České vysoké učení technické v Praze
Fakulta elektrotechnická

DIPLOMOVÁ PRÁCE

Využití optického rozpoznávání pro řízení robota v rámci automatizace testů v automobilovém průmyslu

Praha, 2009 **Autor: Bc. Tomáš Novák**

Prohlášení

Prohlašuji, že jsem svou diplomovou práci vypracoval samostatně a použil jsem pouze podklady uvedené v přiloženém seznamu.

V Praze dne 22.12.2009

Tomáš Novák

Abstrakt

Testování jako jeden ze základních článků vývojového procesu často vyžaduje ke svému urychlení a zpřesnění využít automatizaci. V průběhu testování elektronických zařízení jsou obvykle měřeny různé veličiny jako napětí, příkon nebo teplota. Často je však vyžadována i optická kontrola pomocí kamery.

Následující práce seznamuje s automatizovaným testerem autorádií firmy *e4t electronics for transportation s.r.o.*, navrhuje jeho rozšíření o systém optického rozpoznávání s použitím průmyslové kamery uEye a nabízí možnosti realizace navržených úloh. Vybrané úlohy jsou implementovány s využitím knihovny funkcí pro zpracování obrazu OpenCV. Kamera slouží jako podpůrný nástroj, který poskytuje urychlení a zpřesnění procesu automatizovaného testování.

Abstract

Testing as one of the main parts of developmental process requires often to be automatized because of the speed and accuracy requirements. During the electronic system test several variables as voltage or temperature are measured. Often we also require optical verifying of the system by a camera.

This thesis introduces the automated radio-testing system from *e4t electronics for transportation l.t.d.* The optical-system improvement of the testing system is proposed using the uEye camera. Selected tasks are implemented using the OpenCV image processing library. The camera serves as a technical support which allows speeding up and improving accuracy of the automated tests.

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra řídicí techniky

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: **Bc. Tomáš Novák**

Studijní program: Elektrotechnika a informatika (magisterský), strukturovaný
Obor: Kybernetika a měření, blok KM1 - Řídicí technika

Název tématu: **Využití optického rozpoznávání pro řízení robota v rámci automatizace testů v automobilovém průmyslu**

Pokyny pro vypracování:

Seznamte se s průmyslovou kamerou uEye od firmy IDS Imaging Development Systems GmbH.
Seznamte se se softwarem pro zpracování a rozpoznávání obrazu TOM Line od firmy GÖPEL electronic GmbH.
Seznamte se s robotem a softwarem pro ovládání robota od firmy KUKA Roboter GmbH.
Ověřte použitelnost kamery uEye v kombinaci se software TOM Line na automatické rozpoznávání ovládacích elementů autorádií.
Navrhněte interface pro přenos informací o rozpoznaných elementech ze software TOM Line do automatizovaného testovacího systému TestAut2 od zadavatele DP.
Proveďte funkční testy a vše řádně zdokumentujte.

Seznam odborné literatury:

TOM Line User manual, GÖPEL electronic GmbH
KUKA Software Documentation, KUKA Roboter GmbH
User Manual uEye Cameras, IDS Imaging Development Systems GmbH
Dokumentace testovacího systému TestAut2, e4t electronics for transportation s.r.o.

Vedoucí: Ing. Jan Krákora, Ph.D.

Platnost zadání: do konce zimního semestru 2010/2011

prof. Ing. Michael Šebek, DrSc.
vedoucí katedry

L.S.

doc. Ing. Boris Šimák, CSc.
děkan

V Praze dne 8. 6. 2009

Obsah

Abstrakt **v**

Obsah **ix**

Seznam obrázků **xi**

Seznam tabulek **xiii**

1 Úvod **1**

1.1 Související práce 2

1.2 Struktura automatizovaného testeru 2

1.3 Cíl práce 4

1.4 Struktura práce 4

2 Automatizované testování **5**

2.1 Program TestAut 6

2.1.1 Programovací jazyk TCL 7

2.2 Robot KUKA kr3 9

2.2.1 Komunikace 10

2.2.2 Programovací jazyk robota 11

2.2.2.1 Druhy pohybu 11

2.2.2.2 Funkce a řízení programu 12

2.2.2.3 Čítače a přerušení 14

2.2.2.4 Advance mode 15

2.2.3 Úpravy řídicího algoritmu 15

2.2.3.1 Řídicí program 18

2.3 Kamera uEYE SE 19

2.4 Nedostatky stávajícího systému 20

3 Optické rozpoznávání pro testy **23**
3.1 Optická zpětná vazba . . . 23
3.1.1 Odkódování rádia . . . 23
3.1.2 Nastavení hlasitosti . . . 24
3.1.3 Kontrola volitelných funkcí . . . 24
3.1.4 Verifikace RDS . . . 25
3.2 Ovládání dotykového displeje . . . 25
3.3 Rozpoznávání barev . . . 26
3.4 TOM Line . . . 27
3.5 Knihovna OpenCV . . . 30
3.6 Zpracování obrazu . . . 31
3.7 Ekvalizace obrazu . . . 31
3.8 Prahování . . . 34
3.9 Čtení stavu Progressbaru . . . 36
3.10 Rozpoznávání znaků a symbolů . . . 37
3.10.1 Tesseract . . . 38
3.10.2 GOCR . . . 38
3.11 Interface pro TestAut . . . 39

4 Experimenty **41**
4.1 Pohyby robota . . . 41
4.2 OCR . . . 42
4.3 Analýza progressbaru . . . 43

5 Závěr **45**

Literatura **46**

Seznam symbolů a zkratek **49**

A Knihovna OpenCV **I**
A.1 Struktura IPLImage . . . I
A.2 Získání barevného obrazu z kamery pro OpenCV . . . II

B Testy **III**
B.1 Funkce pro odkódování rádia . . . III
B.2 Hiss-noise – příklad z normy . . . V

Seznam obrázků

1.1 Architektura testeru TestAut 3

2.1 Příklad výstupního grafu – test Hiss noise 6
2.2 Hlavní okno aplikace TestAut 7
2.3 Manipulátor KUKA kr-3 a detail nástrojů 10
2.4 Způsob získání souřadného systému ze šesti parametrů 12
2.5 Příklad dvou různých pozic robota se stejnou polohou nástroje 13
2.6 Trajektorie P0 → P3: (a) bez aproximace, (b) s aproximací 16
2.7 Zóny pohybu robota s příkladem trajektorie 16
2.8 Pohyby robota – vývojový diagram 17

3.1 Polovina spektra demodulovaného stereo FM signálu s RDS 25
3.2 Grafická reprezentace HSV kruhu 26
3.3 Okno *Teach* při učení znaků 27
3.4 Nastavení parametrů funkce *brightness-check* 29
3.5 Analýza ukazatele hlasitosti funkcí *pixel-count* 29
3.6 Příklad jasové transformace – *zvýraznění kontrastu* 32
3.7 Příklad jasově normalizovaného snímku – tmavý originál 33
3.8 Příklad jasově normalizovaného snímku – světlý originál 34
3.9 Velikost okna u adaptivního prahování: a) malé, b) vyhovující, c) zbytečně velké 35
3.10 Prahování: část snímku z kamery, jeho normalizace a prahovaný obraz . . 36
3.11 Progressbar – nalezená spojitá komponenta a otvor uvnitř 36
3.12 Komponenty symetrického ukazatele 37
3.13 Učení a klasifikace do tříd 38
3.14 Schéma implementace navrženého rozšíření 39

Seznam tabulek

2.1 Příkazy pro nastavení parametrů pohybu 12
2.2 Typy proměnných v jazyce KRC 13
2.3 Vybrané parametry použité kamery 19
2.4 Vybrané funkce pro práci s kamerou 20
2.5 Výstupní formáty obrazu kamery kompatibilní s OpenCV 20

4.1 Zrychlení programu: nastavení bezpečnostního kódu 9775 41
4.2 Čas vykonání funkce `Center` 42
4.3 Texty rozpoznané pomocí různých OCR 42

Kapitola 1

Úvod

Během vývoje nového výrobku hraje nezastupitelnou roli testování, jenž má za úkol odhalit chyby v návrhu, respektive v jeho prototypové realizaci. V případě předkládané práce je takovým prototypovým výrobkem nový model autorádia. Samotné testování může probíhat manuálně, což znamená, že u rádia procházejícího ověřením funkčnosti sedí tester, přepíná různé funkce a sleduje všechny měřené veličiny. Jinou možností, která nás bude zajímat v následujícím textu, je testování automatizovat.

V případě automatizovaného testování odpadají rušivé vlivy jako únava či nepozornost testera, narůstá rychlost testování, odpadají zbytečné prostoje a test může probíhat několik dní bez přestávky. Výsledkem takového testu je přehledně zpracovaná výstupní zpráva, ve které jsou zvýrazněny chybné výsledky, například naměřené hodnoty konkrétní veličiny mimo její povolený rozsah daný příslušnou normou. Vlastní obsluha testů se potom redukuje na sestavení testovací sekvence, její spouštění a následnou analýzu generovaných výsledků.

Použitý systém automatizovaného testování využívá k mechanickému ovládání rádia průmyslového robota. Robot umožňuje stiskávat ovládací tlačítka a otáčet inkrementálními senzory. Většinu takovýchto úkonů lze provádět i bez zpětné vazby, například knoflíkem hlasitosti otočíme doleva tolikrát, aby byla s jistotou nastavena hlasitost na nulu, a potom odkrokujeme zpět požadovanou hodnotu. Prodlužuje se však zbytečně doba potřebná k ukončení požadované operace. Navíc může vlivem nepřesností zcela náhodně docházet například k odskakování inkrementálních senzorů v koncové poloze, což ve výsledku způsobí chybně naměřené údaje.

Prezentovaná práce řeší připojení kamery do stávajícího systému automatizovaného testeru a zabývá se možnostmi jejího využití pro zrychlení, zpřesnění a vylepšení testů

spolu s jejich následnou implementací. Pro analýzu obrazu měl být původně, z důvodů kompatibility s koncernem VW, použit software TOMLine. V průběhu práce se ukázalo, že jeho spolehlivost ani možnosti nejsou pro danou aplikaci dostatečné, a tak bylo navrženo a vytvořeno vlastní řešení.

Jako programátorské prostředky jsou použity jazyky **C++** jednak pro operace zpracování obrazu a jednak pro vytvoření rozšiřujících balíčků do ovládacího softwaru TestAut, **TCL** [16] pro psaní algoritmu vlastních testů a **KRC** [10] pro ovládání robota. Práce zahrnuje i základní seznámení s programováním pomocí COM [9] a na něm založeném vzdáleném přístupu pomocí OPC [7].

1.1 Související práce

Lze konstatovat, že o podobných plně funkčních projektech univerzálního automatizovaného testeru nejsou veřejně dostupné žádné informace. Jako konkurenční projekt můžeme chápat testovací systém EXAM německé firmy Micronova. Systém EXAM je universální testovací platforma využívaná například automobilkou VW. Systém je stále ve stádiu vývoje a testy se neobejdou bez asistence obsluhy. Optické rozpoznávání je řešeno vlastním balíkem VW IP.

V oblasti rozpoznávání znaků existuje mnoho různých přístupů [17], od jednoduché a nespolehlivé korelace, srovnávání s deformovatelnými vzory, využitím projekčních diagramů, přes grafovou reprezentaci kostry znaku po různé geometrické invariantní momenty nebo Fourierovu transformaci. Často jsou použity metody statistického rozpoznávání spolu se systémy neuronových sítí. Různé přístupy se navzájem liší výpočetní složitostí, dosažitelnými výsledky a spolehlivostí. Tato práce nemá ambice na implementaci vlastního řešení optického rozpoznávání znaků a bude těžit z již existujících funkčních nástrojů.

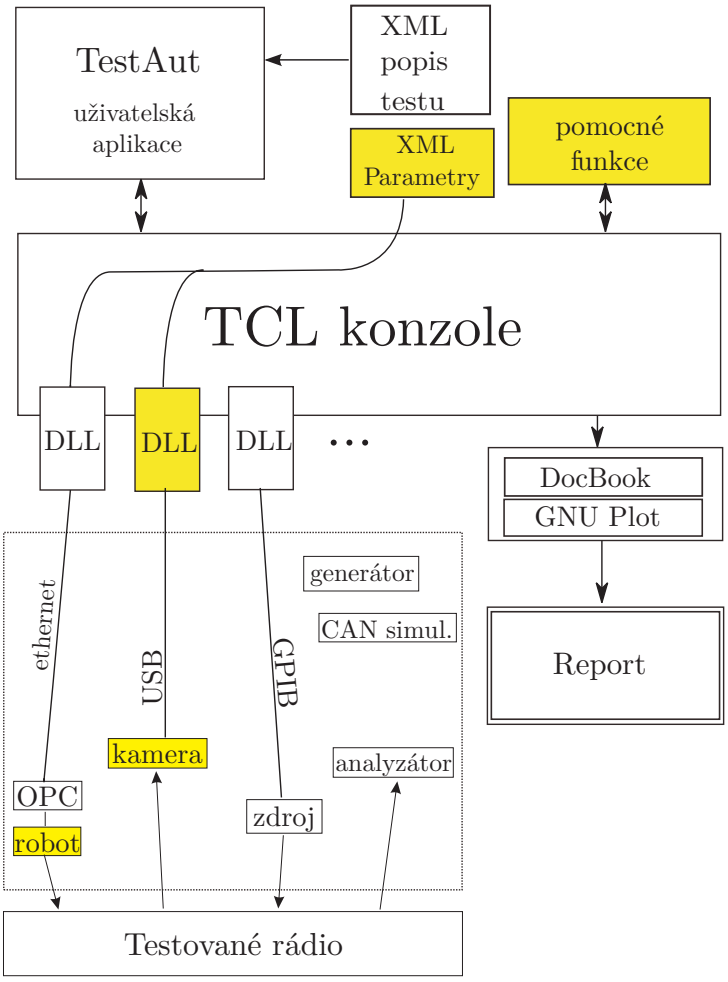
1.2 Struktura automatizovaného testeru

Celý systém automatizovaného testování TestAut je navržen velmi obecně a naprosto nezávisle na prováděných testech. Vlastní testy a jejich parametry jsou popsány v XML souborech, algoritmy testů jsou vytvářeny pomocí interpretovaného skriptovacího jazyka TCL [16]. Všechna zařízení potřebná pro správný průběh testů, jako jsou napájecí zdroje, měřicí přístroje, přepínače nebo manipulátory, jsou z pohledu SW do systému připojeny

jako jednotlivé balíčky (packages). Vlastní komunikace s těmito přístroji je pak řešena pomocí DLL knihoven.

Datové soubory XML jsou použity i pro uložení dalších parametrů, jakými může být například databáze pohybů robota pro jednotlivé modely testovaných zařízení.

Architekturu především softwarové části testeru reprezentuje obrázek 1.1, ve kterém jsou rovněž vyznačeny části, přímo se dotýkající prezentované práce. Z pohledu testů je nejdůležitější částí TCL konzole, která je zodpovědná za vykonávání všech testů. Zásuvné moduly DLL zprostředkovávají komunikaci po různých fyzických rozhraních, jako např. USB, ethernetu nebo sběrnici GPIB [19]. Uživatelská aplikace TestAut spouští vybrané položky z hlavního seznamu testů v konfiguračním XML souboru. Spuštěné testy pak dále volají pomocné funkce, závislé na daném testovaném zařízení. Z naměřených údajů je automaticky generována výsledná zpráva s grafy.



Obrázek 1.1: Architektura testeru TestAut

1.3 Cíl práce

Prezentovaná práce obohacuje původní systém o možnost použití kamery při automatizovaném testování. Během seznámení s projektem testeru budou analyzovány možné oblasti využití kamery. Úkolem kamery má být především podpora testů, cílem není vytvoření autonomního obrazem řízeného systému.

