu, že vynulujeme čítač T_1 – i výpadek po dobu jediného snímku tak může vést k zdržení o dalších 15 sekund. Tento problém ovšem elegantněji vyřeší úpravy popsané v následující podkapitole, které podobné výpadky řeší v obecné rovině.



Obr. 3.3: Zpracování příchodu pokladní.

3.2.2 Automat se znalostí topologie

Složitější automat z předchozí kapitoly se v reálném provozu osvědčil dobře, většina (ne)spuštění alarmu již byla smysluplná. Přesto jsme měli k dispozici ještě jednu informaci, se kterou jsme zatím nepracovali – fyzickou topologii prodejny. Podstatnou část snímané oblasti pravděpodobně představují překážky, které jsou za běžných okolností neprostupné (zdi, regály, pokladní pás...). Ty nám dávají další informaci o možném toku osob prodejnou – určují přirozené bariéry, skrz které pravděpodobně nikdo nevstupuje ani neodchází. Existující zóny tak můžeme rozdělit na několik částí, z nichž některé označíme jako vstupně/výstupní, jiné jako vnitřní.

Pokud pak zaznamenáme vstup osoby do zóny zájmu (či její výstup), můžeme zkontrolovat, jestli k němu opravdu došlo skrz vstupně/výstupní oblast. Pokud ano, vyhodnotíme změnu jako validní, jestliže ne, nejspíše jde o chybu sledování – zákazník zmizel uprostřed obchodu, případně odešel zdí. Takovou událost ovšem nemůžeme ignorovat úplně – je možné, že systém cíl ztratil na dobu dostatečnou k tomu, aby zónu opravdu opustil, nebo že chybná byla už předchozí informace, že v zóně někdo je (např. kvůli falešným odrazům, vyhodnocení špatného objektu, např. nákupního vozíku, jako osoby...).

K ošetření těchto situací proto opět použijeme časovače, podobně jako u předchozího systému. Diagram ostatně zůstane velmi podobný jako v předešlé podkapitole, pouze události příchodu a odchodu rozdělíme na validní (cíl prošel okrajovou zónou) a sporné (cíl zmizel ve vnitřní oblasti nebo se v ní naopak záhy objevil). Samotný automat tak bude mít stejný charakter, jako měl Obr. 3.2 z předchozí kapitoly, změní se pouze zpracování příchodů a odchodů.

Podsystém pro vyhodnocování příchodů a odchodů zachycuje Obr. 3.3.[2] Na rozdíl od předchozích modelů už ale nepoužívá binární hodnocení přítomnosti (prázdná / neprázdná zóna), ale počítá přesný počet osob v dané zóně.[3]

Při příchodu pokladní ověříme, jestli opravdu přišla skrz hraniční zónu (tj. jestli její pozice není v oblasti označené jako „vnitřní“). Pokud ano, příchod okamžitě zaznamenáme a nadřazenému automatu oznámíme změnu v zóně. Pokud ovšem ke změně došlo ve vnitřní zóně, předpokládáme, že s velkou pravděpodobností šlo pouze o chybu ve sledování. Změnu proto dočasně nepropagujeme a spustíme časovač T_3. Pokud za jeho běhu pokladní opět detekujeme, vyhodnotíme změnu jako falešnou a vrátíme se do výchozího stavu. Pokud časovač doběhne a změna stále trvá, odchod uznáme jako platný a propagujeme výše.

Tento přístup řeší problém nastíněný na konci předchozí podkapitoly – nyní, pokud na chvíli ztratíme sledovaného zákazníka, neresetujeme okamžitě časovač pro spuštění alarmu, ale dáme nejprve sledovacímu algoritmu příležitost jej opět zachytit.

K ovládání alarmu můžeme použít model automatu s pamětí navržený v předchozí podkapitole – stačí nad všemi změnami počtů osob nejprve spustit zde navržený algoritmus, až po jeho skončení pak případně vyvolat změnu v systému. Při přidání tohoto algoritmu by ještě bylo vhodné upravit odpovídajícím způsobem existující časovače T_1 a T_2, které může zejména delší interval u T_3 výrazně ovlivnit. Minimálně by měl být zkrácen nastavený interval časovače T_2 o prodlevu způsobenou T_3, jelikož oba časovače plní podobnou roli, budou působit aditivně.

[2]Diagram popisuje situaci při odchodu pokladní, nicméně algoritmus bude analogicky zpracovávat příchody i odchody v libovolné zóně, mění se pouze označení stavů a přechodů.

[3]Vzhledem k tomu, že jednotlivé stavy již nejsou binární, ale počítají si svůj vnitřní „stav“, i k tomu, že může docházet k odpálení několika přechodů souběžně (zatímco běží časovač v zóně pokladna, odejde i zákazník ze zóny fronta), podle formální definice již nejde o stavový automat, ale o Petriho síť. Ta umožňuje modelovat komplexní systémy řízené diskrétními událostmi (příchody a odchody osob) a počítat cíle v jednotlivých zónách, jde o silnější model než běžný stavový automat (zatímco automat modeluje pouze regulární jazyky, Petriho sítě pokrývají i jazyky bezkontextové). Jelikož jde ale u našeho algoritmu prakticky o superpozici dvou stavových automatů, pro potřeby tohoto popisu používám pojem automat i přesto, že není formálně zcela správný.

