SLOVENSKÁ TECHNICKÁ UNIVERZITA, FAKULTA ELEKTROTECHNIKY A INFORMATIKY

VIZS – úloha 1

Návrh optickej odometrie pomocou optického toku

Adam Sojka 72515, Filip Štec 72520
9.4.2017

Úloha

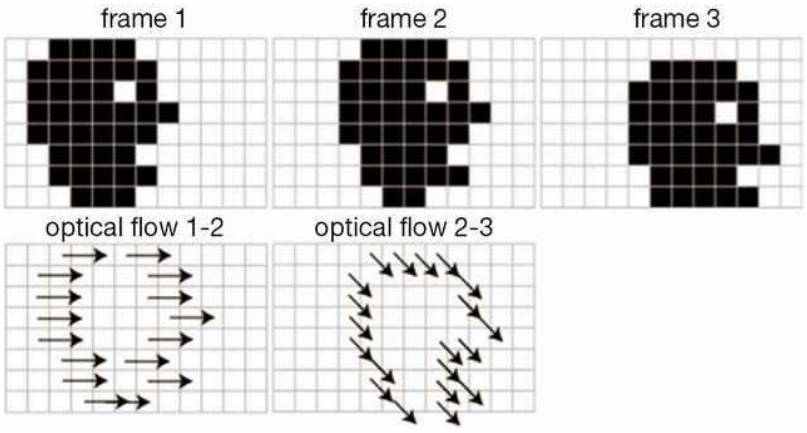
Navrhnite a naprogramujte program (aplikáciu), ktorá bude schopná využitím optického toku určiť vzdialenosť, o ktorú sa posunie kamera (optická odometria). Posunutie kamery sa pokúste určiť na každej z osí *(x, y, z)*, ak je to možné.
Počas vypracovávania zadania budete prechádzať cez jednotlivé metódy spracovania obrazu, ktoré na seba nadväzujú (konverzie medzi farebnými modelmi, hranové operácie a iné filtrácie obrazu, segmentácia, detekcia objektov, ...). Pre každú túto časť vypracujte analýzu možných riešení a odôvodnite výber Vášho postupu (napr.: prečo ste použili daný farebný model a nie iný, ...).

***Použiť môžete**: knižnicu OpenCV (alebo iné voľne dostupné knižnice), internetové zdroje, skriptá a učebnice.*
***Programovacie prostredia**: Microsoft Visual Studio, Android Studio, Matlab,...*

Teoretický postup

Optický tok

Pod pojmom optický tok rozumieme vizuálny pohyb predmetov na obrázkoch zachytených práve za sebou pomocou kamery. Pohyb pritom môže byť vyvolaný pohybom objektov vo videu alebo aj samotným pohybom kamery. Tento pohyb je zaznamenaný pomocou vektorov, ktoré spájajú určitý bod z predchádzajúceho záberu s tým istým bodom v aktuálnom zábere.

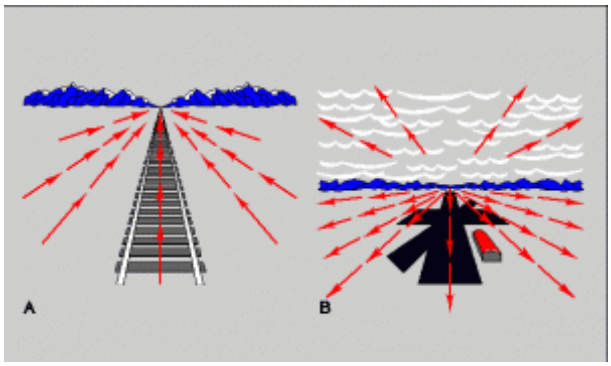


Obrázok 1 Prinícp určenie optického toku

Výpočet optického toku po jednotlivých pixeloch nie je možný, lebo sa dostávame do jednej rovnice s dvomi neznámymi, preto na výpočet použijeme Lucas-Kanade metódu. Táto metóda pozostáva z predpokladu, že susedné body sa pohybujú podobne ako vybratý, tým dostávame preurčenú sústavu rovníc. Metóda rieši aj problém s malými a veľkými pohybmi, lebo používa pyramídu. Pohybom hore po pyramíde sa odstraňujú malé pohyby a veľké pohyby sa stávajú malými, čím dostávame aj mierku.