Na základě návrhu na rozšíření testeru bude potom vytvořen rozšiřující balíček, který umožní získat snímky z kamery, tyto následně zpracovat a v souladu s potřebami testů analyzovat. Nedílnou součástí bude nastudování způsobu řízení použitého průmyslového robota a změna jeho programu s ohledem na manipulaci s kamerou. Nakonec bude potřeba zakomponovat optické rozpoznávání do stávajících testů a posoudit přínos celého řešení.

1.4 Struktura práce

Protože je práce zaměřena na rozšíření již funkčního projektu, bude její první součástí seznámení se stávajícím systémem automatizovaného testeru rádií. Popisu je věnována následující, druhá kapitola. Ve třetí kapitole nejprve uvedeme konkrétní motivaci pro zavedení optického vstupu, následovat bude popis potřebného hardwaru a nakonec rozbor řešení. Kapitola 4 je věnována experimentům, popisuje tedy především způsob ověření funkčnosti vytvořeného řešení a dále prezentuje přínos práce na základě časové úspory v průběhu testů.

Kapitola 2

Automatizované testování

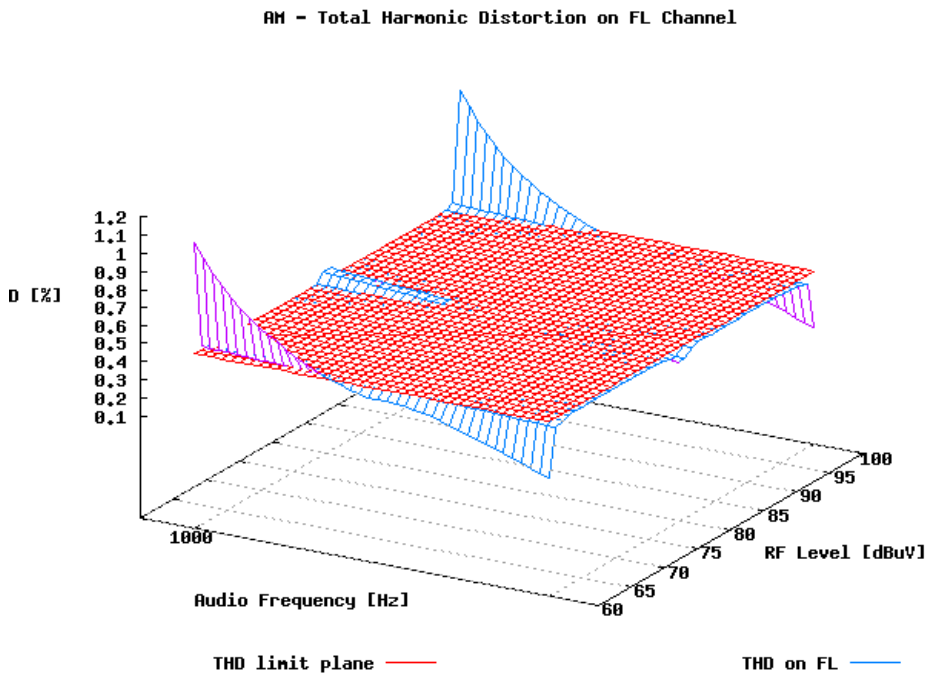
Součástí původního systému pro komplexní automatizované testování prototypových autorádií byly různé pro testování nezbytné přístroje s možností vzdáleného ovládání, jako stejnosměrný napájecí zdroj, generátor NF signálu, univerzální analyzátor, modulátor rádiového signálu FM/AM, simulátor sběrnice CAN nebo reléový přepínač vstupů a výstupů rádia. K ovládání mechanických prvků rádia slouží průmyslový robot se vhodně upravenými ovládacími nástroji. Součástí systému je také držák pro uchycení testovaného zařízení. Vlastní průběh testů je řízen pomocí speciálního programu v PC. Výsledkem každého testu je přehledná zpráva se zvýrazněním nalezených chyb a nepřípustných odchylek.

Testovací systém se specializuje na měření audio-parametrů, ale i tunerové, elektrické či dlouhodobé testy.

Jako příklad ze skupiny audio-testů uveďme test *Volume*, který má za úkol proměřit závislost výstupního napětí na zatěžovacím odporu 4 Ω na nastavené úrovni hlasitosti. Nastavování hlasitosti je prováděno postupně otočným ovladačem za pomoci robota a jeho speciálně uzpůsobeného úchopového mechanismu.

Tunerovým testem je pak například *Hiss noise* podle odstavce 4.4.6 koncernové normy VW 80972. Odpovídající část normy je obsažena v příloze. Úkolem je ověřit, zda úroveň zkreslení THD nepřekračuje danou mez pro žádnou hodnotu frekvenčního zdvihu, vstupního výkonu a frekvenci modulačního signálu v předepsaných intervalech. Obrázek 2.1 ukazuje jeden z výstupních grafů takového měření. Z množství naměřených bodů se můžeme snadno přesvědčit o skutečné potřebě automatizovaného testování.

Dlouhodobým testem může být několikadenní intenzivní mačkání tlačítek spolu s pravidelným ověřováním jejich bezchybné funkčnosti, nebo dlouhodobé měření stálosti úrovně hlasitosti výstupního zvukového signálu.



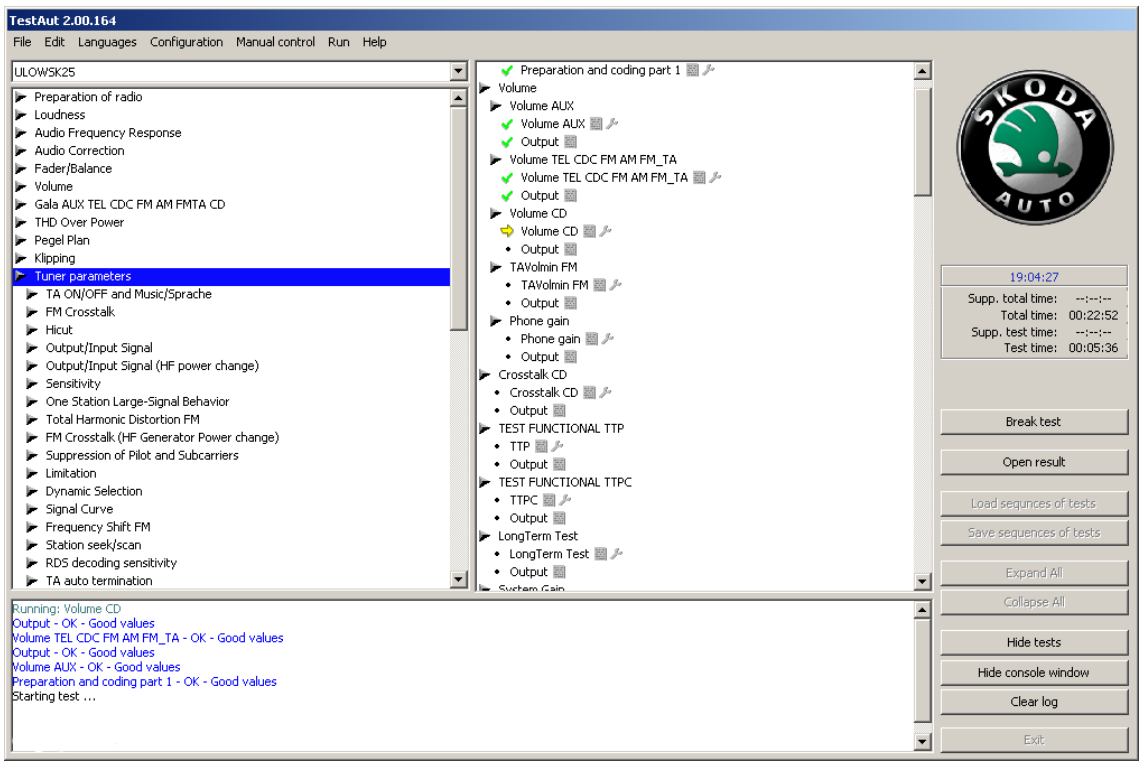
Obrázek 2.1: Příklad výstupního grafu – test Hiss noise

V této kapitole seznámíme čtenáře se stávajícím systémem, respektive jeho částmi, které mají přímou vazbu na rozšíření systému o optické rozpoznávání.

2.1 Program TestAut

TestAut je aplikace sloužící k sestavování a spouštění automatizovaných testů. Aplikace zprostředkovává uživatelské rozhraní, jehož hlavní okno je znázorněno na obr. 2.2. Po výběru typu rádia je z konfiguračního souboru XML načten a v levé části okna zobrazen kompletní seznam dostupných testů. Jednotlivé testy a jejich kombinace lze přetažením přesunout do pravé části okna a vytvořit tak *testovací sekvenci*. Dále může uživatel změnit jednotlivým testům jejich parametry, kterými může být například počet opakování testu nebo povolený rozsah měřených veličin. Dialog pro změnu parametrů lze vyvolat poklepáním na název testu.

Aby byla zajištěna možnost jednoduché změny a rychlého vývoje testů, je potřeba, aby ovládací aplikace byla binárně nezávislá na vlastních testech. Nezávislosti aplikace na testech je dosaženo využitím skriptovacího jazyka TCL pro vytváření testů a dále ukládáním všech parametrů ve značkovacím jazyce XML. Pro připojení fyzického zařízení



Obrázek 2.2: Hlavní okno aplikace TestAut

(generátoru, analyzátoru, napájecího zdroje, ale i kamery) je potřeba vytvořit balíček v podobě speciální DLL knihovny, který rozšíří tabulku funkcí jazyka TCL.

2.1.1 Programovací jazyk TCL

Interpretovaný skriptovací jazyk TCL[1] je specializovaný na snadnou a rychlou tvorbu prototypových aplikací a je připraven pro zabudování do dalších aplikací. Patří stále mezi průmyslově užívané standardy. Jazyk ideově vychází z funkcionálního jazyka LISP [12], má však zjednodušenou syntaxi, což zvyšuje čitelnost zdrojového kódu. Téměř vše je chápáno jako textové řetězce, neexistuje tedy například více různých typů proměnných. Samotný jazyk nedefinuje příkazy pro tvorbu matematických výrazů, smyček ani podmínek, vše je řešeno pomocí funkcí.

Cenou za jednoduchost a univerzalitu je pomalejší vykonávání programů ve srovnání s kompilovanými jazyky a v neposlední řadě složitější odlaďování skriptů, protože chyby v syntaxi nejsou odhaleny během kompilace. Jazyk také neumožňuje objektově orientované programování, což je v našem případě vyváženo existencí jmenných prostorů,

[1] Tool Command Language

takzvaných `namespaces`.

Jmenných prostorů je v testeru využito pro vytváření hierarchie a abstrakce jak jednotlivých druhů použitých přístrojů, tak prováděných testů. Jako příklad uveďme balíček pro komunikaci s robotem. Jmenná třída `ROBOT` definuje abstraktní funkci `Move`, ta však není přímou implementací komunikace s robotem. Při inicializaci systému je v souladu s konfiguračním souborem načten balík příkazů konkrétního robota, v našem případě namespace `KUKA`, který obsahuje vlastní implementaci funkcí. Tyto funkce jsou posléze volány z abstraktních funkcí namespace `ROBOT`. Ve zdrojových kódech testů je však vždy volána funkce `ROBOT::Move`, což zajišťuje nezávislost na skutečném hardware. Je možno použít více zařízení stejné třídy (např. dva napájecí zdroje), zařízení jsou pak rozlišena svým jednoznačným identifikátorem.

Obdobným způsobem je abstrakce využita v případě rádií. Na nejvyšším stupínku hierarchie je definována třída `RADIO` a v ní například funkce `RADIO::Center`. Úkolem této funkce je nastavit všechny akustické parametry (bass, treble, fader, balance) na výchozí hodnotu. Funkce ve třídě `RADIO` neobsahuje vlastní kód, ale volá stejnojmennou funkci z namespace konkrétního rádia (např. `ULOWSK25::Center`). Funkce `Center` je volána na začátku každého testu, který provádí měření výstupních nízkofrekvenčních akustických signálů. Vyměníme-li testované rádio za nový typ, je potřeba nově implementovat funkci `Center` v namespace nového rádia, ale nemusí být zasahováno do kódu žádného s akustických testů.

Dále zde uvedeme nezbytné základy jazyka Tcl, sloužící k jednoduchému porozumění napsaným testům a umožňující další testy vytvářet a modifikovat.

Proměnné jsou vytvářeny a povinně inicializovány příkazem `set nazev hodnota`, z paměti je lze volitelně odebrat pomocí `unset nazev`. K hodnotě proměnné lze přistupovat pomocí prefixu `$`, tedy například `$frequency`.

Jak bylo uvedeno výše, syntaxe jazyka nedefinuje příkazy pro aritmetické operace. Všechny výpočty či porovnání jsou prováděny pomocí funkce `expr`. Chceme-li volat funkci jinde než na začátku řádky, je potřeba ji vložit mezi hranaté závorky. Použití uvádí následující příklad, který do proměnné `delta` dosadí rozdíl hodnot proměnných `value` a `desired`.

```
set delta = [expr $value - $desired]
```

K vytváření vlastních uživatelských funkcí slouží klíčové slovo `proc` následované seznamem parametrů a tělem funkce. Vzhledem k neexistenci typů proměnných se nerozlišují

funkce s návratovou hodnotou od procedur bez návratové hodnoty; funkce může vrátit jakoukoli hodnotu příkazem `return`. Jsou-li funkce součástí jmenného prostoru, musí být jejich kód uveden uvnitř definice tohoto prostoru, jak ukazuje následující příklad. Vytvořme namespace `KUKA` se dvěma prázdnými funkcemi `Move` a `WaitForMotionEnd`.

```
namespace KUKA {
  proc Move {motionID} {
    ...
  }
  proc WaitForMotionEnd {} {
    ...
  }
}
```

Pro zapisování podmíněných výrazů slouží funkce `if`, která má tvar

```
if {$voltage > $threshold} {
   set $exceeded = true
}
```

Cykly jsou vytvořeny funkcí `for` následovanou inicializací, podmínkou, inkrementem a tělem smyčky, jak uvádí následující příklad zvýšení hlasitosti o 10 kroků.

```
for {set i 0} {$i<10} {incr i} {
  ROBOT::Move VOLUME_UP
}
```

Jinou možností je použití funkce `while`, následované pouze podmínku a tělem smyčky.

Dále se věnujme programovacímu jazyku určenému k ovládání robota.

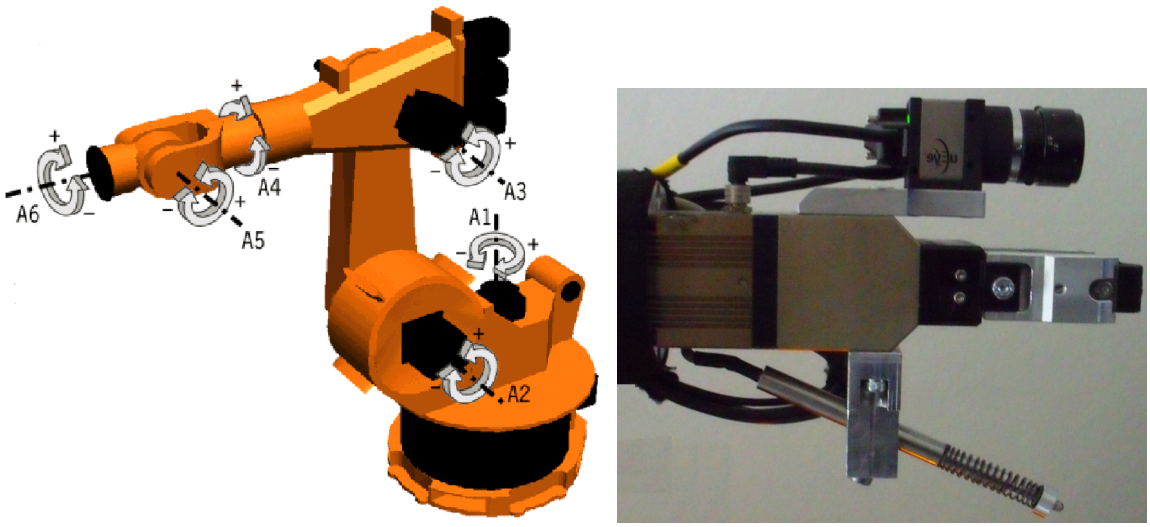
2.2 Robot KUKA kr3

Průmyslový robot kr-3 firmy KUKA [11] je manipulátor se šesti stupni volnosti, znázorněný na obr. 2.3. Na přírubě robota jsou upevněny dva nástroje: elektrický úchopový mechanismus (greifer) a pružný kolík, který slouží ke stisknutí tlačítek.