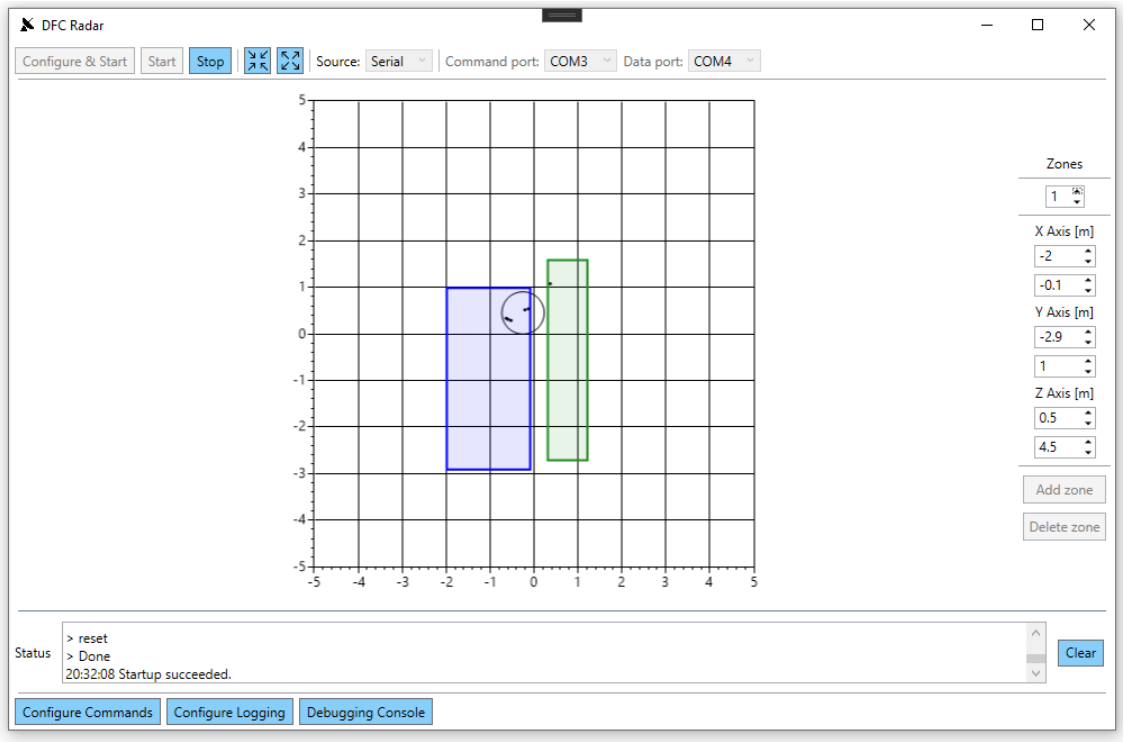
4 Vizualizace

Hlavním výstupem praktické části práce je vizualizační aplikace, která umožňuje komunikaci s radarem, jeho konfiguraci, zobrazení dat v reálném čase a ukládání statistik. Vizualizace je navržena pro platformu Windows, ke komunikaci využívá dva sériové porty (datový a příkazový). Komunikační protokol ze značné části vychází z demo-aplikace *Overhead 3D People Counting* od výrobce radaru, firmy Texas Instruments.

Prvním krokem při vývoji aplikace byla volba platformy. Zmíněná demo-aplikace od Texas Instruments používá k vizualizaci program Matlab, tomu jsem se ale chtěl vyhnout jednak kvůli problémům s licencováním případného komerčního využití aplikace (k dispozici mám pouze univerzitní licenci, která takové využití neumožňuje), jednak kvůli omezenému ekosystému a výrazně méně efektivnímu kódu. Další možností byl některý z nízkoúrovňových jazyků (např. C či C++), což by sice pro rychlé zpracování dat bylo výhodné, pro žádný z těchto jazyků se mi ale nepovedlo najít moderní, aktivně udržovanou knihovnu s permisivní licencí, která by splňovala potřeby mého programu (podporované typy grafů, rychlost překreslení v reálném čase, absence závislostí na samostatně instalovaných podprogramech, jako je například „Gnuplot“).

Proto jsem se rozhodl využít jazyk C# s grafickou platformou WPF (*Windows Presentation Foundation*), ve kterém jsem využil knihovnou pro kreslení grafů ScottPlot [8]. Jazyk C# využívá propracovanou JIT kompilaci[1], která pouze mírně zaostává za rychlostí výkonných kompilovaných jazyků, na rozdíl od nich ale poskytuje také velmi kvalitní prostředí pro design GUI v programu Visual Studio 2019. K definici grafických prvků slouží při použití WPF jazyk XML, přímo navázaný na *code behind* (tj. programovou logiku), což umožňuje snadné použití návrhových vzorů jako *Model-View-ViewModel*. Jednou z dalších výhod platformy WPF je zabudovaná podpora flexibilního uspořádání prvků, což uživateli umožňuje měnit velikosti oken, rozložení některých sekcí a podobně, přičemž uživatelské rozhraní se změnám automaticky přizpůsobí (to je zásadní výhoda oproti starším platformám jako *Windows Forms*, které umožňují vytvářet primárně aplikace se statickým rozložením). Jednotlivá okna navíc implicitně podporují rozdělení na vlákna, tj. je možné mít otevřeno několik pomocných oken a pracovat se všemi současně.

[1] *Just in Time* kompilace – program je překládán do strojového kódu za běhu programu.



Obr. 4.1: Hlavní okno vizualizace.