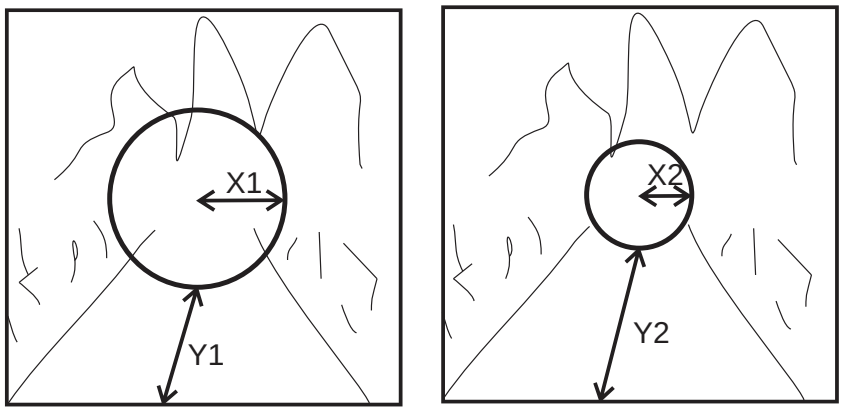
Vizuálna odometria

Najprv sa budeme zaoberať jednoduchšou úlohou, a to rozpoznanie smeru pohybu v osiach x, y a z. Pre os x, horizontálny pohyb s kamerou do strany a os z, vertikálny pohyb, je identifikácia smeru pohybu jednoduchá, keďže vektory sa pohybujú vždy iba jedným smerom, horizontálne alebo vertikálne. Pre pohyb v smere osi y, to je pohyb s kamerou dopredu a dozadu, je optický tok zobrazený na nasledujúcom obrázku. Ako vidno, pri pohybe dopredu vektory optického toku smerujú akoby z fotky a zas pri cúvaní idú do fotky. To by sa malo dať zistiť porovnávaním uhla natočenia vektora s jeho absolútnou pozíciou na fotke, keďže vieme predpokladať, že napr. v pravom dolnom rohu fotky pri pohybe dopredu by mal vektor mať uhol približne 45 stupňov a smerovať vpravo dole.



Obrázok 2 Princíp určenia pohybu v smere osi Y

Iným spôsobom určenia pohybu kamery v osi y je pomocou veľkosti detegovaného objektu na obrázku. Je potrebné najprv určiť závislosť medzi vzdialenosťou a veľkosťou objektu z dvoch obrázkov, kde vieme v akej vzdialenosti sa objekt nachádza od kamery a poznáme jeho skutočnú veľkosť aj veľkosť na obrázku.



Obrázok 3 Polomer kruhu sa so zväčšujúcou sa vzdialenosťou zmenšuje

Podľa určenej závislosti a zmeranej veľkosti objektu v pixeloch potom vieme určiť skutočnú vzdialenosť objektu. Závislosť je:

$$Y = k * x + q$$

Kde Y je reálna vzdialenosť v centimetroch, k je smernica priamky, q je posunutie priamky a x je veľkosť pixelu v centimetroch.

$$x = \frac{X_{cm}}{X}$$

Kde X_{cm} je veľkosť objektu v centimetroch a X je veľkosť objektu v pixeloch. Z uvedených vzťahov vyplýva, že vzdialenosť objektu od kamery závisí nepriamoúmerne od veľkosti objektu. Riešením sústavy rovníc o dvoch neznámych získame koeficienty rovnice priamky k a q.

Programové vybavenie

Ku zdrojovým kódom ako aj tejto dokumentácii je možné sa dostať na umiestnení:
https://github.com/Smadas/VIZSul1

Všetky zdrojové kódy sú písané v C++ s použitím openCv 3.1, aj staršie verzie by mali byť kompatibilné.