Otevírání a zavírání greiferu je ovládáno dvěma signály, kterými je napájení a směr pohybu. Rozteč čelistí lze tedy ovlivnit pouze délkou přivedeného napájecího pulsu. Změna rozteče v celém rozsahu pohybu je provedena za cca 700 ms. Systém je navíc vybaven

dvěma indikátory koncové polohy, jejichž binární výstup lze použít jako ochranu proti zničení šnekového převodu v případě příliš dlouhého napájecího pulsu.

Síla působící při stisku tlačítek je měřena elektromagnetickým snímačem. Výstupní signál snímače je připojen do analogové části vstupně výstupního modulu kontroléru robota.



Obrázek 2.3: Manipulátor KUKA kr-3 a detail nástrojů

2.2.1 Komunikace

Aby se mohl robot pohybovat po požadovaných trajektoriích, je potřeba v jeho kontroléru spustit řídicí program definující pohyby nástroje. Pro zadávání povelů z nadřazeného testeru je dále nutno umožnit vzdálené ovládání. Firma KUKA poskytuje pro účely vzdáleného přístupu *OPC server*, pomocí kterého lze sledovat a měnit globální proměnné programu.

Technologie OPC (OLE for Process Control) poskytuje unifikované propojení aplikací založených na OS Windows přímo s hardwarem pro řízení technologií. Jedná se o otevřený standard zajišťující nepřetržité získávání dat z připojených hardwarových zařízení. Metody získávání dat jsou nezávislé na typu připojeného zařízení. Tvorbou, rozšiřováním a šířením OPC specifikací se zabývá organizace OPC Foundation. Technologie OPC vychází z technologií pro tvorbu komponentových aplikací – COM (Component Object Model) a DCOM (Distributed Component Object Model).

Ovládací aplikace automatizovaného testeru, která je spuštěna na hlavním počítači, vystupuje v komunikaci jako OPC klient. Datová komunikace mezi hlavním počítačem

a kontrolérem robota probíhá po lokální síti LAN. Po inicializaci spojení mohou být přenášeny hodnoty proměnných typu textový řetězec, celá a reálná čísla.

2.2.2 Programovací jazyk robota

Robot může pracovat ve dvou režimech: *manuálním* a *automatickém*. Manuální režim je určen pro takzvané učení, tedy navedení robota do žádané (např. referenční) pozice a její následné vložení do programu tak, aby byla později dosažena v režimu automatickém. Ovládat pozici ramene lze v souřadné soustavě dané šesti osami kloubů robota A1 – A6, světovými souřadnicemi, nebo v soustavě spojené s pohyblivým nástrojem.

2.2.2.1 Druhy pohybu

Automatický režim práce robota je řízen programem, napsaným ve speciálním jazyce KRC. Tento jazyk umožňuje programovat pohyb referenčního bodu nástroje mezi definovanými body v prostoru. Inverzní kinematická úloha, tedy transformace polohy nástroje v kartézských souřadnicích na úhly jednotlivých kloubů robota, je transparentně řešena řídicí aplikací robota. Použitý robot umožňuje definovat tři druhy pohybu:

- point–to–point (PTP), kdy referenční bod nástroje provede nejrychlejší, předem nepredikovatelný pohyb. Kontrolér vypočte řešení inverzní transformace pouze pro koncový bod a potom všechny klouby synchronně přestaví do nové polohy.
- lineární pohyb (LIN), kdy se nástroj pohybuje konstantní rychlostí po přímce a
- pohyb po kružnici (CIRC), kdy nástroj konstantní rychlostí opisuje kružnici danou třemi body.

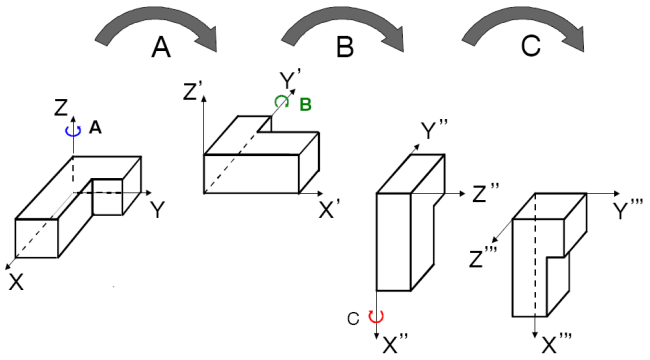
Základním parametrem lineárního a kruhového pohybu je rychlost v ms^{-1}. Pro pohyb z bodu do bodu (PTP) se potom rychlost udává v procentech maximální rychlosti, kde maximální rychlost je dána dominantním kloubem, který musí vykonat nejdelší pohyb. Dále je možno omezit akceleraci jednotlivých kloubů; v případě ponechání výchozí hodnoty maximální akcelerace hrozí zastavování programu z důvodu přetížení dominantního pohodu. Příkazy jazyka KRC pro nastavení pohybových parametrů jsou uvedeny v tabulce 2.1.

Každý *bod v prostoru*, do kterého má být naveden nástroj, musí být definován minimálně šesti parametry – kartézskými souřadnicemi x, y, z a úhly natočení kolem os z, y a x postupně a, b, c (obr. 2.4). Protože robot může určitou podmnožinu bodů v pracovním

parametr	příkaz	nastavená hodnota
rychlost LIN, CIRC	BAS(#VEL_CP, 2.0)	2 ms^{-1}
rychlost PTP	BAS(#VEL_PTP, 100)	100 %
zrychlení PTP	BAS(#ACC_PTP, 80)	80 %

Tabulka 2.1: Příkazy pro nastavení parametrů pohybu

prostoru dosáhnout více způsoby (obr. 2.5), jsou navíc zavedeny rozšiřující stavy *Status*(S) a *Turn* (T), jejichž přidáním ke kartézským souřadnicím je zajištěn jednoznačný výsledek inverzní transformace, tedy jednoznačná poloha všech šesti kloubů A1–A6.



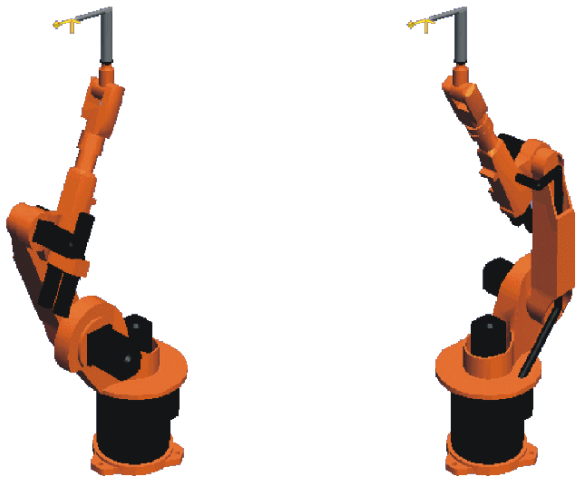
Obrázek 2.4: Způsob získání souřadného systému ze šesti parametrů

Pokud definujeme souřadný systém pevného předmětu `$base` a souřadný systém nástroje upevněného na přírubě robota `$tool`, budou pohyby robota automaticky přepočteny do souřadného prostoru předmětu. Toho lze využít pro manipulaci s více stejnými předměty na různých místech v prostoru pomocí stejného algoritmu s numericky shodnými parametry.

2.2.2.2 Funkce a řízení programu

Základní předdefinované typy *proměnných* jsou shrnuty v tabulce 2.2. V případě potřeby je možno za jejich použití definovat vlastní datové struktury. Proměnné je nutno před jejich prvním použitím deklarovat pomocí příkazu `DECL typ název` a naplnit korektní inicializační hodnotu.

V jazyce KRC je možné vytvářet *procedury* bez a *funkce* s návratovou hodnotou. Oběma typům je možno předat volitelné vstupní parametry. Jako příklad uveďme proceduru pro přivření čelistí greiferu, jejímž parametrem je doba sepnutí napájecího pulsu šnekového pohonu.



Obrázek 2.5: Příklad dvou různých pozic robota se stejnou polohou nástroje

Název	Typ proměnné
`BOOL`	binární hodnota
`INT`	celé číslo
`REAL`	reálné číslo
`FRAME`	souřadná soustava {X, Y, Z, A, B, C}
`POS`	rozšířená poloha v 3D {X, Y, Z, A, B, C, S, T}
`AXIS`	úhly natočení jednotlivých os {A1, A2, A3, A4, A5, A6}

Tabulka 2.2: Typy proměnných v jazyce KRC

```
DEF ZAVRIT(time:IN)
  DECL REAL time
  SMER = TRUE
  NAPAJENI = TRUE
  WAIT SEC time
  NAPAJENI = FALSE
END
```

Pro řízení *toku programu* v jazyce KRC slouží

- nepodmíněný skok `GOTO`,
- podmínky `IF podmínka THEN ... ELSE ... ENDIF`,

- čítací cyklus `FOR ... ENDFOR`,
- smyčka `WHILE podmínka ... ENDWHILE` a
- větvení pomocí `SWITCH`

Z programátorského hlediska jednoduše je řešena *transformace souřadnic,* například pohyb o 50 mm ve směru *osy z nástroje* provede následující fragment zdrojového kódu

```
DECL FRAME souradnice_nastroj
posun = {x 0,y 0,z 50,a 0,b 0,c 0}
souradnice_nastroj = $pos_act
LIN souradnice_nastroj:posun
```

kde operátor „:“ (dvojtečka) provádí právě transformaci souřadnic a systémová proměnná `$pos_act` obsahuje vždy aktuální pozici nástroje v pracovním souřadném systému.

2.2.2.3 Čítače a přerušení

Přerušení mohou být vyvolána například časovačem, nebo vnějším binárním i analogovým signálem. Příkladem binárního vstupu je koncový spínač čelistí greiferu. Deklarace přerušení určuje prioritu, spouštěcí podmínku a funkci, která je volána jako obsluha. Například vypnutí napájení greiferu po dosažení koncové polohy otevření čelistí, kde signál snímače je pojmenován `SOPEN`, zajistíme následovně.

```
INTERRUPT DECL 10 WHEN (SOPEN == TRUE) DO greifer_stop()
```

Speciálním případem přerušení je situace, kdy potřebujeme *předčasně ukončit* naprogramovaný pohyb. Okamžitého zastavení pohybu dosáhneme použitím příkazu `BRAKE`. Následuje-li po něm příkaz `RESUME`, dojde k přerušení vykonávání všech vnořených funkcí a návratu programu do úrovně, ve které bylo přerušení deklarováno. Takové přerušení je v řídicím programu použito při mačkání tlačítek, kdy je pohyb ukončen po vyvinutí dostatečné síly. Zdrojem tohoto přerušení je překročení mezní hodnoty odpovídajícího vstupního analogového napětí ze snímače tlaku. Velikost vstupního napětí na A/D převodnících je uložena v proměnných `$ANIN[1]` ...`$ANIN[32]`. Vstupní napětí v rozsahu $\langle -10\,V; 10\,V\rangle$ je mapováno na reálnou hodnotu v rozsahu $\langle -1{,}0; 1{,}0\rangle$. Deklarace přerušení aktivovaného analogovým signálem po překročení napětí 1,5 V je následující.

```
INTERRUPT DECL 6 WHEN $ANIN[1]>0.15 DO button_stop()
```

Programátor má dále k dispozici 16 časovačů. Hodnota časovače uložená v proměnné `$TIMER[i]` je inkrementována každou milisekundu. Příznak každého časovače označený `$TIMER_FLAG[i]` je nastaven, pokud má hodnota časovače kladnou hodnotu. Případné přerušení, citlivé na hranu signálu, je tedy generováno po inkrementaci $0 \rightarrow 1$. Pro spouštění a zastavování časovače slouží systémová proměnná `$TIMER_STOP[i]`, která je v okamžiku spuštění programu nastavena na `TRUE`, kdy je čítač zastaven. Vykonání určité funkce za 2,5 s po spuštění časovače provede následující fragment kódu.

```
INTERRUPT DECL 3 WHEN $TIMER_FLAG[1] DO funkce()
$TIMER[1] = -2500
$TIMER_STOP[1] = FALSE
```

2.2.2.4 Advance mode

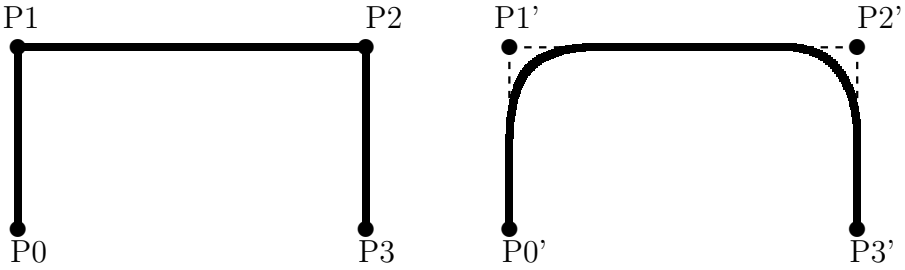
Řídicí program robota je specifický možností „dopředného“ vykonávání instrukcí. Je-li povolen tzv. *advance run mode*, muže kontrolér v předstihu vypočítat trajektorie až pěti pohybů následujících za pohybem právě prováděným. Počet připravených trajektorií volíme hodnotou systémové proměnné `$advance`. V čase výpočtu musí být však k dispozici všechny hodnoty potřebné pro výpočty, nebo vyhodnocení podmínek. Dopředné vykonávání je automaticky pozastaveno například při práci se vstupy a výstupy.

Smysl tohoto módu je především v urychlení reakcí manipulátoru, protože časově náročná úloha inverzní kinematiky může být vyřešena během vykonávání předcházejícího pohybu a není potřeba mezi dvěma pohyby zbytečně čekat na výsledek složitého výpočtu. Následující pohyb tak započne okamžitě po skončení pohybu předchozího.

Dalším aspektem je možnost povolit aproximaci pohybů, kdy není přímo dosažen zadaný koncový bod, ale pohyb plynule přechází na další zadanou trajektorii (obr. 2.6). V případě, že není potřeba nástroj navést přímo do cílové pozice, se tak plynulostí změn pohybu snižuje mechanické zatížení celého systému, pohyb je rychlejší a pro pozorovatele se jeví přirozeněji.