4.1 Uživatelské rozhraní

Jedním z cílů při vytváření uživatelského rozhraní bylo zajistit na jedné straně jednoduchost ovládání pro běžnou obsluhu, na druhé extenzivní konfigurovatelnost a odladitelnost při nasazování systému. Rozhodl jsem se proto nechat v hlavním okně pouze základní ovládací prvky a pokročilé subsystémy přesunout do samostatných vedlejších oken, s nimiž lze díky zmiňovanému miltithreadingu pracovat paralelně s hlavním oknem.

Pohled na hlavní okno programu zachycuje Obr. 4.1. Okno je rozděleno na několik základních celků – hlavní lišta (nahoře), nastavení a zobrazení zón (vpravo), informace o stavu aplikace (pole *Status* ve spodní části okna), lišta pro zpřístupnění pokročilých nastavení (dole) a vizualizace samotná (graf uprostřed).

V grafu můžeme pozorovat několik různých prvků. Drobné černé body odpovídají signálům, které zachycuje radar (jde tedy o výstup bloku Detekce, zmiňovaného na začátku kapitoly o algoritmech). Kružnice, vykreslená kolem souřadnic $(0,5; -0,5)$, představuje zachycený cíl (tj. výstup algoritmu *Group Tracker*) – její střed odpovídá středu hmoty, poloměr je úměrný kovarianci a disperzi jednotlivých bodů, ze kterých je cíl vytvořen. Posledním prvkem zobrazení jsou samotné zóny zájmu: modrá představuje oblast zákazníků, zelená pokladen. V případě, že je aktivován alarm, se barva pokladní zóny změní na červenou. Z obrázku tak můžeme vyčíst, že v dané

době byl ve sledované oblasti jediný zákazník, který právě přišel k pokladně (jelikož alarm zatím nebyl aktivován).

4.1.1 Ovládací prvky

Při zapnutí aplikace je prvním krokem volba datového zdroje. Automaticky je přednastaveno vyčítání dat ze sériové linky s předposledním existujícím portem nastaveným jako příkazový kanál, posledním jako datový (odpovídá to situaci, kdy je zařízení připojeno přes USB konvertor k PC bez nativních RS232 portů). Oba porty je ale samozřejmě možné libovolně změnit.

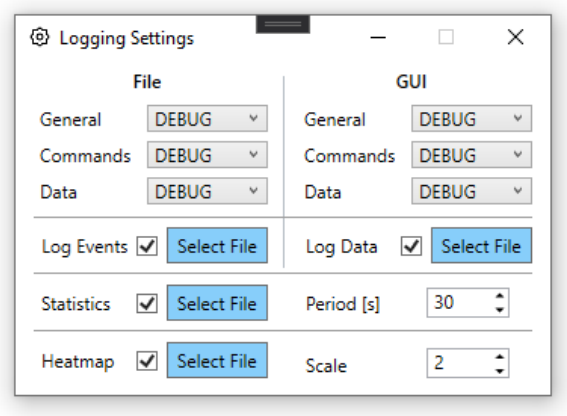
Druhým implementovaným zdrojem dat je zpětné přehrávání ze souboru. Aplikace dokáže zaznamenávat přenesená data, opatřit je časovými značkami a následně je opětovně přehrát, k tomu ale více v sekci Formát dat a jejich zpracování. Další datové zdroje momentálně podporované nejsou, aplikace je ale připravená na případné rozšíření o další technologie – vhodná by byla např. sběrnice CAN či bezdrátové připojení pomocí technologie Bluetooth Low Energy.

Dalším krokem je nastavení radaru a vymezení zón zájmu. Pokud je ve složce s aplikací přítomen konfigurační soubor, příkazy i zóny jsou automaticky načteny z něj – od obsluhy tak není vyžadována žádná akce. Pokud by ale přesto bylo nutné provést v nastavení změny (nebo by konfigurační soubor nebyl dostupný), je možné přímo v GUI měnit, přidávat i mazat zóny a případně i měnit příkazy pro nastavení radaru (o tom více v kapitole Konfigurace radaru).

4.2 Záznam a zpracování dat

Vizualizace má mimo samotného monitorování i druhou funkci – dokáže získaná data zapisovat na disk a případně dále zpracovat. Krom již zmíněného ukládání výsledků měření dokáže ještě generovat statistiky přítomnosti osob v zónách v závislosti na čase a tzv. teplotní mapu (*heatmap*), která zachycuje prostorové rozložení aktivity ve sledovaném prostoru (tj. představuje 2D informaci o toku osob monitorovanou oblastí). Samozřejmostí je pak také možnost zaznamenávání událostí (časy spuštění a ukončení aplikace, nestandardní jevy, chybové stavy apod.) do souboru.

Okno s nastavením parametrů zobrazuje Obr. 4.2, přístupné je z hlavního okna pomocí tlačítka *Configure Logging*. Horní polovina okna slouží k nastavení logování událostí. Je možné nastavit úroveň detailů zvlášť pro ukládání do souboru a pro grafické rozhraní (je tak možné ukládat detailní informace o běhu programu do souboru a získat tím informace zásadní například pro analýzu případných selhání, zároveň ale obsluze v aplikaci zobrazovat pouze to nejnutnější a zbytečně ji nezatěžovat tím,



Obr. 4.2: Nastavení ukládání a zpracování dat.