Programové vybavenie sa skladá z nasledovných knižníc:

1. Main – spúšťanie programu a všetkých podprogramov, hlavná slučka, ktorá prebieha všetky obrazy
2. captureVideo – záznam obrázkov z kamery pripojenej k počítaču a ich uloženie na disk
3. readImagesFromDirectory – načítanie všetkých obrázkov z umiestnenia na disku do pamäti RAM
4. DetectTarget – detekcia objektu záujmu
5. opticalFlowLucasKanade – výpočet optického toku
6. detectMotionXZ – detekcia pohybu kamery v osi X a Z
7. detectMotionY – detekcia pohybu kamery v osi Y
8. displayData – grafické a číselné zobrazenie pohybu kamery

1. Funkcia main

V tejto funkcii sa vykonáva chod celého programu a skladá sa z nasledujúcich častí

1. Načítanie obrázkov pohybu z priečinka, parameter invert otáča poradie načítaných obrázkov
2. Nájdenie vhodných bodov na sledovanom objekte
3. Kalibrácia parametrov pre pohyb Y pomocou kalibračných obrázkov v priečinku calibY
4. Postupné načítavanie obrázkov po jednom a ich vyhodnocovanie
   4.1. Výpočet optického toku
   4.2. Vykreslenie vypočítaných bodov optickým tokom do obrázka
   4.3. Výpočet pohybu v smere Y, ak spĺňa určenú hranicu veľkosti pohybu, je vypísaný pohyb Y do obrázka
   4.4. Ak pohyb Y je malý (pod určenou hranicou), vyráta sa pohyb XZ a vykreslí sa do obrázka
5. Zobrazenie vyhodnoteného obrázka na krátky čas a pokračovanie na ďalší obrázok
6. Zastavenie programu pri zobrazenom poslednom obrázku

2. Záznam obrázkov z kamery

Obsahuje funkcie saveFrame(uloženie obrázka) a captureVideoAsImages(záznam videa ako obrázkov).

saveFrame – uloží cv maticu do bitovej mapy na disk do umiestnenia captureVid, ktoré je umiestnené pri spustiteľnom súbore.

captureVideoAsImages – zobrazuje obrázky z kamery do okna a spúšťa funkciu saveFrame každých 100ms.

3. Načítanie obrázkov do pamäte RAM

Obsahuje tri funkcie:

- getImageFileNames – načítanie názvov všetkých bitových máp v danom umiestnení, vstupom je umiestnenie súborov a výstupom je vektor reťazcov názvov.
- readImgFiles – načítanie bitových máp zo súborov do pamäte RAM, vstupom je umiestnenie obrázkov a výstupom je vektor cv matíc načítaných obrázkov.
- showImagesFromDirectory – demo zobrazenie obrázkov z daného umiestnenia – je potrebné zabezpečiť dáta do umiestnenia captureVidX.

4. Detekcia objektu záujmu

Slúži na nájdenie objektu v obrázku, podľa ktorého určujeme odometriu.

- detect – funkcia sa skladá z troch častí, najprv nájdeme kružnicu opisujúcu náš objekt, následne nájdeme určitý počet bodov na vylepšenej kružnici a nakoniec si zistíme súradnice stredu a polomer vylepšenej kružnice
- findCircle – nájdeme kružnice v obrázku pomocou funkcie HoughCircles a následne odfiltrujeme neplatné kružnice pomocou farebnej škály
- findPoints – funkcia určí koordináty bodov na kružnici pomocou rovnomerného rozloženia, body sú vylepšené funkciou betterPoints
- betterPoints – zoberie si súradnice bodu a uhol k stredu na zväčšenej kružnici, posúva bod po malých krokoch smerom ku stredu, dokým nebude splnená podmienka farebnej škály cieľového objektu
- calcCircleCenter – vyráta nový stred kružnice pomocou troch rovnomerne rozložených bodov