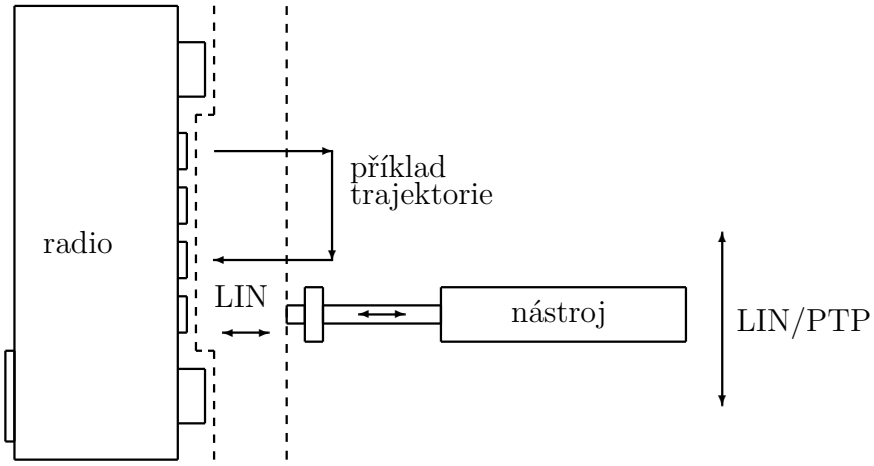
2.2.3 Úpravy řídicího algoritmu

Protože místo pro připevnění kamery bylo zvoleno na přírubě robota, vznikla potřeba úprav řídicího skriptu s důrazem na snížení jejího mechanického zatížení. Úkolem byla eliminace nadbytečných pohybů při zachování stávající bezpečnostní oblasti a dále zvýšení plynulosti pohybu s minimem akcelerace.



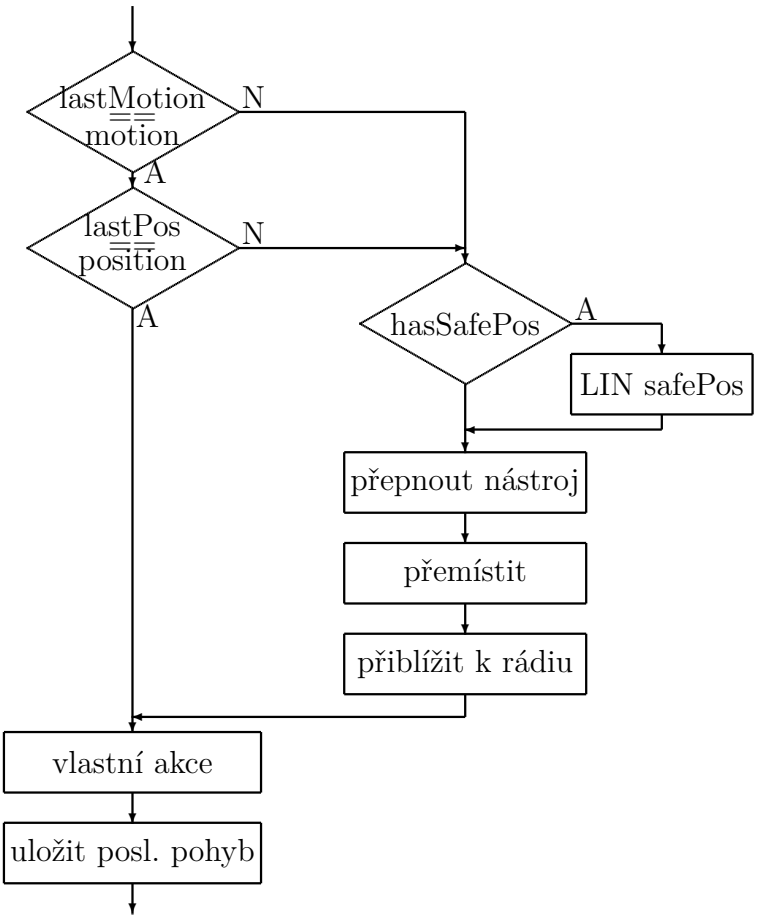
Obrázek 2.6: Trajektorie P0 → P3: (a) bez aproximace, (b) s aproximací

Na obrázku 2.7 je z perspektivy znázorněn pohled na rádio spolu s vymezením zón pohybu. Z bezpečnostních důvodů je vymezena zóna před rádiem, ve které se smí nástroj pohybovat pouze lineárními pohyby, a to jen ve směru kolmém na rovinu rádia. Je-li požadován přesun nad jiný ovládací prvek, je potřeba nejprve opustit tuto zónu, tedy vzdálit nástroj od rádia, pak přesunout nástroj rovnoběžně s rádiem a nakonec přiblížit na novou pozici.



Obrázek 2.7: Zóny pohybu robota s příkladem trajektorie

Původně byl návrat mimo lineární zónu prováděn po každém pohybu, což bylo zbytečné při vícečetném použití jednoho ovládacího prvku. Pro optimalizaci počtu pohybů byly zavedeny proměnné polohy `safePosition`, do které je na konci každého pohybu uložena odpovídající klidová poloha mimo lineární zónu, dále binární `hasSafePosition` sloužící pro oddělení pohybů, které z principu končí mimo lineární zónu a nevyžadují tedy přesun do klidové polohy a nakonec `lastMotion` a `lastPos` pro uložení posledního pohybu a odpovídajících souřadnic. Hodnota `TRUE` proměnné `hasSafePos` značí, že poslední dokončený druh pohybu disponuje speciální bezpečnostní polohou `safePos` a v případě potřeby musí být robot do této polohy naveden nejdříve. Na obrázku 2.8 je pomocí vývojového diagramu znázorněn upravený algoritmus přemisťování nástroje.



Obrázek 2.8: Pohyby robota – vývojový diagram

Optimalizace pohybu je tedy provedena tak, že například po stisknutí tlačítka je nástroj oddálen pouze o 2 mm, kde čeká na další povel. Je-li tímto povelem stisknutí totožného tlačítka, je možné stisk ihned vykonat a dochází k časové úspoře oproti původní koncepci pohybů, kdy byl nástroj před každým pohybem vzdálen vždy přibližně 15 cm od roviny rádia. V případě, že je požadován přesun nástroje nad jiný ovládací prvek, je tento proveden pomocí plynulého pohybu bez zastavení v bezpečnostní pozici a nedochází k žádnému prodloužení odezvy vůči původnímu řídicímu algoritmu. Aby však nástroj nesetrvával v těsné blízkosti rádia neustále, je s každým ukončeným pohybem vynulován a spuštěn časovač, který odměřuje dobu neaktivity. Pokud od posledního pohybu uplyne více než 5 s, nelze pravděpodobně očekávat další povel a nástroj je automaticky přesunut do bezpečnostní pozice dále od rádia. Interval neaktivity je měřen časovačem, jehož způsob použití byl popsán v sekci 2.2.2.

Z důvodu snížení mechanického zatížení celé mechanické části systému a především pak kamery je při přemisťování hlavice s nástroji využit *advance* mód spolu s aproxi-

mací pohybů, jak bylo zmíněno v předchozí sekci. Zavedení *advance* módu vyžadovalo další změny v algoritmu takové, aby nedocházelo k automatickému přerušení dopředných výpočtů čekáním na předem neznámou hodnotu proměnné, která závisí na skutečném aktuálním stavu systému a nelze stanovit s předstihem.

2.2.3.1 Řídicí program

Algoritmus pro řízení robota v popisovaném systému automatizovaného testeru se skládá z několika částí:

- deklarace proměnných,
- definice pevných bodů v pracovním 3D prostoru,
- inicializace časovače a přerušení,
- povinného PTP pohybu do výchozí polohy a
- nekonečné řídicí smyčky se stavovým automatem.

Pomocí stavového automatu (SA) v nekonečné smyčce `loop` jsou prováděny všechny pohyby robota: mačkání tlačítek, otáčení knoflíků, otevírání a zavírání greiferu nebo pohyb do pozice snímání obrazu.

Každému pohybovému příkazu odpovídá jeden stav hlavního SA. Stav automatu je reprezentován celočíselnou hodnotou a je uložen v proměnné `motionSate`. Po spuštění programu je automat ve stavu „nenakalibrováno“. Před započetím prvního pohybu je proto nutné provést kalibraci na daný model rádia. Kalibrace spočívá v načtení souřadnic referenčního bodu, ke kterému se vztahují pozice všech ovládacích prvků. Následné povely pro stisknutí tlačítka nastavují jen relativní souřadnice ovládacího prvku a dobu stisku, povely pro otáčení pak pozici a úhel otočení.

Pro účely snímání obrazu byl vytvořen nový povel `ROBOT::ViewDisplay` a jemu odpovídající stav hlavního SA. Tento povel namíří kameru na rádio tak, aby byla dosažena minimální ostřicí vzdálenost kamery. Bylo potřeba nalézt takový kompromis rychlosti a zrychlení pohybu od rádia do snímací polohy, aby byl přesun dostatečně rychlý a nedošlo ke změně zobrazeného údaje, ale zároveň aby nebyl příliš prudký a neotřásala se celá konstrukce testeru a nehrozilo poškození kamery.

Porovnání původní a nové verze řídicího algoritmu je uvedeno v kapitole Experimety.

2.3 Kamera uEYE SE

Výběr kamery lze rozdělit na dva podproblémy, a to výběr vlastní kamery se snímacím čipem a výběr vhodného objektivu. Tělo kamery bylo vybíráno v souladu s původním záměrem použití SW TomLine, byla zvolena barevná kamera *IDS USB uEYE UI-2240SEC* s rozlišením 1280×1024 px a maximální vzorkovací frekvencí 15 snímků za sekundu [5].

Jelikož je kamera umístěna na rameni robota, musí objektiv umožňovat ostření na krátkou vzdálenost, aby nemusel být vykonáván příliš velký pohyb od ovládacích prvků rádia do snímací polohy. Zároveň nesmí při pohybech robota docházet k samovolným změnám zaostření nebo nastavení clony, je tedy potřeba, aby objektiv disponoval aretací obou nastavení. Ohnisková vzdálenost, která ovlivňuje šířku záběru, musí být taková, aby bylo v záběru celé rádio, avšak aby ještě nedocházelo k nadměrnému geometrickému zkreslení obrazu (vinětaci). Z několika různých objektivů byl nakonec vybrán objektiv ohniskem 8 mm a minimální ostřicí vzdáleností 30 cm.

Nejdůležitější parametry jsou shrnuty v tabulce 2.3.

Parametr	hodnota/rozsah
Model	USB uEYE UI-2240SE-C
Rozlišení	1280×1024 (1,3 MPx)
Senzor	CCD 1/2"
Barevná hloubka	32 bit
Vzorkovací perioda	max 15 fps
Ostřicí vzdálenost	30 cm – 1
Ohnisková vzdálenost	8 mm

Tabulka 2.3: Vybrané parametry použité kamery

Pro komunikaci s kamerou je od výrobce k dispozici aplikační rozhraní (API) ve formě dynamické knihovny DLL [4]. Knihovna `ueye_api.dll` poskytuje přes 130 funkcí pro inicializaci a odpojení kamery, přes získávání obrazu, správu paměti a nastavování parametrů snímání, po ukládání obrázků a videí. Zajímavé pro nás budou funkce uvedené v tabulce 2.4.

Posledním důležitým parametrem je výstupní formát obrazu. Zvolit lze obraz černobílý nebo barevný s různou bitovou hloubkou. Obrazové formáty kompatibilní s použitými nástroji pro zpracování obrazu uvádí tabulka 2.5.

Vlastní API pak používá pro přístup ke kameře komponentový model COM. Protože

Název	funkce
`is_InitCamera()`	inicializace kamery
`is_ExitCamera()`	odpojení kamery
`is_SetColorMode()`	nastavení barevného modelu
`is_FreezeVideo()`	sejmutí jednoho snímku
`is_CopyImageMem()`	přesun bitmapy na dané paměťové místo

Tabulka 2.4: Vybrané funkce pro práci s kamerou

ColorMode	BitsPerPixel	formát
IS_SET_CM_Y8	8	černobílý (1 kanál), 8 bpp
IS_SET_CM_RGB32	32	barevný (4 kanály), 32 bpp

Tabulka 2.5: Výstupní formáty obrazu kamery kompatibilní s OpenCV

připojení kamery inicializuje systém COM a vypnutí jej opět deaktivuje, docházelo po odpojení kamery k přerušení komunikace s OPC serverem. Technologie OPC je totiž postavena na COM a připojením všech balíčků s DLL knihovnami do jednoho procesu TCL konzole (viz obr. 1.1) došlo ke vzájemnému ovlivnění komponent kamery a OPC serveru. Aktivace a deaktivace COM se vždy vztahuje pouze k jednomu vláknu (thread), problém se tedy podařilo odstranit vytvořením nového vlákna, které obstarává pouze komunikaci s kamerou.

2.4 Nedostatky stávajícího systému

Veškeré úkony s ovládacími prvky na čelním panelu rádia lze bez optického vstupu provádět pouze v přímé vazbě. V případě otočných knoflíků dochází vlivem nesymetrického uchopení knoflíku robotem a nepřesnosti úhlu natočení k odskočení nastavené polohy do vedlejší aretované pozice. Dojde-li k odskočení, je měněný parametr nastaven na chybnou hodnotu. Bez optického vstupu neexistuje žádná možnost, jak ověřit správný efekt použití ovládacích prvků.

Rádia uchovávají svá nastavení v perzistentní paměti, proto k obnovení továrního nastavení nedochází ani po odpojení napájení, a proto se rádio po zapnutí nachází v předem neznámém stavu. Příprava rádia a jeho nastavení do definovaného počátečního stavu je časově velmi náročné, přitom by v mnoha případech stačilo jen vizuálně ověřit, zda jsou všechny parametry správně nastaveny, a korigovat jen ty, které je potřeba měnit.

Pro tunerové testy se používá vlastní modulátor rádiového signálu, připojený přímo na anténní vstup rádia. Rádio však může při automatickém ladění nalézt kromě testovací stanice i jinou skutečnou rádiovou stanici. Bez optického vstupu lze toto odhalit pouze pomocí analyzátoru, který porovná vysílaný signál se zvukovým výstupem rádia. Toto měření může výrazně prodloužit dobu testu, pokud se proces ladění často opakuje.

Jako poslední příklad situace, ve které se projevuje absence vizuální informace, uveďme aktivaci či deaktivaci volby TP (traffic program). Je-li TP aktivováno, sleduje rádio, zda není podle zpráv RDS vysíláno dopravní zpravodajství. Zpravodajství je signalizováno příznakem TA (traffic announcement) a rádio po jeho přijetí zesílí hlasitost a v případě přehrávání z jiného zdroje (CD, CD–měniče,externího vstupu AUX, paměťové karty) přepne dočasně na rádio. Tlačítko volby TP má funkci přepínače a bez informace z kamery nelze stanovit, zda je funkce aktivní či vypnutá. Je tedy potřeba nalézt speciální sekvenci úkonů (stisku různých tlačítek za splnění dalších speciálních podmínek), která bez ohledu na počáteční stav nastaví nebo zruší volbu TP. Samotné nalezení takové sekvence může být časově náročné, nebo taková sekvence ani nemusí existovat.

Kapitola 3

Optické rozpoznávání pro testy

V předcházející kapitole byly uvedeny některé příklady nedostatků stávajícího systému, který postrádal informaci o vizuálním stavu rádia. Nyní navrhneme způsob využití kamery v testech a zdůvodníme význam takového rozšíření. Dále bude uveden přehled nástrojů použitých pro řešení možných úloh optického rozpoznávání, následovaný představením vlastní implementace navrženého rozšíření.

3.1 Optická zpětná vazba

Prvním způsobem využití obrazového vstupu je zavedení zpětné vazby, která pomůže urychlit či zpřesnit vykonávání některých úkonů. Uveďme několik konkrétních příkladů.

3.1.1 Odkódování rádia

Chronologicky prvním funkčním testem rádia je prvotní zapnutí a zadání (známého) bezpečnostního kódu rádia. Rádia jsou kódována z důvodu prevence proti krádeži; kód je vyžadován po každém odpojení napájecího napětí. V automobilu může být kód uložen v kombi–jednotce, ze které je získán bez zásahu uživatele. My se však budeme zabývat odkódováním ručním, kdy uživatel musí pomocí funkčních kláves na rádiu zadat bezpečnostní kód, aby mohl zařízení začít používat. Po zapnutí napájení se na displeji objeví čtyřmístné číslo, které je potřeba změnit na požadovaný kód. Inicializační hodnota tohoto čísla může být 0000, nebo také 1000, či nemusí být předem známa. Při známé inicializační hodnotě musí být tato pevně zadána v parametrech testů rádia, byla-li by náhodná, nebude možno automatizované odkódování provést.