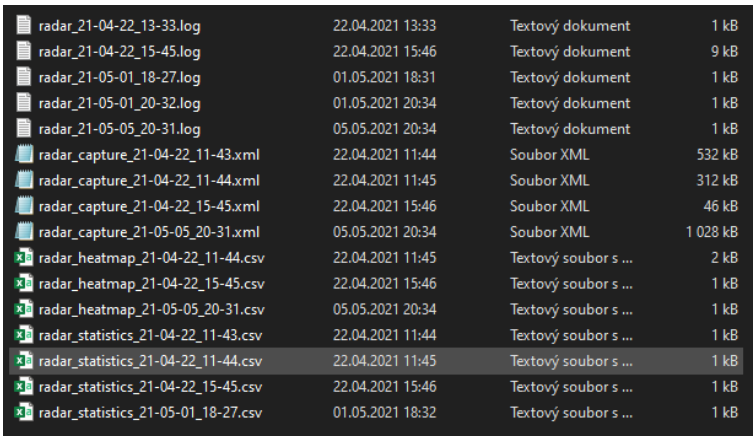
jaké příkazy jsme zrovna radaru poslali). Samotné události jsou rozděleny do tří kategorií:

- **General:** Obecné informace o běhu aplikace, chyby při načítaní či ukládání souborů, změny nastavení a jiné události, které nespadají pod zbylé dvě kategorie.
- **Commands:** Informace o odesílaných příkazech, odpovědi od radaru, manipulace s příkazovým portem.
- **Data:** Chyby při rozboru obdržených dat (ztráta synchronizace, chybný kontrolní součet), manipulace s datovým portem.

V každé kategorii pak můžeme vybrat jednu ze čtyř úrovní:

- **All:** Zaznamenávají se všechny dostupné informace: stav aplikace, veškerá komunikace, varování i chyby.
- **Debug:** Podobně jako předchozí stupeň, nezaznamenávají se ale informace o standardním běhu programu s malou výpovědní hodnotou (např. prodlevy při čekání na nová data, nevyžádané údaje z příkazové linky).
- **Warning:** Zaznamenávají se informace, které jsou pro běh aplikace pravděpodobně problematické – nepřijaté příkazy, nedostupné soubory, chyby při běhu.
- **Error:** Zaznamenávají se pouze tvrdé chyby znemožňující správný běh programu (fatální selhání komunikace, čtení nezpracovatelného souboru, . . .).

Zbývající řádky tohoto okna slouží k povolení (respektive zakázání) jednotlivých funkcí. Zaškrtnutím *Log Events* povolujeme ukládání údajů o událostech do souboru. *Log Data* aktivuje ukládání dat přijímaných z radaru (a tím umožňuje zpětné přehrávání). Nastavení *Statistics* ovládá generování statistik, tj. informací o počtu osob v jednotlivých zónách. Parametr *Period* určuje, jak často mají být statistiky počítány. Poslední prvek, *Heatmap*, umožňuje generování teplotní mapy a parametrem *Scale* říká, s jakým rozlišením mají být data generována (jeden „pixel“ mapy



Obr. 4.3: Soubory generované ve složce data.

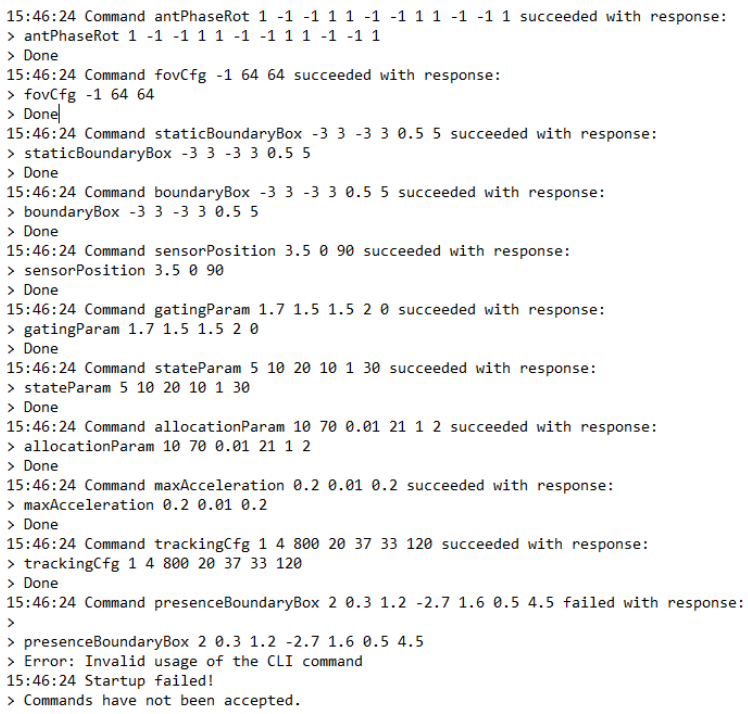
bude mít délku strany $\frac{1}{scale}$ metrů).

4.2.1 Formát dat a jejich zpracování

Všechna data jsou implicitně ukládána do složky **data**, vytvořené v adresáři s aplikací. Název každého souboru se skládá z typu záznamu a času pořízení, jak je patrné z Obr. 4.3. Připojení data a času ke jménu souboru zaprvé zajišťuje unikátnost názvu (pokud tedy radar není spouštěn několikrát během jedné sekundy, což ale rozhodně neodpovídá očekávanému využití), zadruhé usnadňuje orientaci v souborech. Toto chování je ale možné změnit a umístění i jména všech souborů nastavit ručně v okně *Logging Configuration*.

Pro záznam událostí je použit jednoduchý textový soubor (přípona .log), kde jsou události opatřeny časovou značkou a ukládány jedna na řádek, s případnými pokračovacími řádky uvedenými znakem '>', jak ukazuje Obr. 4.4. Tento jednoduchý formát byl zvolen pro snadnou čitelnost, jelikož pro tento soubor se nepředpokládá strojové zpracování (obsahuje informace, na základě kterých je možné vyřešit chyby či ladit program, při běžném, bezproblémovém provozu ale nemá využití). Při selhání aplikace tak stačí najít čas, kdy k němu došlo, vyhledat odpovídající časovou značku v souboru a přečíst si, co se v té době dělo.

Oproti tomu u záznamu dat se nepředpokládá ruční čtení, je určen primárně pro zpětné přehrávání záznamu. K ukládání tak byl zvolen formát Extensible Markup Language (přípona .xml), pro jehož serializaci i deserializaci má jazyk C# velmi dobrou podporu. Pro práci se souborem stačí nadefinovat datový typ, který popisuje ukládaná data, pomocí atributů přímo v definici typu nastavit jména jednotlivých položek a o zbytek se postará standardní knihovna jazyka. Vygenerovaný soubor nesplňuje striktní standard XML, jelikož neobsahuje předepsanou hlavičku a má více než jeden kořenový uzel, tato odchylka ale umožňuje výrazně snazší a efektivnější



Obr. 4.4: Záznam událostí.

serializaci toku paketů – není nutné upravovat již existující obsah souboru, stačí přidávat nové pakety na jeho konec. Proto si myslím, že drobná odchylka od standardu je odůvodněná, obzvlášť s přihlédnutím k tomu, že očekávaným konzumentem souboru je náš program, který s ní při deserializaci počítá.

Soubor se statistikami je ukládán ve formátu *Comma Separated Values (CSV)*, který umožňuje snadný import dat do programu Excel či jiných kancelářských programů. Díky tomu je jednoduché data jak zobrazit a číst, tak s nimi dále pracovat (například vytvářet grafy obsazenosti zón v závislosti na čase). Soubor používá při serializaci systémový jazyk počítače, na kterém program běží, jelikož například Excel v českém prostředí používá jiný formát souboru CSV než v anglickém (hlavním rozdílem je použití středníku jako oddělovače polí namísto čárek, jelikož čárky jsou použity pro zápis desetinných čísel). Využití systémové lokalizace tak zjednodušuje přenos vygenerovaného souboru k dalšímu zpracování.

Posledním typem výstupu je teplotní mapa. Ta opět používá formát CSV, kde první řádek a sloupec představují souřadnicové osy, ostatní buňky pak míru aktivity na daných souřadnicích. Hustotu souřadnic lze nastavit v okně *Logging Configuration* parametrem *scale*, viz sekce o konfiguraci výše. Míra aktivity je definována jako integrál počtu osob v čase po dobu běhu programu, tj. vysoké číslo odpovídá oblasti, kterou prochází mnoho zákazníků. Díky tomu lze například vyhodnotit, jak se zákazníci pohybují prodejnu, kterým uličkám věnují málo pozornosti, kde dochází

```
<ParsedPacket Time="2021-05-05T20:32:08.6169626+02:00" FrameNumber="283">
  <Points />
  <Targets>
    <Target ID="0" Radius="1.0408397">
      <Center X="-0.22359824180603027" Y="0.6760501861572266" />
    </Target>
  </Targets>
  <DebugInfo>
    <Entry>0:    0</Entry>
    <Entry>1:    7</Entry>
    <Entry>2: off.</Entry>
    <Entry>3:    0</Entry>
    <Entry>4:    0</Entry>
    <Entry>5:    2</Entry>
    <Entry>6: 30oC</Entry>
    <Entry>7: 24oC</Entry>
  </DebugInfo>
</ParsedPacket>
<ParsedPacket Time="2021-05-05T20:32:08.7196602+02:00" FrameNumber="284">
  <Points />
  <Targets>
    <Target ID="0" Radius="1.0408397">
      <Center X="-0.22359824180603027" Y="0.6760501861572266" />
    </Target>
  </Targets>
  <DebugInfo>
    <Entry>0:    0</Entry>
    <Entry>1:    7</Entry>
    <Entry>2: off.</Entry>
    <Entry>3:    0</Entry>
    <Entry>4:    0</Entry>
    <Entry>5:    2</Entry>
    <Entry>6: 30oC</Entry>
    <Entry>7: 24oC</Entry>
  </DebugInfo>
</ParsedPacket>
```

Obr. 4.5: Záznam dat.

Time	Zone_1	Zone_2
11:44:03	0	1
11:44:33	1	1
11:45:03	2	1
11:45:33	2	1
11:46:03	2	1
11:46:33	3	1
11:47:03	3	1
11:47:33	1	1
11:48:03	0	1
11:48:33	0	1
11:49:03	0	0
11:49:33	2	0

Obr. 4.6: Záznam statistik.