Za předpokladu, že budeme mít s pomocí kamery informaci o údaji zobrazeném na

displeji rádia, může proběhnout zjištění inicializačního kódu automaticky. Po nastavení bezpečnostního kódu, který musí být stále vložen jako parametr testu, může být provedena kontrola, zda bylo zadání úspěšné – některá rádia mohou úmyslně vykazovat chybu při zadávání čísel. Kontrola je důležitá proto, že dvojnásobným zadáním chybného kódu dojde k zablokování rádia. V případě chybně zadané hodnoty proběhne ihned oprava bez zásahu uživatele. Je-li zadaný kód shodný s kódem bezpečnostním, je tento potvrzen a další testy zařízení mohou následovat.

3.1.2 Nastavení hlasitosti

Nastavení hlasitosti je obvykle prováděno otáčecím knoflíkem s inkrementálním snímačem polohy. Aby bylo možno bez zpětné vazby nastavit požadovanou hlasitost, je nutno nejprve knoflíkem otočit doleva natolik, aby s naprostou jistotou došlo k nastavení hlasitosti na nulu. Poté je teprve možno otočit o vypočtený úhel zpět a nastavit tak požadovanou úroveň hlasitosti.

Vlivem nepřesnosti použitých levných inkrementálních senzorů či nerovnoměrným uchopením knoflíku robotem dochází nepředvídatelně k odskočení knoflíku na konci pohybu o jeden krok vpřed nebo vzad. To znamená, že akustické parametry rádia budeme měřit se špatně nastavenou hlasitostí a vznikne proto falešná chyba.

Optická zpětná vazba umožní nastavit požadovanou hlasitost pouhou relativní změnou vzhledem k aktuální hodnotě. Stačí vyvolat nastavování hlasitosti a z obrazovky přečíst aktuální hodnotu. Robot otočí knoflíkem o potřebný počet kroků a nakonec je provedena kontrola dosažené hodnoty s případnou korekcí. Proces nastavení žádané úrovně hlasitosti lze tedy pomocí kamery urychlit a zpřesnit.

Úkolem je potom v první řadě zjistit, jaká úroveň hlasitosti je rádiem indikována. Pokud rádio zobrazuje aktuální hlasitost číslem, lze hodnotu přečíst pomocí OCR. Pokud je zobrazen jen grafický prvek *progressbar*, je potřeba získat aktuální hodnotu z jeho tvaru.

Jiným, avšak velmi podobným problémem je například nastavení vyvážení levého a pravého zvukového kanálu. Zde je nutno číst údaj indikovaný posuvníkem s nulou uprostřed stupnice, protože mohou být zobrazeny jak kladné, tak záporné hodnoty.

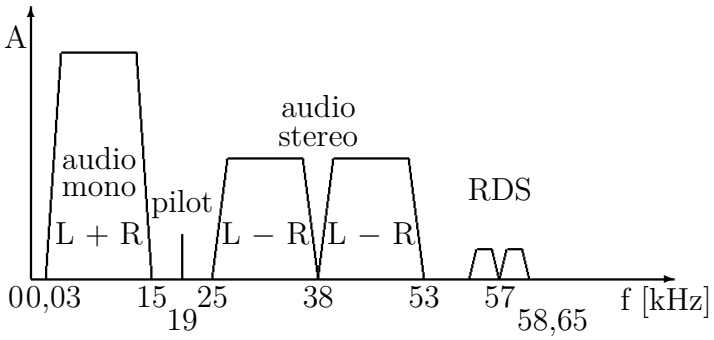
3.1.3 Kontrola volitelných funkcí

V sekci 2.4 byla zmíněna například volba TP. Funkce Traffic Program je natolik důležitá, že její indikátor bývá stále zobrazen na hlavní obrazovce rádia. Musí-li být například pro

určitý test rádia volba TP vypnuta, z obrazu kamery lze jednoduše určit její aktuální stav, a bylo-li TP zapnuto, jednoduše jej vypnout, obvykle samostatným tlačítkem.

3.1.4 Verifikace RDS

Kromě akustického signálu mohou rádiové stanice vysílat další textové a stavové informace pomocí systému RDS (Radio Data System). Služba RDS je pro každou rádiovou stanici modulována na trojnásobku 19 kHz pilotního signálu, původně určeného pro demodulování stereofonní části signálu. Kanál RDS má tedy ve spektru stanice vyhrazeno úzké pásmo kolem kmitočtu 57 kHz. Přenosová rychlost je pouze 1 187,5 bps, kanál je proto použitelný pouze pro textové přenosy. Rozložení spektra signálu před FM modulací znázorňuje obrázek 3.1. V kanálu RDS mohou být přenášeny různé informace, jako například alternativní frekvence stanice (AF), jméno stanice, druh programu nebo název přehrávané skladby.



Obrázek 3.1: Polovina spektra demodulovaného stereo FM signálu s RDS

Určitá skupina RDS textů je zobrazovnána na displeji rádia. Úkolem testeru může být například změřit, jak silný signál přijímaný z vysílače je potřeba pro příjem RDS, či kontrolovat, zda jsou texty správně zobrazeny. Testy týkající se RDS nelze bez kamery vůbec provádět.

3.2 Ovládání dotykového displeje

Rádia bez dotykového displeje mají fixní polohu ovládacích prvků. V případě dotykové obrazovky se může poloha tlačítek měnit v závislosti na dalších údajích na displeji zobrazených. Optické rozpoznávání umožní najít aktuální pozici hledaného tlačítka pomocí textu, který je na něm vyznačen; tato metoda může ovšem selhat ve chvíli, kdy bude na displeji zobrazen vícekrát stejný text.

Rozpoznávání textu by mohlo také z pohledu vytváření testů usnadnit procházení menu rádia. Bude-li známa pouze stromová struktura menu, stačí určit koncovou volbu, kterou chceme vybrat, tedy list takového stromu. Postup procházení struktury menu, stejně jako určení souřadnice jednotlivých položek na displeji, můžou být řešeny automaticky.

3.3 Rozpoznávání barev

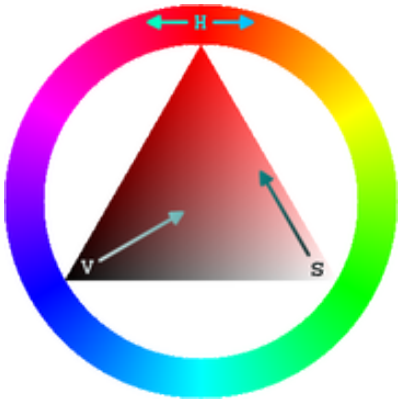
S rozšiřující se škálou multifunkčních rádií s barevným displejem přichází potřeba zpracování barevného obrazu. Různé indikátory zapnuto/vypnuto reprezentují svůj stav pomocí barev (např. zelená/červená), které nelze jednoduše rozeznat z černobílého obrazu. Pro rozpoznávání základních odstínů barev lze s výhodou použít barevný model HSV[18]. Tento model nejblíže koresponduje s lidským vnímáním barev. Každá barva je specifikována třemi složkami, které tomuto barevnému modelu daly název:

Hue reprezentuje převládající barevný tón, udává se jako poloha na standardním barevném kole (0° – 360°),

Saturation udává sytost barvy. Představuje množství šedi v poměru k barevnému odstínu, měří se v procentech od 0% (šedá) do 100% (plně sytá barva),

Value lze potom chápat jako hodnotu jasu, nebo množství bílého světla.

Složka H (viz obr. 3.2) tedy přímo reprezentuje odstín barvy a lze ji využít pro roztřídění nebo filtrování pixelů obrázku mezi základní používané indikační barvy, jakými jsou především červená, zelená nebo žlutá.



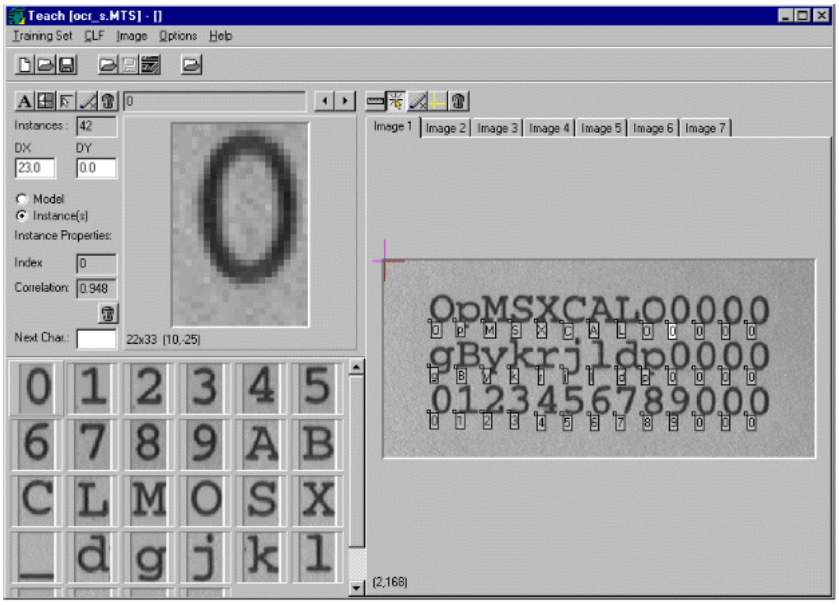
Obrázek 3.2: Grafická reprezentace HSV kruhu

3.4 TOM Line

Pro všechny úlohy optického rozpoznávání měl být dle zadání použit software TOMLine (Teachable Object Model). Jedná se o nástroj postavený na systému pro zpracování a analýzu obrazu Common Vision Blox.

Funkcionalitu produktu můžeme rozdělit do dvou částí: rozpoznávání symbolů a analýza jasu či barvy vybrané části obrazu.

Do rozpoznávání symbolů spadá i čtení znaků, TOM Line však nedisponuje žádnou vestavěnou databází znaků. Pro učení symbolů je určena aplikace *Teach–module*. V té otevřeme obrázek se vzorovým textem. Vybereme obdélník kolem prvního písmene, zvolíme referenční bod a pojmenujeme vybraný znak. K této nové instanci pak přiřadíme alespoň dalších 8 výskytů stejného znaku stejné velikosti. Do ohraničení písmene je potřeba zahrnout i volný prostor za ním, odpovídající mezeře mezi písmeny. Je vhodné volit ohraničení vždy stejně vysoká pro všechna písmena. Jedná se o velmi zdlouhavý a nijak neautomatizovaný úkon. Příklad okna aplikace během učení znaků je znázorněn na obr. 3.3. V levé části vidíme velký obrázek jedné z instancí, cifru 0. Pod ní se nachází tabulka všech učených znaků a nakonec v pravé polovině je jeden ze zdrojových obrázků s fontem určeným k naučení. U každého písmena vlevo dole je zvýrazněn referenční bod. Výsledkem procesu učení je tzv. *klasifikační soubor*.



Obrázek 3.3: Okno *Teach* při učení znaků

Pro rozpoznávání naučeného textu slouží program *Model-teach*, který implementuje model neuronové sítě spolupracující se dříve vytvořeným klasifikačním souborem. Z na-

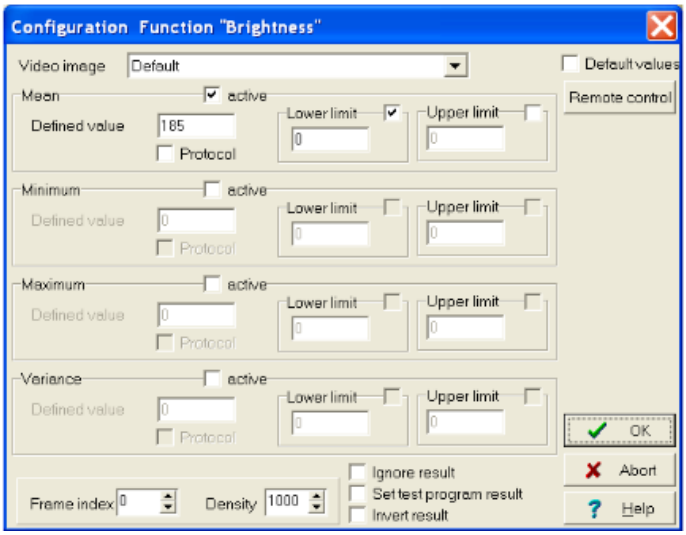
bídky funkcí vybereme rozpoznávání textu OCR, přiřadíme dříve vytvořenou databázi a vybereme ručně oblast, ve které má být text rozpoznán. Oblast je potřeba vybrat velmi přesně a rozpoznávaný font musí mít shodnou velikost s trénovacími daty.

Rozpoznávání znaků bylo pro demonstrační účely použito ke čtení čísla skladby přehrávané z kompaktního disku, kdy se na displeji objevuje nápis *track 01, track 02, ...* Stačilo tedy naučit pět různých písmen a deset cifer. I přes výbornou obrazovou kvalitu a výrobcem doporučený rozsah trénovacích dat nebyly výsledky uspokojivé, TOMLine opakovaně vynechával písmeno „c“. Po naučení celé abecedy se výsledky ještě zhoršily. Úspěšnost rozpoznávání byla velmi citlivá i na malé změny okolního osvětlení, které způsobí změnu jasu obrazu. Možným řešením problému se zdálo být předzpracování obrazu z kamery vlastním software tak, aby výsledkem byl vždy obraz s velmi podobným rozložením jasu, jak bude popsáno v části o knihovně OpenCV. Předzpracování obrazu přineslo kladné výsledky, nicméně spolehlivost rozpoznávání znaků nedosahuje spolehlivosti jiných, zdarma dostupných nástrojů.

Naučené symboly je možno použít i v další z funkcí TOMLine – vyhledávání symbolu v obraze. Zvolíme jeden symbol, který má být ve snímané scéně nalezen a lokalizován. Funkce potom v případě úspěchu vrátí souřadnice objektu, nebo hlásí selhání testu v případě, že symbol ve scéně není. Této funkce je možno využít pro testování různých příznaků, které jsou na displeji rádia zobrazeny v podobě symbolů. Příkladem může být symbol přeškrtnutého reproduktoru, který symbolizuje vypnutý zvuk. Symbol se může vyskytovat na různých místech displeje a proto není možné předem přesně určit oblast, kde má být hledán. Použijeme tedy funkci pro lokalizaci symbolu, z jejíhož výstupu využijeme pouze informaci, zda byl symbol na displeji rádia nalezen, nebo se ve snímané ploše nenachází a tedy zobrazen není.

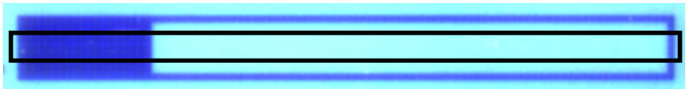
Je-li pozice hledaného symbolu předem známa, můžeme použít výpočetně mnohem jednodušší funkci pro analýzu jasu, *Brightness-check*. Program v zadané oblasti spočte průměrný jas obrazových bodů. Pokud tento spadá do zadaného intervalu, je výsledek testu kladný, jinak záporný. Dále je možné zadat omezení pro minimální a maximální jas či interval možné jasové variace. Funkce je vhodná pouze pro velmi kontrastní scénu s konstantním osvětlením, protože změna celkového jasu by mohla způsobit falešný výsledek testu.

Podobnou funkcí je počítání pixelů s jasem v určitém intervalu. Parametry funkce je minimální a maximální jas počítaných pixelů, návratovou hodnotou je počet pixelů v zadaném intervalu. Modifikace této funkce umožňuje navíc zadat povolený rozsah vý-



Obrázek 3.4: Nastavení parametrů funkce *brightness-check*

sledku a funkce vrací úspěch v případě, že je podmínka na počet pixelů splněna. Využití funkce počítání pixelů k určení hodnoty ukazatele hlasitosti spočívá ve výběru vhodné podoblasti ukazatele, jak vyznačuje obr. 3.5. Počet tmavých pixelů uvnitř naznačeného černého obdélníku je přímo úměrný zobrazené hodnotě hlasitosti. Známe-li maximální možný počet tmavých bodů, můžeme lehce spočítat zobrazovanou hodnotu.