	-5	-4,5	-4	-3,5	-3	-2,5	-2	-1,5	-1	-0,5	0	0,5	1	1,5	2	2,5	3	3,5	4	4,5
-5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-4,5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-3,5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-2,5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-1,5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-0,5	0	0	0	0	0	0	0	0	0	0	0	3	3	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	4	6	21	7	4	13	0	0	0	0	0	0
0,5	0	0	0	0	0	0	0	0	3	10	9	0	17	22	8	27	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	5	10	22	53	0	0	0	0
1,5	0	0	0	0	0	0	0	0	0	0	0	0	1	3	0	2	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	30	0	0	0	0	0	0	0
2,5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3,5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4,5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Obr. 4.7: Teplotní mapa.

k tvoření dlouhých front a podobně.

4.3 Konfigurace radaru

Příkazy pro konfiguraci radaru z podstatné části vycházejí z komunikačního schématu původního firmware, poněvadž jejich změna by znamenala poměrně velké zásahy do existujícího kódu bez větších benefitů (některé příkazy by sice bylo možné zjednodušit či odstranit, jelikož jejich hodnoty se již nemění, výsledkem by ale bylo pouze zkrácení již vytvořeného konfiguračního souboru o několik řádků a zrychlení startu radaru o několik milisekund, nutných k odeslání nadbytečných příkazů). Nesoustředil jsem se proto na optimalizaci příkazů samotných, ale na jejich přehledné zobrazení a snadné úpravy.

Za tímto účelem jsem vytvořil okno *Configure Commands*, které dokáže zpracovat konfigurační soubor, načíst jej do přehledného grafického rozhraní a k jednotlivým příkazům přiřadit smysluplné názvy, vysvětlení funkce a významu jednotlivých parametrů. V neposlední řadě tento modul také provádí automatickou kontrolu validity zvolených hodnot při každé změně. Namísto kryptického příkazu

```
profileCfg 0 61.2 60.0 17.0 50 328965 0 55.27 1 64 2000.0 2 1 36
```

tak můžeme využít pohodlnější grafické rozhraní, které ukazuje obrázek 4.8. Toto nastavení vyvoláme tlačítkem *Configure Commands* z hlavního okna. Pro každý z příkazů zde vidíme název, seznam parametrů, jejich názvy a hodnoty. Při přidržení

Obr. 4.8: Okno pro nastavení příkazů.

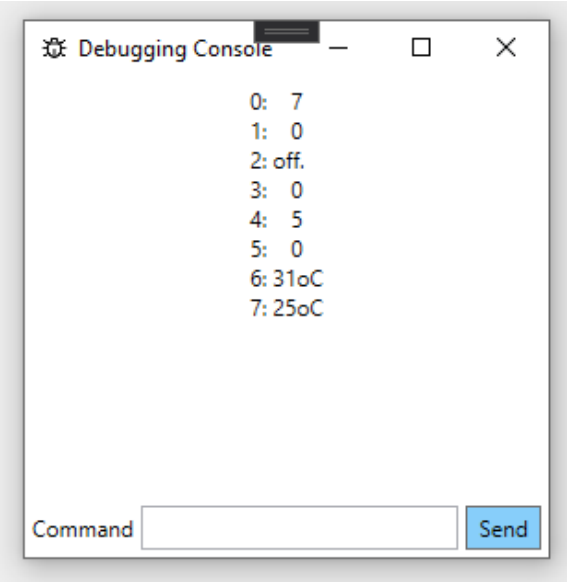
kurzoru nad názvem příkazu či parametru se zobrazí jejich detailní popis, při podržení nad hodnotou parametru se zobrazí povolený rozsah hodnot. Hodnoty jsou automaticky validovány – při zadání parametru mimo rozsah se pole zbarví červeně, viz například parametr `profileId` na obrázku.

4.4 Ladění

Posledním oknem, které zbývá popsat, je *Debugging Console*. To ale není určeno pro přímé použití při běžném běhu programu, bylo vytvořeno primárně pro účely ladění aplikace. Jelikož jsme aplikaci testovali vzdáleně, bez přímého přístupu k lokálnímu PC a fyzickému přípravku, bylo poměrně obtížné hledat chyby, ke kterým mohlo dojít na straně radaru, proto bylo nutné data nějak přenášet do vizualizace a zobrazovat. Za tímto účelem byl komunikační protokol rozšířen o pakety typu `debugging_message`, které umožňují přenášet libovolné informace, zakódované jako pole čtyřbajtových hodnot o proměnné délce.

Debugging Console pak souží k zobrazení těchto údajů na straně vizualizace. Jednotlivé prvky jsou rozloženy na čtveřice ASCII znaků (je tak možné přenášet i jednoduché textové informace), které obsahují údaje jako teplota senzoru, stavy čítačů, aktivace alarmu a podobně.[2] Všechny přijaté informace jsou navíc součástí

[2]Přesné složení, délka i pořadí jednotlivých polí vychází z úprav na straně radaru, samotná vizualizace jim žádný konkrétní význam nepřipisuje. Důvodem je, že tyto informace byly využívány



Obr. 4.9: Okno *Debugging Console*.

záznamu, pořizovaného z běhu aplikace, jsou tedy i zpětně dohledatelné.

Druhou funkcí, kterou tato konzole zpřístupňuje, je manuální ovládání radaru. V případě nutnosti (například při testování změn komunikačního protokolu, které ještě nebyly zakomponovány do vizualizace) umožňuje ruční zadání libovolného příkazu a jeho odeslání přes příkazovou sériovou linku, jestliže je připojena.

k rychlému prototypování, často se tak měnilo, které informace bylo vhodné zobrazit – nemělo ale smysl kvůli tomu pokaždé upravovat vizualizaci, kompilovat ji a nasazovat novou verzi na prodejnu.

Závěr

Hlavním cílem práce bylo vytvořit aplikaci použitelnou pro monitorování zón zájmu v rámci prodejny, vizualizaci pohybu osob a sběr statistických dat. Primární využití této aplikace je automatizace pokladen, tj. sledování fronty zákazníků a přivolání obsluhy, pokud u pokladny není prodavač. Ke sledování byla využita radarová technologie mmWave, která dosahuje velké přesnosti i při malých fyzických rozměrech senzoru.

K realizaci projektu byla nutná základní znalost radarové technologie: měření pomocí frekvenčně modulovaného vlnění, použité fyzikálních principy, podporovaný dosah a rozlišení a podobně. Na základě těchto informací bylo možné vybrat například umístění radaru – zvolil jsem instalaci na stropě, která optimalizuje sledovací úhly, zjednodušuje výpočty a minimalizuje chybu linearizace. Dalším krokem pak bylo nastudování algoritmů, využitelných k monitorování osob. Rozhodl jsem se pro technologii *Group Tracker* od firmy Texas Instruments, která v době realizace patřila k nejúčinnějším dostupným metodám, navíc je základní implementace dostupná přímo ve firmware radaru IWR6843, který jsem k realizaci využil.

Při testování v reálném provozu se ovšem sledovací algoritmus sám o sobě ukázal jako nedostatečný. Drobné výpadky v signálu z radaru vedly ke zbytečnému zapínání alarmu – ten se sice vzápětí vypnul, když byl signál opětovně zachycen, zbytečně ale vyrušoval obsluhu. Podobných problematických situací nastávalo hned několik, bylo tak nezbytné systém upravit. Pro řešení těchto problémů jsem se rozhodl využít stavový automat, který by mi umožnil ošetřit jednotlivé problémové stavy, do kterých se algoritmus dostával. Podrobněji návrh automatu i řešené problémy popisuje kapitola Návrh stavového automatu.

Takto posílený systém již fungoval v souladu se zamýšleným použitím, jednotlivé parametry stavového automatu navíc dávají možnost, jak algoritmus přizpůsobit konkrétním problémům, které se mohou vyskytnout při jednotlivých instalacích (například upravením délky jednotlivých časovačů). Pří následném testování a optimalizaci jsme upravili parametry senzoru, sledovacího algoritmu i stavového automatu tak, že aplikace na prodejně plnila svůj primární účel bez větších zádrhelů.

Posledním krokem tak bylo navržení vizualizační aplikace – těžko od obsluhy vyžadovat, aby radar ovládala ručním vpisováním příkazů do příkazové řádky. Začal jsem jednoduchým rozhraním, které umožnilo načíst příkazy pro radar ze souboru, odeslat je přes sériovou linku a následně zobrazovat měřená data. To se ovšem ukázalo být poměrně nepohodlným při optimalizaci jednotlivých nastavení – jelikož konfigurační soubor obsahuje desítky příkazů, z nichž mnohé mají osm i více parametrů, provádění změn a jejich testování nebylo triviální (změna prahu pro alokaci znamenala najít si v dokumentaci senzoru příkaz `allocationParam`, zjistit, koli-

kátý parametr odpovídá prahu, najít si odpovídající řádek v souboru a odpočítat, který parametr chceme změnit). Hlavní okno aplikace jsem proto rozšířil o modul *Configure Commands*, který umožňuje zobrazit, upravovat i verifikovat jednotlivé příkazy a jejich parametry (i s krátkým popisem funkcionality) přímo v aplikaci, bez nutnosti použít dokumentaci.

Další funkcí, kterou aplikace vyžadovala, byly nástroje pro záznam událostí a ladění. Jelikož testování probíhalo na prodejně přes připojení na vzdálenou plochu, neměli jsme možnost ji spouštět v debuggeru, ani v reálném čase monitorovat všechny problémy. Komunikační schéma proto bylo rozšířeno o „debugging“ pakety, které umožňují z firmware radaru odeslat libovolná data, která jsou následně zobrazena v aplikaci v okně *Debugging Console*. Zároveň bylo v této fázi vývoje doplněno i okno s nastavením pro zobrazování událostí v grafickém rozhraní a jejich ukládání do souboru, což nám umožnilo zpětně projít činnost aplikace a zjistit, kde mohlo dojít k chybě.

Nyní již byla aplikace ve stavu, kdy mohla být reálně použita, mohli jsme se tak věnovat sekundárním funkcím, o které potenciální uživatelé projevili zájem – analýze nasbíraných dat. Krom ukládání celého měření pro případné pozdější přehrání jsme se rozhodli analyzovat pohyb zákazníků ve dvou oblastech – časové a prostorové. Časová analýza je poměrně jednoduchá: aplikace do souboru CSV periodicky zaznamenává, kolik zákazníků v daném časovém úseku prošlo každou ze zájmových zón.[3] Tyto údaje lze využít například ke sledování velikosti fronty u pokladny v závislosti na denní době, sledování počtu zákazníků vcházejících do obchodu a vycházejících z něj a podobně. Použití formátu CSV navíc umožňuje snadný převod dat do kancelářských programů a další zpracování, například vytváření grafů. Data o prostorovém toku zákazníků jsou zaznamenána ve formě teplotní mapy – dvourozměrného grafu, který pro jednotlivé části prodejny[4] počítá míru aktivity.