Obrázek 3.5: Analýza ukazatele hlasitosti funkcí *pixel-count*

Nevýhodou funkce pro počítání pixelů je skutečnost, že musí být pevně zadána oblast, ve které má být výpočet proveden. Tato oblast je v SW TOM Line uložena v parametrech testu a nelze ji zadat „zvenčí“. Bez přímého zásahu uživatele je možno test jen spustit a získat výsledek. V konečném důsledku to znamená, že se při přesunu nebo změně velikosti progressbaru dané výměnou autorádia musíme znovu definovat všechny testy, ačkoli by stačilo mít k dispozici databázi se souřadnicemi, a ty vkládat jako parametr.

Spolu s nedostatečnou spolehlivostí rozpoznávání znaků se neukázal TOM Line jako vhodný pro plánované rozšíření automatizovaných testů. Proto jsme se rozhodli hledat jiný způsob řešení úlohy optického rozpoznávání, než je TOM Line. K takovému rozhodnutí vedla i neúměrně vysoká cena tohoto software.

Pro zpracování obrazu existují různé nástroje , mezi nimi volně dostupný balík OpenCV nebo AForge.NET. Vzhledem k orientaci aplikace TestAut na jazyk C++ byla pro další

vývoj zvolena knihovna OpenCV, neboť AForge je určeno pro jazyk C# a platformu .NET. Pro úlohu OCR pak byly testovány volně dostupné OCR engine Tesseract a gOCR.

3.5 Knihovna OpenCV

Knihovna OpenCV [2] (Open Source Computer Vision Library) byla vytvořena firmou Intel [6] a je poskytnuta volně k dispozici i pro komerční účely. Knihovna obsahuje rozsáhlou kolekci funkcí a tříd napsaných v jazyce C, implementující různé algoritmy pro zpracování obrazu a počítačové vidění. OpenCV je kompatabilní s knihovnou Intel Image Processing Library (IPL), které poskytuje nízkoúrovňové funkce pro zpracování obrazu. Knihovna OpenCV obsahuje rovněž relativně primitivní funkce jakými jsou binarizace, filtrování nebo statistické funkce nad obrázky, je však především zaměřena na funkce vyšší úrovně, především analýzu tvaru, segmentaci, rozpoznávání a dále kalibraci kamer, sledování objektů nebo 3D rekonstrukci obrazu.

Funkce knihovny jsou optimalizovány pro použití na procesorech Intel a výkonnost může být výrazně zvýšena využitím funkcí Intel Integrated Performance Primitives (IPP). OpenCV poskytuje platformně nezávislé rozhraní pro použití v jazyce C.

Knihovna je rozdělena do několika částí:

CXcore – definuje datové typy, dynamické struktury, funkce pro správu paměti a funkce pro kreslení

CV – obsahuje algoritmy zpracování obrazu a počítačového vidění

highgui – umožňuje vytváření jednoduchých dialogů pro interakci s uživatelem a definuje funkce pro načítání a ukládání obrázků

MLL – poskytuje funkce pro strojové učení

CVCam – zprostředkovává funkčnost některých kamer

Základním datovým typem je struktura `IPLImage`, která slouží pro uložení obrazové bitmapy. Tato struktura pochází z *Intel Image Processing Library* a knihovna OpenCV podporuje jen určitou podmnožinu parametrů. Definice datové struktury `IPLImage` je uvedena v příloze A.1. Struktura kromě bitmapy s vlastními obrazovými daty obsahuje další informace o obrázku, jako jeho rozměry, barevný model, bitovou hloubku či tzv. Area Of Interrest. Area Of Interrest (AOI) je aktivní obdélníková podoblast obrázku, která je

zpracovávána všemi funkcemi. Je-li AOI nastavena, zůstává zbytek obrázku mimo AOI nedotčen.

Protože bude obrazová část struktury `IPLImage` naplněna daty z kamery, je třeba zvolit vhodnou bitovou hloubku a počet kanálů obrázku, se kterým budeme pracovat. Kompatibilní formáty OpenCV a uEye byly uvedeny v tab. 2.5 v sekci věnované průmyslové kameře uEye.

Fragment zdrojového kódu v jazyce C++, který je určen k sejmutí barevného obrazu kamerou a jeho převedení do struktury `IPLImage` tak, aby mohl být následně zpracován pomocí knihovny OpenCV, je uveden v příloze A.2.

3.6 Zpracování obrazu

Objektiv kamery zaostřuje svazek světelného záření vycházející ze snímané scény na snímací čip CCD. Fotony dopadající na jednotlivé segmenty CCD čipu způsobují generování elektrického náboje. Naintegrovaný náboj je po určité době přesunut na vstup AD převodníku, který přímo určuje numerickou hodnotu jasu uloženou ve vytvořené bitmapě. Množství dopadajících fotonů, a tedy i jas pixelů výsledného obrázku, je pro totožnou scénu velmi závislý na světelných podmínkách, jakými je například počet, intenzita nebo umístění světelných zdrojů.

Aby bylo možno obraz dále zpracovávat a výsledky nezávisely právě na sebemenších změnách osvětlení, je potřeba obraz nejprve normalizovat tak, aby se vliv měnícího se osvětlení co nejvíce minimalizoval. Tato úloha může být řešena tzv. *ekvalizací obrazu* [13].

Často potřebujeme ve snímané scéně oddělit objekty od pozadí a ty následně podrobit dalšímu zkoumání. Takzvaná *segmentace obrazu* může být provedena na základě rozdílu jasu objektů a pozadí. Černobílý obraz můžeme binarizovat, tedy převést odstíny šedi na pouhé dvě barvy, černou a bílou. Takový proces nazýváme *prahováním.*

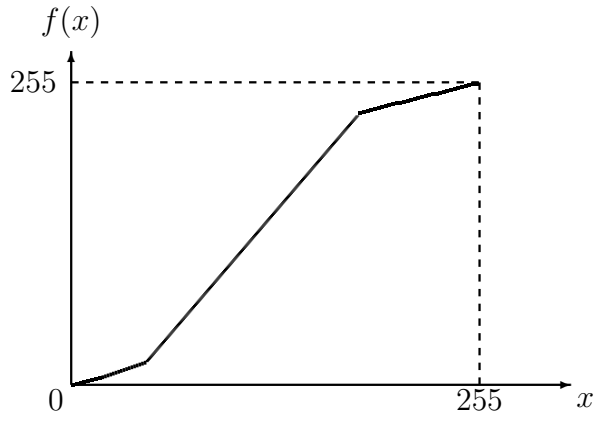
3.7 Ekvalizace obrazu

Jak uvádí [3], můžeme metody předzpracování obrazu roztřídit na

- transformace jasové stupnice,
- jasové korekce,
- geometrické transformace,

- lokální filtraci a
- restauraci obrazu.

Transformace jasové stupnice ovlivňuje rovnoměrně všechny body obrazu. Každému pixelu vstupního obrazu o jasu x v rozsahu $x \in \langle 0; 255 \rangle$ přiřazuje ve výstupním obraze hodnotu $f(x)$, kde f je obvykle nelineární funkce (např. zvýraznění kontrastu na obr. 3.6). Jednou z jasových transformací je ekvalizace histogramu. Jejím úkolem je pro pozoro-



Obrázek 3.6: Příklad jasové transformace – *zvýraznění kontrastu*

vatele – člověka zvýšit kontrast obrazu plným využitím jasové stupnice, nebo v případě strojového zpracování obraz jasově normalizovat.

Do procesu ekvalizace histogramu vstupuje obraz s histogramem $H(p)$ s jasovou stupnicí $p = \langle p_0; p_k \rangle$, cílem je nalézt monotónní transformaci jasové stupnice $q = \tau(p)$ takovou, aby výsledný histogram $G(p)$ byl rovnoměrný pro celý výstupní interval jasů $q = \langle q_0; q_k \rangle$, přičemž

$$\sum_{i=0}^{k} G(q_i) = \sum_{i=0}^{k} H(p_i) \tag{3.1}$$

Řešením je transformace daná kumulativním histogramem

$$\tau(p) = \frac{q_k - q_0}{N^2} \sum_{i=p_0}^{p} H(i) + q_0, \tag{3.2}$$

kde N je celkový počet pixelů v obrázku.

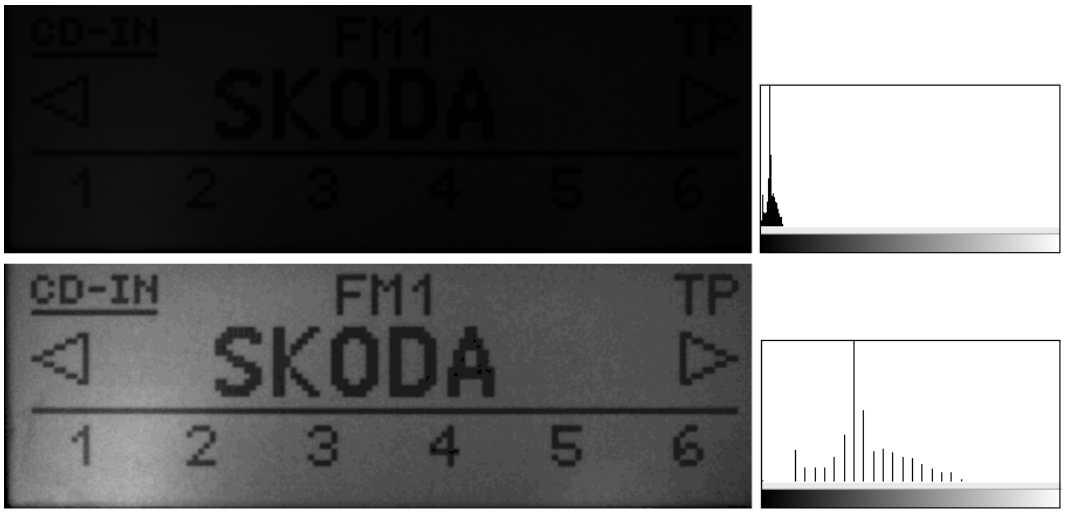
Ekvalizace histogramu potlačuje vliv změn okolního osvětlení na výstupní obraz. Předpokladem je ale zastoupení celého spektra jasů ve snímané scéně, což není případ monochromatického LCD displeje, který zobrazuje jediný odstín černé barvy. Ekvalizace

histogramu snímku displeje se snaží o rovnoměrné zastoupení všech jasů, což není očekávaný výsledek. Rádi bychom zachovali binární podstatu obrazu, ten však přesto nějakým způsobem jasově normalizovali.

Ukazuje se, že postačující normalizací je „napasování“ histogramu do stále stejného rozsahu jasu $q = \langle q_0; q_k \rangle$ při zachování prostorového rozložení jednotlivých čar histogramu. Taková transformace má tvar lomené křivky a je dána vztahem

$$\tau(p) = \begin{cases} q_{\min} & \text{pro } p < p_{\min} \\ \frac{(p-p_{\min})(q_{\max}-q_{\min})}{p_{\max}-p_{\min}} + q_{\min} & \text{pro } p_{\min} \leq p < p_{\max} \\ q_{\max} & \text{pro } p \geq p_{\max} \end{cases} \tag{3.3}$$

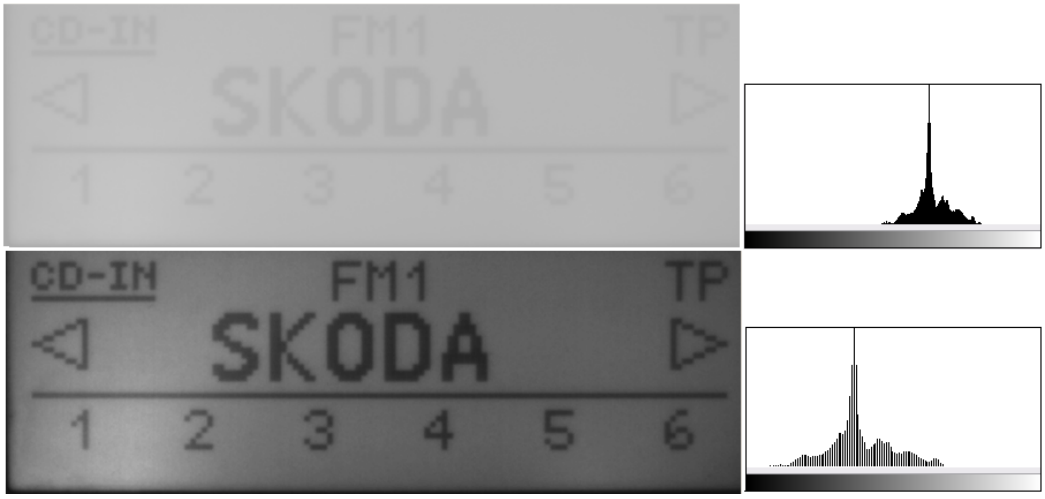
Při volbě $q_{\min} = 20$ a $q_{\max} = 170$ to znamená, že je histogram smrštěn či roztažen do intervalu 150 úrovní jasů, počínaje hodnotou 20. Nenulový offset byl zvolen především z důvodu lepšího vnímání lidským zrakem. Obrázky 3.7 a 3.8 prezentují výsledky normalizace jasu pro tmavý a světlý obrázek spolu s histogramy před a po normalizaci. Vidíme, že histogramy normalizovaných obrázků mají velmi podobný tvar, a to ve stejném intervalu hodnot jasu.



Obrázek 3.7: Příklad jasově normalizovaného snímku – tmavý originál

Normalizovaný obrázek je možno použít pro další zpracování programem TOM Line nebo jinými nástroji pro rozpoznávání znaků. Usnadňuje také proces transformace obrázku do binární podoby.

Je však potřeba si uvědomit, že normalizace nemůže úspěšně zpracovat obrázek sejmutý ve velmi nevhodných světelných podmínkách, jakými jsou tma nebo přímé sluneční paprsky. Obraz sejmutý v extrémních podmínkách nenese již žádnou užitečnou informaci



Obrázek 3.8: Příklad jasově normalizovaného snímku – světlý originál

o snímané scéně. Je tedy na místě zabránit přímému osvícení lesklého displeje rádia sluncem či jiným intenzivním zdrojem světla.

3.8 Prahování

Prahování je další z jasových transformací. Jedná se o proces, na jehož vstupu je šedotónový obraz obsahující všechny odstíny šedi a na výstupu je obraz obsahující pouze bílé a černé body. Takový výstupní obraz nazýváme obrazem *binárním*. Pixely bílé barvy jsou pixely aktivní, proto je, na rozdíl od obvyklého vnímání např. černobílého textu, při operacích s binárním obrazem vždy chápána černá (nulový jas) jako pozadí a bílá (maximální jas) jako popředí.