Takto vytvořená aplikace byla dlouhodobě testována v reálném provozu, ve kterém dokázala, že může plnit svou funkci. Základní funkcionalita funguje bez problémů, jisté úpravy či rozšíření, které by i přesto bylo možno provést, zmiňuji v následující podkapitole.

4.5 Možnosti rozšíření

V budoucnosti je možno vytvořenou aplikaci dále zdokonalovat, vhodnou oblastí by byly zejména komunikační možnosti. Jak bylo zmíněno v kapitole o vizualizaci, komunikační protokol z podstatné části vychází z původního firmware pro IWR6843

[3] Implicitně je použitý interval 30 sekund, je ale nastavitelný v okně *Logging Configuration*.

[4] Oblasti jsou definovány mřížkou kartézských souřadnic, údaje jsou počítány pro každé pole této souřadnicové sítě. Hustotu mřížky lze nastavit v okně *Logging Configuration*.

(mmWave SDK). Komunikace sice nemá znatelné negativní dopady na rychlost zpracování ani kvalitu sledování, nicméně v paketech jsou posílány i některé údaje, které není nutné přenášet, a příkazy k nastavení radaru jsou zbytečně komplikované. Pročištění komunikačních protokolů by tak minimálně zlepšilo čitelnost programu a usnadnilo konfiguraci a ladění.

Citelnějším problémem použitého komunikačního modelu je fyzické propojení pomocí sériové linky. Vzhledem k nutnosti umístění radaru na stropě místnosti, ideálně ve středu sledované oblasti, není instalace komunikačního kabelu a jeho přivedení k obslužnému počítači příliš pohodlná. Bylo by tedy vhodné implementovat i bezdrátový komunikační protokol (jako vhodný kandidát se jeví Bluetooth Low Energy), který by k instalaci fyzické spojení nevyžadoval. Tato změna by ale vyžadovala nejen netriviální změny ve firmware radaru, ale hlavně hardwarové úpravy, jelikož IWR6843 nemá integrovanou podporu pro Bluetooth.

Další zajímavou cestou by byla experimentální implementace některého ze složitějších sledovacích algoritmů. Sledování osob pomocí mmWave senzorů je velmi aktivní výzkumnou oblastí (jak ostatně ukazuje i to, že oba algoritmy zmiňované jako možné alternativy na závěr kapitoly Zpracování dat a detekce osob byly publikovány v roce 2021, tedy vznikaly souběžně s touto prací), zatímco tak ještě nedávno byl *Group Tracker* jednoznačně na špičce dostupných implementací, je pravděpodobné, že řada nových algoritmů jej v brzké době překoná. Pro potřeby naší aplikace to sice není zásadní, jelikož dosažená přesnost je již nyní adekvátní, analýza možných nových uplatnění s pokročilejším sledováním by ale mohla přinést zajímavé výsledky.

Literatura

[1] European Parliament and Council of European Union. Regulation (EU) no 679/2016, Duben 2016. URL: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A02016R0679-20160504&qid=1609687095701.

[2] Tracking radar targets with multiple reflection points. Technical report, Texas Instruments, Únor 2021. URL: https://dev.ti.com/tirex/explore/content/mmwave_industrial_toolbox_4_7_0/labs/people_counting/docs/Tracking_radar_targets_with_multiple_reflection_points.pdf.

[3] C. Iovescu and S. Rao. The fundamentals of millimeter wave sensors. Technical report, Texas Instruments, Červen 2020. URL: https://www.ti.com/lit/wp/spyy005a/spyy005a.pdf?ts=1620006030065&ref_url=https%253A%252F%252Fwww.ti.com%252Fsensors%252Fmmwave-radar%252Fwhat-is-mmwave.html.

[4] Luděk Dudáček. Radarové měření vzdálenosti. Bakalářská práce, Západočeská univerzita v Plzni. Katedra aplikované elektroniky a komunikací, 2012.

[5] Texas Instruments. *IWR6843, IWR6443 Single-Chip 60- to 64-GHz mmWave Sensor*, 2020. URL: https://www.ti.com/product/IWR6843.

[6] Xu Huang et al. Indoor detection and tracking of people using mmwave sensor. *Journal of Sensors*, 2021. URL: https://doi.org/10.1155/2021/6657709.

[7] Jacopo Pegoraro et al. Multiperson continuous tracking and identification from mm-wave micro-doppler signatures. *IEEE Transactions on Geoscience and Remote Sensing*, 59(4):2994–3009, 2021. URL: https://doi.org/10.1109/TGRS.2020.3019915.

[8] Scott Harden. *Knihovna ScottPlot*, 2021. URL: https://swharden.com/scottplot/.

5 Přílohy

5.1 Obsah přiloženého CD

Přiložené CD obsahuje dvě složky – zdrojový kód vizualizace (složka `source`) a ukázku souborů, generovaných za jejího běhu (složka `data`). V kořenovém adresáři je umístěn také soubor `README.md`, který obsahuje podrobnější popis jednotlivých souborů a složek včetně stručného návodu ke kompilaci.

Generovaná data zahrnují soubor událostí, záznam z měření, statistické údaje a teplotní mapu. Slouží primárně k ilustraci výstupů, které aplikace poskytuje. Detaily k jednotlivým souborům, jejich funkcím a formátu viz kapitola Formát dat a jejich zpracování. Zároveň jsou zde umístěny také konfigurační soubory pro případně testování funkcionality aplikace.