Nejjednodušší metodou prahování je stanovení konstantního prahu I_{thresh}, tedy mezní hodnoty jasu, kdy obrazové body s jasem I_{xy} nižším budou považovány za pozadí a body s jasem vyšším než mezním pak za popředí, tedy

$$I_{\mathrm{bin}\,xy} = \begin{cases} 0 \text{ pro } I_{xy} \leq I_{thresh}, \\ 255 \text{ jinak} \end{cases} \tag{3.4}$$

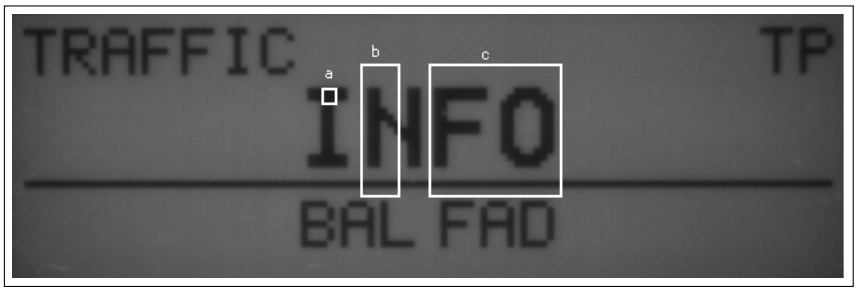
Výsledek takového prahování je velmi závislý na volbě prahu, a správného výsledku, tedy oddělení objektů od pozadí, lze dosáhnout jen při naprosto homogenním osvětlení scény. Protože ideální světelné podmínky obvykle nejsou zajištěny, je potřeba aplikovat složitější metody robustní právě vůči postupným změnám osvětlení.

Jednou z těchto metod je *adaptivní prahování*. Úkolem adaptivního prahování je pro každý pixel v obraze stanovit dynamicky práh, který bude záviset na okolí zpracovaného

pixelu. Myšlenka adaptivního prahování je taková, že změny osvětlení (tedy i odstínu pozadí) jsou plynulé a pomalejší oproti změnám odstínů snímané scény. V dostatečně malém výseku obrázku tedy můžeme barvu pozadí považovat za konstantní.

Prvním úkolem adaptivního prahování je stanovení prahu pro každý pixel. V okolí každého pixelu je vypočten průměrný jas, který je po přičtení případného offsetu považován za prahovou hodnotu. Aby byla získaná hodnota dostatečně reprezentativní, je potřeba, aby bylo pozadí obrazu se ve studovaném okolí zastoupeno více, nežli popředí s odlišným jasem. Tomu je potřeba přizpůsobit rozměry průměrovacího okna.

Budeme-li se blíže zabývat tím, jaké údaje a jakým způsobem může rádio zobrazovat, dospějeme k závěru, že pro dostatečné zastoupení barvy pozadí je potřeba volit okno spíše obdélníkové s delší vertikální stranou, protože rozšiřování okna nemá často na poměr pozadí/objekty výrazný vliv, a přitom větší plocha okna nutně implikuje delší výpočetní čas. Vliv velikosti okna je vidět na obr. 3.9.



Obrázek 3.9: Velikost okna u adaptivního prahování: a) malé, b) vyhovující, c) zbytečně velké

Funkce `cvAdaptiveThreshold` knihovny OpenCV implementuje adaptivní prahování pouze pro čtvercové okno, bylo tedy potřeba funkci ze zdrojových kódů specificky upravit. OpenCV používá k výpočtu prahu pro každý pixel jednu ze dvou volitelných metod – buďto medián jasu pixelů v okně daných rozměrů, nebo gaussovsky vážený průměr v identickém okně. V upravené funkci byla použita metoda s váženým průměrem.

Celý proces prahování od sejmutí obrázku je znázorněn ve třech krocích na obrázku 3.10 a sestává ze sejmutí černobílého snímku, jeho ekvalizaci a adaptivního prahování.

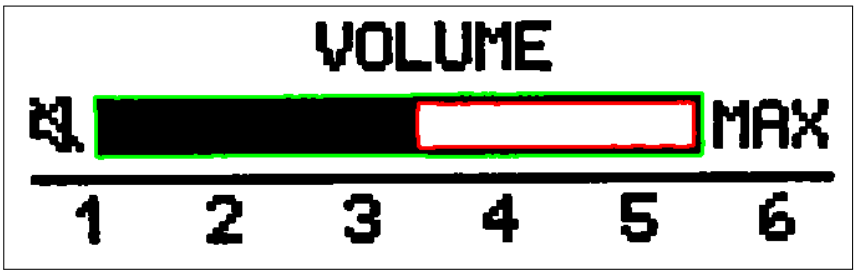
Obrázek 3.10: Prahování: část snímku z kamery, jeho normalizace a prahovaný obraz

3.9 Čtení stavu Progressbaru

Testy často vyžadují nastavení konkrétní hodnoty hlasitosti, či jiného numerického parametru. Hodnota nastavované veličiny je někdy indikována číslem, jindy pouze grafickým ukazatelem, tzv. *progressbarem*. Nyní si ukažme, jak lze snadno určit nastavenou hodnotu ze snímku progressbaru.

Nejprve je potřeba výše uvedeným způsobem získat binární obraz displeje. V binárním obraze můžeme nalézt všechny spojité komponenty. Spojitou komponentou rozumíme každou maximální podmnožinu bodů nesoucích barvu popředí takovou, že každý bod přímo sousedí s nějakým dalším bodem této množiny; hledání spojitých komponent je algoritmicky poměrně jednoduché [1]. Obdélník ohraničující komponentu ukazatele bude pravděpodobně vždy (tzn. nehledě na indikované hodnotě) stejně velký, protože komponenta musí graficky trvale zvýrazňovat maximální možnou hodnotu. Nalezené spojité komponenty můžeme tedy roztřídit podle velikosti – hledáme obdélník, jehož délka l_{bar} i výška leží v určitém intervalu hodnot.

Pokud nalezneme takovou spojitou komponentu, můžeme ji dále analyzovat. Funkce `cvFindContours` z knihovny OpenCV umožňuje nalézt kromě spojitých komponent i otvory uvnitř každé komponenty. Všechny tyto vyhledané objekty můžeme opět filtrovat a vyloučit případné nezajímavé otvory, které vznikly nečistotou na displeji, odrazem světla nebo chybou prahování. Zajímá nás objekt, jehož pravý okraj leží v blízkosti okraje vlastní spojité komponenty a jehož výška je větší alespoň než jedna polovina výšky nalezeného ukazatele. Obrysy těchto komponent jsou zvýrazněny na obrázku 3.11.



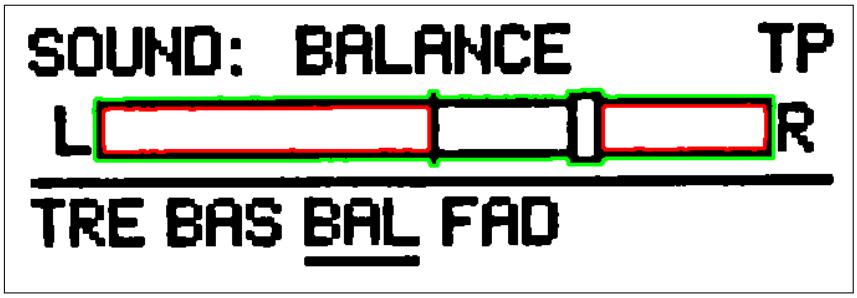
Obrázek 3.11: Progressbar – nalezená spojitá komponenta a otvor uvnitř

Z poměru šířek takto nalezených komponent můžeme určit procentuální výplň ukazatele. Známe-li navíc počet kroků zobrazované hodnoty v_{max}, je tato rovna

$$v = v_{max} * \left(1 - \frac{l_{hole}}{l_{bar}}\right), \tag{3.5}$$

kde šířka celého ukazatele l_{bar} byla zmenšena o šířku obrysové čáry.

Podobný přístup volíme při získávání hodnoty symetrického ukazatele s nulou uprostřed (obr. 3.12). Zde je potřeba nalézt dva otvory, dotýkající se levé a pravé stěny celého ukazatele. V dalším kroku vybereme ten, který nese informaci, tedy otvor s menší šířkou. Obdobným výpočtem jako (3.5) opět dostaneme hodnotu tímto ukazatelem zobrazenou. Zda je zobrazována hodnota kladná či záporná potom určíme z umístění menšího z obou otvorů, menší levý značí zápornou hodnotu.



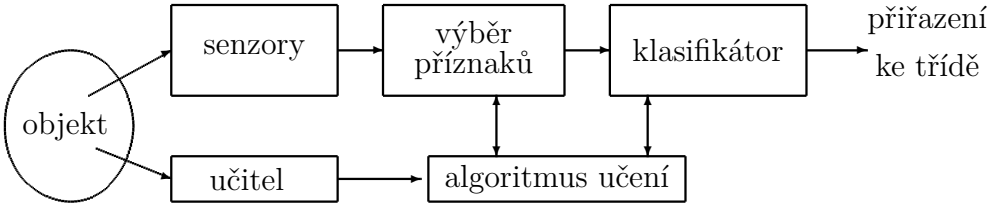
Obrázek 3.12: Komponenty symetrického ukazatele

Výhodou přístupu pomocí hledání spojitých komponent oproti počítání pixelů uvedeném v části 3.4 je nezávislost na přesné znalosti polohy ukazatele ani na jeho přesně známé délce, nevýhodou pak vyšší výpočetní složitost.

3.10 Rozpoznávání znaků a symbolů

Úkolem rozpoznávání (Pattern Recognition) je zařadit pozorování – v případě rozpoznávání znaků jsou pozorováním obrázky jednotlivých písmen – do předem známých tříd – těmi jsou jednotlivá písmena abecedy – podle nějakého rozhodovacího pravidla. Rozhodovací pravidlo (klasifikátor) nemusí na rozdíl od matematického modelování vystihovat složitou podstatu objektu, o němž rozhoduje; rozhodovací pravidlo je naučeno empiricky z mnoha pozorovaných příkladů. Metody učení rozdělujeme na *učení s učitelem* a *učení bez učitele* [15]. Komponenty rozpoznávacího systému jsou schématicky znázorněny na obr. 3.13. Statistické (příznakové) rozpoznávání reprezentuje objekty jako body vektoro-

vého prostoru, kde souřadné osy odpovídají číselně vyjádřeným pozorováním. Číselné vyjádření můžeme v případě obrazových dat spočíst například jako invariantní momenty [8].



Obrázek 3.13: Učení a klasifikace do tříd

Pro řešení úlohy optického rozpoznávání znaků byly vybrány dva volně dostupné nástroje Tesseract a GOCR.

3.10.1 Tesseract

Tesseract OCR je *open source* nástroj pro rozpoznávání znaků. Systém byl původně vyvinut v laboratořích HP, později byl uvolněn pro neomezené použití.

Náplní práce není detailní rozbor systému OCR, uveďme pouze, že Tesseract je vybaven klasifikačním souborem pro rozpoznávání znaků, není tedy nutné provádět učení písma. Klasifikační soubor byl vytvořen na základě naskenovaných textů a je možno jej rozšířit pro zvýšení přesnosti rozpoznávání netypických řezů písma. Systém dále disponuje obsáhlou databází slov, která zvyšuje úspěšnost čtení.

Tesseract je možno připojit do vlastního projektu v podobě dynamické knihovny DLL. Podmínkou je nastavení cesty k adresáři s klasifikačními soubory do proměnné prostředí `TESSDATA_PREFIX`.

3.10.2 GOCR

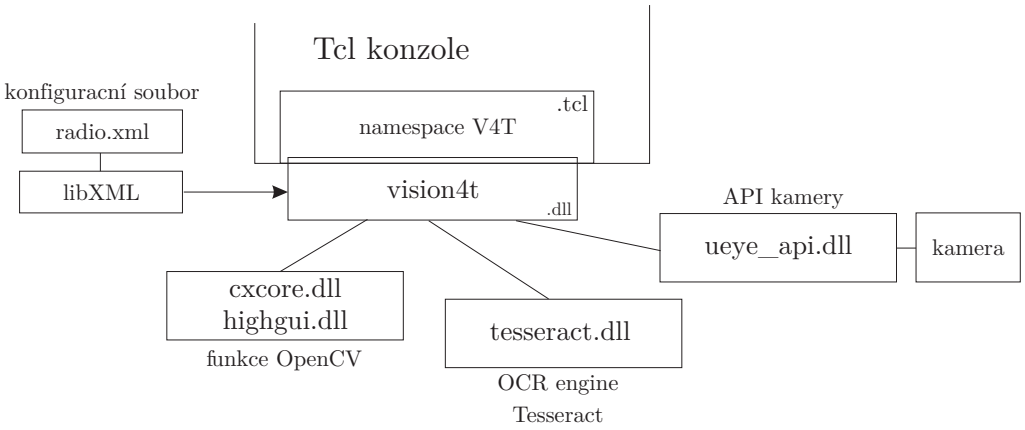
Program GOCR, vyvíjený pod GNU Public licencí, slouží rovněž pro rozpoznávání tištěného textu. Navíc disponuje možností jednoduchého naučení symbolů, kdy je vytvořena databáze symbolů, obsahující vždy jednu instanci naučeného symbolu. Vzorové rádio zobrazuje pouze jediný symbol, kterým je označení vypnutého zvuku (mute). Program GOCR je pro demonstraci schopností využit právě pro rozhodnutí, zda je symbol mute zobrazen.

Narozdíl od Tesseract je GOCR k dispozici pouze jako spustitelný program, nelze jej tedy připojit jak dynamickou knihovnu.

3.11 Interface pro TestAut

Celá práce by postrádala smysl, kdyby neposkytovala propojení nové funkcionality se stávajícím automatizovaným testerem. Jak bylo uvedeno v popisu testeru, k rozšíření jeho funkcionality slouží skriptovací jazyk Tcl a zásuvné moduly v podobě DLL knihoven, které musí být kompatibilní s použitým interpreterem Tcl.

Základem bylo tedy vytvořit v jazyce C++ rozšiřující balíček `vision4t`. Kompatabilitu s Tcl zajišťuje hlavičkový soubor `tcl.h`, distribuovaný spolu s Tcl. Vytvořený balíček dále sám používá dynamické knihovny balíku s algoritmy zpracování obrazu OpenCV, knihovnu se systémem OCR Tesseract a v neposlední řadě API kamery uEye. Schéma propojení jednotlivých funkčních celků reprezentuje obr. 3.14. V jazyce Tcl byl založen jmenný prostor `V4T`, který zaobaluje veškerou funkcionalitu rozšiřujícího balíčku `vision4t`. Dynamická knihovna `vision4t` obsahuje kód zajišťující získání snímku z kamery, předzpracování obrazu a jeho analýzu nebo zpracování pomocí OCR.



Obrázek 3.14: Schéma implementace navrženého rozšíření

Po sejmutí snímku kamerou je získaný obraz oříznut na velikost displeje. Aby bylo možno jednoduše spravovat souřadnice displeje a jeho dalších podoblastí, je potřeba pro každé rádio vytvořit kalibrační soubor s odpovídajícími souřadnicemi. Pro vyčítání dat z tohoto konfiguračního XML souboru slouží knihovna `libXML` [14].

Implementované rozšíření systému je navrženo tak, aby bylo jeho použití z pohledu vytváření testů co nejjednodušší. Jako příklad uveďme zjištění hodnoty zobrazené progressbarem. Aby mohla být hodnota určena, musí být nejprve sejmut obrázek z kamery, z něj vyříznuta oblast s displejem a ta předzpracována jasovou transformací. Následuje proces adaptivního prahování a segmentace výsledného binárního obrazu, filtrování spo-

jitých komponent a teprve nakonec analýza a výpočet zobrazené hodnoty. Tento řetězec operací je pro uživatele skryt pod jediným voláním funkce `V4T::GetPBOneside`, jejímž jediným parametrem je maximální hodnota, kterou může daný ukazatel zobrazit.

Kapitola 4

Experimenty

Provedené experimenty můžeme rozdělit do několika částí. Nejprve předvedeme výsledky úprav ovládacího programu robota. Dále uvedeme porovnání systémů pro rozpoznávání obrazu a nakonec ověříme funkci navrženého řešení přímo v testech radií.

4.1 Pohyby robota

Jako měřítko k porovnání upraveného řídicího programu robota se programem původním byl zvolen čas k provedení určitých rozsáhlejších úkonů.

Prvním experimentem bylo nastavení bezpečnostního kódu rádia na hodnotu 9775. Nastavení probíhá pomocí čtyř virtuálních kláves pod displejem, z nichž každá ovládá jeden řád. Jedním stiskem tlačítka zvýšíme hodnotu korespondující cifry o 1. Nastavovací proces tedy vyžaduje 28 stisků tlačítka a čtyři přesuny nad tlačítko. Tabulka 4.1 shrnuje výsledek experimentu, který ukázal na 50% zrychlení celé operace.

původní program	41"
nový program	20"
časová úspora	52%

Tabulka 4.1: Zrychlení programu: nastavení bezpečnostního kódu 9775

Jako další experiment byla zvolena v předchozím textu popisovaná funkce `Center`, která slouží k vystředění hodnot *bass, treble, fader* a *balance*. Porovnány jsou časy vystředění původním programem, novým programem bez optické zpětné vazby, dále s kamerou za předpokladu, že již byly všechny hodnoty nastaveny na požadovanou hodnotu 0 a nakonec s kamerou v nejhorším případě, kdy každá ze 4 voleb byla nastavena na

mezní hodnotu 9 nebo −9. Funkce `Center` je volána na začátku každého testu, proto je její zrychlení velmi důležité z hlediska celkového času testu. Ve většině případů jsou všechny čtyři parametry nastaveny na nulovou hodnotu již před započetím testu, tedy stačí nastavení pouze zkontrolovat. Dostáváme nejlepší čas uvedený v tabulce 4.2, který přináší urychlení o více než jednu minutu oproti původnímu řešení.

Jakékoli zkrácení manipulačního času prodlužuje životnost jak vlastního manipulátoru, tak kamery, která se rovněž pohybuje a na kterou může negativně působit každé zrychlení, respektive změna pohybu.

Program	čas (min:sec)
původní program	1:44
nový program bez kamery	1:13
s kamerou nejlepší	0:40
s kamerou nejhorší	1:24

Tabulka 4.2: Čas vykonání funkce `Center`

4.2 OCR

K porovnání nástrojů pro rozpoznávání znaků byl použit předváděcí test rozpoznání přehrávané skladby a několik dalších RDS textů. Výsledky tohoto experimentu vedly k použití jiných rozpoznávacích software, nežli doporučený TOM Line.

Zobrazený text	TOM Line	Tesseract	gOCR
TRACK 01	TRAK 0l	TRACK 01	TRACK 01
TRACK 02	TRAK02	TRACK 02	TRACK 02
TRACK 03	TRACE 03	TRACK 03	TRACK 03
TRACK 04	TRAK O4	TRACK 04	TRACK 04
TRACK 18	TRAK 18	TRACK 18	TRACK 18
RADIO	–	RADIO	RADIO
TESTER	–	TESTER	TESTER
SKODA	–	SKODA	SKODA

Tabulka 4.3: Texty rozpoznané pomocí různých OCR

4.3 Analýza progressbaru

Správná funkce rozpoznávání grafického ukazatele progressbar byla ověřena na testovacím rádiu ULOWSK25 opětovným nastavením hodnot z celé škály ukazatele a verifikací, zda není přečtená hodnota chybná.

Ověření bylo provedeno na třech různých ukazatelích

- Hlasitost (Volume) je zobrazována lineárním ukazatelem hodnot 0 až 30,
- pro basy (Bass) je použit symetrický ukazatel s nulou uprostřed a rozsahem zobrazovaných hodnot -9 až 9 a
- Intenzita podsvícení nabývá hodnot 0 až 3

Všechny hodnoty získané pomocí funkce pro analýzu ukazatele progressbar přesně odpovídaly zobrazeným údajům.

Kapitola 5

Závěr

Práce uvedla stručný popis původního systému pro automatizované testování autorádií. Čtenář byl seznámen se základy poměrně atypických programovacích jazyků pro ovládání průmyslového robota a pro vytváření testů tak, aby mohl jednoduše porozumět vytvořeným zdrojovým kódům orientovat se v nich.

Po rozboru existujících testů byly navrženy metody využití kamery pro automatizované testování. Úkolem kamery je pouze podpora a vylepšení stávajících testů; optický vstup do testovacího systému slouží k odstranění určitých jinak neodhalitelných chyb a k urychlení testovacího procesu. Kamera neslouží jako základní článek, na kterém by testování nezbytně záviselo.

Následně byl čtenář seznámen s programovým balíkem TOM Line pro řešení úloh optického rozpoznávání. Na příkladu funkcí TOM Line bylo předvedeno, jak je možno navržené úlohy řešit. Od skutečného použití tohoto software však bylo následně ustoupeno, protože se jeho spolehlivost ukázala jako nedostačující pro požadovanou aplikaci. Na místo tohoto programového balíku bylo vytvořeno vlastní specializované řešení na základě využití knihoven pro zpracování obrazu OpenCV a nástrojů pro rozpoznávání textu. S tím souvisí navržený způsob propojení kamery, programu pro zpracování obrazu a nástrojů pro rozpoznávání textu.

Vytvořený balíček pro zpracování a analýzu obrazu z kamery byl zakomponován do stávajícího testeru. Byly upraveny některé pomocné funkce testů tak, aby rozšíření systému produktivně využily. Další úpravy zohledňující použití kamery byly rovněž provedeny v řídicím programu průmyslového robota. Pomocí testů s rádiem ULOWSK25 byl nakonec demonstrován kladný přínos práce, spočívající v urychlení a zkvalitnění automatizovaných testů.

Literatura

[1] H.H. Atkinson, T.R. Walsh, and I. Gargantini. Counting regions, holes and their nesting level in time proportional to the border. *Computer Vision, Graphics, and Image Processing*, 29:196–215, 1985.

[2] G. Bradski and A. Kaehler. *Learning OpenCV*. O'Reilly Media, September 2008.

[3] V. Hlaváč and M. Sedláček. *Zpracování signálů a obrazů, skriptum FEL*. Vydavatelství ČVUT, 2005.

[4] IDS Imaging Development Systems, Oberslum. *User Manual – uEye Software Development Kit*, 2007.

[5] IDS Imaging Development Systems, Oberslum. *User Manual – uEYE Cameras*, 3.30 edition, 2009. [online].

[6] Intel. *Open Source Computer Vision Library, reference manual*, 2001.

[7] Frank Iwanitz and Jürgen Lange. *OPC - Fundamentals, Implementation, and Application*. Huthig Pub Ltd, January 2002.

[8] X.Y. Jiang and H. Bunke. Simple and fast computation of moments. *Pattern Recognition*, 24:801–806, 1991.

[9] D. Krčmář. *Programujeme v COM a COM+*. Computer Press, říjen 2000.

[10] KUKA Roboter GmbH. *KR C2/C3 Expert Programming*, 5.2 edition, September 2003. [CD-Disk].

[11] KUKA Roboter GmbH. *KR C2/C3 Configuration*, 5.2 edition, August 2005. [CD-Disk].

[12] D. Novotný. Jemný úvod do LISPu. [online], duben 2004. {http://www.root.cz/clanky/jemny-uvod-do-lispu/}.

[13] M. Šonka, V. Hlaváč, and R.D. Boyle. *Image processing, analysis, and machine vision*. PWS, Boston, 1999.

[14] M. Riehl and I. Sterin. *XML and Perl*. Sams Publishing, 2002.

[15] M.I. Schlesinger and V. Hlaváč. *Ten lectures on statistical and syntactic pattern recognition*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 2002.

[16] P. Tišnovský. Programovací jazyk TCL. ⟨http://www.root.cz/clanky/programovaci-jazyk-tcl⟩ [cit. 10.2009], červenec 2005.

[17] O.D. Trier, A.K. Jain, and T. Taxt. Feature extraction methods for character recognition – a survey. *Pattern Recognition*, 29(4):641–662, 1996.

[18] Wikipedia. Barevný model HSV. [online]. ⟨http://cs.wikipedia.org/wiki/HSV⟩ [cit. 11.2009].

[19] Wikipedia. Sběrnice GPIB. [online]. ⟨http://cs.wikipedia.org/wiki/GPIB⟩ [cit. 10.2009].

Seznam zkratek

API APplication Interface, aplikační rozhraní

COM Component Object Model, komponentové programování pro Windows

DLL Dynamic Link Library, dynamická knihovna

IPL Intel Image Processing Library

IPP Intel Integrated Performance Primitives

OCR Optical Character Recognition

OLE Object Linking and Embedding

OPC OLE for Process Control

LAN Local Area Network

GPIB General Purpose Interface Bus

HSV Hue–Saturation–Value, barevný model

RDS Radio Data System

TA Traffic Announcement, dopravní zpravodajství

TCL Tool Command Language, skriptovací jazyk

TP Traffic Program, rádiová stanice vysílající TA

XML eXtensible Markup Language, značkovací jazyk

Příloha A

Knihovna OpenCV

A.1 Struktura IPLImage

```
typedef struct _IplImage
    {
        int  nSize;         /* sizeof(IplImage) */
        int  ID;            /* version (=0)*/
        int  nChannels;     /* Most of OpenCV functions support 1,2,3 or 4 channels */
        int  alphaChannel;  /* ignored by OpenCV */
        int  depth;         /* pixel depth in bits*/
        char colorModel[4]; /* ignored by OpenCV */
        char channelSeq[4]; /* ditto */
        int  dataOrder;     /* 0 - interleaved color channels, 1 - separate color channels.*/
        int  origin;        /* 0 - top-left origin,
                               1 - bottom-left origin (Windows bitmaps style) */
        int  align;         /* Alignment of image rows (4 or 8).
                               OpenCV ignores it and uses widthStep instead */
        int  width;         /* image width in pixels */
        int  height;        /* image height in pixels */
        struct _IplROI *roi;/* image ROI. when it is not NULL, this specifies image region to process */
        struct _IplImage *maskROI; /* must be NULL in OpenCV */
        void  *imageId;     /* ditto */
        struct _IplTileInfo *tileInfo; /* ditto */
        int  imageSize;     /* image data size in bytes
                               (=image->height*image->widthStep in case of interleaved data)*/
        char *imageData;  /* pointer to aligned image data */
        int  widthStep;   /* size of aligned image row in bytes */
        int  BorderMode[4]; /* border completion mode, ignored by OpenCV */
        int  BorderConst[4]; /* ditto */
        char *imageDataOrigin; /* pointer to a very origin of image data
                                  (not necessarily aligned) -
                                  it is needed for correct image deallocation */
    }
    IplImage;
```

A.2 Získání barevného obrazu z kamery pro OpenCV

```
m_hCam = (HIDS) 0;              // open next camera
m_Ret = is_InitCamera( &m_hCam, NULL );    // init camera - no window handle required

if( m_Ret == IS_SUCCESS )
{
  // retrieve original image size
  SENSORINFO sInfo;
  is_GetSensorInfo( m_hCam, &sInfo );
  m_nSizeX = sInfo.nMaxWidth;
  m_nSizeY = sInfo.nMaxHeight;

  // setup the color depth for color image
  m_nBitsPerPixel = 32; //8
  m_nColorMode = IS_SET_CM_RGB32;// IS_SET_CM_Y8;
  is_SetColorMode(m_hCam, m_nColorMode);

  // memory initialization
  is_AllocImageMem(m_hCam, m_nSizeX, m_nSizeY, m_nBitsPerPixel, &m_pcImageMemory, &m_lMemoryId);
  is_SetImageMem( m_hCam, m_pcImageMemory, m_lMemoryId );  // set memory active

  is_SetExternalTrigger(m_hCam,IS_SET_TRIG_SOFTWARE);

  if( is_FreezeVideo( m_hCam, IS_WAIT ) != IS_SUCCESS )
  {
    //create image
    IplImage* img;
    img = cvCreateImage(cvSize(m_nSizeX,m_nSizeY),IPL_DEPTH_32S,4);
    //and copy data into new IPLImage
    is_CopyImageMem(m_hCam,m_pcImageMemory,m_lMemoryId,img->imageData);
  }
}
```

Příloha B

Testy

B.1 Funkce pro odkódování rádia

Níže uvedená funkce slouží pro nastavení bezpečnostního kódu na konkrétním modelu rádia. Vstupním parametrem je bezpečnostní kód. Po odpojení rádia od napájení a jeho opětovném zapnutí je pomocí kamery nejprve zjištěna inicializační hodnota, tedy číslo zobrazené na displeji. Pomocná funkce `CalcDiff` vrátí „přírůstek“ kódu, který je potřeba pomocí tlačítek na rádiu nastavit. Poté je kamerou opět zkontrolován zobrazený údaj. V případě chyby proběhne maximálně 4 krát pokus o opravu.

```
proc Coding { number } {
  BLACKBOX::RadioPowerSrc 1
  scan $number "%d" number    ;# parse number (leading zeros problem)
  after 1000
  BLACKBOX::RadioPowerSrc 0
  ROBOT::Move "TURN_ON"
  ROBOT::Viewdisplay 1
  ROBOT::WaitForMotionEnd
  after 4000 ;#wait for radio init
  set init [V4T::ReadNum]
  set failed -1
  while { ($failed < 4) && ($init != $number) } {
    # calc code difference
    set dif [CalcDiff $number $init]
    SetCodeRel $dif      ;# try to set code on radio
    ROBOT::Viewdisplay 1 ;# verify the result
    ROBOT::WaitForMotionEnd
    after 100
    set init [V4T::ReadNum] ;# read number from display
    incr failed
  }
  if { $failed >= 4 } { return 0 } ;#unable to set code
  after 500
```

```
  ROBOT::Move "VK6_SHORT" ;#press "OK" button
  ROBOT::WaitForMotionEnd
  return 1
}
```

B.2 Hiss-noise – příklad z normy

	nach Abschnitt 2.2.1 $E' = E'_R + 3$ dB
Modulation:	1000 Hz, m = 30 %
Frequenzversatz:	(± 10 bzw. ± 20) KHz
NF-Störabstand:	= 20 dB

Messung:

Zusammenschaltung von f_N und f_S über Kombinationsnetzwerk „B"

- HF-Pegel f_S zunächst aus, Empfänger auf f_N abstimmen.
- f_S-Pegel auf den Grenzwert erhöhen und Nutzsendermodulation abschalten,
- Störsenderpegel so lange erhöhen, bis der NF-Störabstand = 20 dB beträgt.

Der HF-Pegel des Störsenders gilt als Maß.

4.4.5 Intermodulation (2 Sender)

Grenzwerte der HF-Eingangspegel:	$E'_{1St} = E'_{2St} \geq 74$ dBµV
Empfängerabstimmung:	f_N = 720 KHz/ 1080 KHz
Störsender 1:	f_{S1} = 840 KHz
Störsender 2:	f_{S2} = 960 KHz
Modulation:	M = 30 % / 1 KHz

Messverhalten:

Zusammenschaltung von f_{S1} und f_{S2} über Kombinationsnetzwerk „B"

- f_{S1} auf f_N einstellen und Radio auf f_N abstimmen, P_{NF} = 0,5 W bei voller Begrenzung einstellen
- f_{S1} und f_{S2} auf Störfrequenzen einstellen
- E'_{1St} und E'_{2St} mit gleichen Pegel so lange erhöhen, bis das Intermodulationsprodukt hörbar ist, (1 kHz-Ton schwebend)
- Modulation f_{S2} ausschalten und f_{S1} nachstimmen bis P_{NF} = max.
- $E'_{1St} = E'_{2St}$ mit gleichen Pegeln so lange einstellen, dass der Intermodulationston 20 dB unter P_{NF} liegt (0,14V / 4 Ω), 1 kHz-Filter benutzen! (DK: UPL Filter? Besser wäre eine allgemeine Anforderung an die Audio Meßtechnik!)

4.4.6 Hiss-Noise (THD über NF)

Grenzwerte:	k ≤ 0,5 %
HF-Nutzpegel	E´ = 60 bis 100 dBµV
Modulationsgrad:	m = 30 bis 80 %
Modulationsfrequenz:	100 Hz bis 2000 Hz
Gemessen mit Tiefpass	10 kHz

Entwurf