Na hľadanie objektu sme použili funkciu HoughCircles, lebo sme si na začiatku zvolili náš objekt záujmu kruhového tvaru, vďaka ktorému je výrazne zjednodušené detegovanie objektu. Pred vložením do funkcie treba záber mierne upraviť. To znamená premeniť farebnú škálu z BGR na GRAY, lebo sa tak zvýraznia kontúry a tiež použiť funkciu na rozostrenie obrázku, v našom prípade MedianBlur, ktorá vyhladí hrany a tak zlepší detekciu. Keďže funkcia HoughCircles môže mať na výstupe väčšie množstvo nájdených kružníc, bolo potrebné vyfiltrovať iba cieľ záujmu, čo sme dosiahli porovnaním farby stredového bodu kružnice s rozsahom farby pre objekt. Pri pokusoch mal výstup funkcie značnú odchýlku v detekcii vrámci polomeru detegovanej kružnice, preto sme pristúpili k doladeniu hľadaných bodov pomocou porovnávania bodov s farebným intervalom cieľa, tým sme docielili nájdenie určitého počtu bodov presne na hrane objektu. Pomocou týchto bodov sa ďalej rátal nový stred a polomer kružnice.

5. Výpočet optického toku

Obsahuje dve funkcie:

- computeOpticalFlow – vypočíta optický tok dvoch zasebou idúcich obrázkov – vstupom je šedotónový obrázok a predošlý obrázok, body objektu a predošlé body objektu. Pri prvom spustení je potrebné inicializovať body individuálnou detekciou objektu.
- showOpticalFlow – demonštrácia použitia funkcie computeOpticalFlow – je potrebné zabezpečiť dáta do umiestnenia captureVidX.

Táto knižnica používa metódu Lucas-Kanade, ktorá počíta posunutie bodov objektu záujmu, ktoré boli vopred určené. Rozhodli sme sa použiť túto metódu, pretože je ju jednoduchšie integrovať to optickej odometrie. Pri iných metódach, kde sa počítajú posunutia po celom obraze by bolo potrebné zložito rozhodovať o tom, ktoré vektory sú relevantné.

Pri určovaní optického toku je použitý šedotónový farebný model kvôli princípu fungovania openCV funkcie na výpočet optického toku.

6. Detekcia pohybu v osi XZ

- vectorParam – funkcia vracia uhol a veľkosť vektora daného polárnymi súradnicami
- getObjectMotionXZ – pomocou priemerovania vektorov optického toku všetkých zisťovaných bodov rátame celkový vektor pohybu v smeroch X a Z a prerátavame na [cm]

Hodnota konštanty CALIB_XY bola zisťovaná porovnávaním polomeru kružnice detegovaného objektu v danej vzdialenosti experimentu s reálnym polomerom.

7. Detekcia pohybu v osi Y

Obsahuje sedem funkcií pre dosiahnutie cieľa zistenia pohybu kamery v smere Y:

- calibrateAxisY – funkcia na kalibráciu výpočtu vzdialenosti kamery od objektu záujmu – vstupom je umiestnenie dvoch kalibračných obrázkov najlepšie znázorňujúce objekt záujmu v čo najmenšej vzdialenosti a čo najväčšej, výstupom je k a q koeficient závislosti vzdialenosti od veľkosti objektu na obrázku.
- getObjectDistance – výpočet vzdialenosti objektu z:
    - veľkosti objektu v pixeloch a veľkosti v centimetroch
    - Detekciou veľkosti objektu z obrázka pomocou knižnice DetectTarget a veľkosti objektu v centimetroch
    - určenia veľkosti objektu z bodov kružnice a veľkosti objektu v centimetroch (vstupom budú body z optického toku)
- getObjectMotionY – výpočet zmeny polohy kamery voči objektu medzi dvomi obrázkami – vstupom je vzdialenosť aktuálna a predchádzajúca.
- getLineEquation – výpočet rovnice priamky z dvoch bodov.
- printDistanceOfMovingObject, printMovementVectorLengthY, printMovementVectorLengthY-optflow – ukážka predošlých funkcií s vypísaním hodnôt posunutia do konzoly.

Výpočet vzdialenosti o ktorú sa kamera posunula je založený na princípe veľkosti objektu v pixeloch, tak ako bolo popísané v teoretickej časti. Uprednostnili sme túto metódu pred výpočtom pomocou vektorov získaných z optického toku kvôli jej jednoduchosti, robustnosti a uľahčeniu separácie pohybu v x, y, z osiach.

8. Zobrazenie pohybu kamery

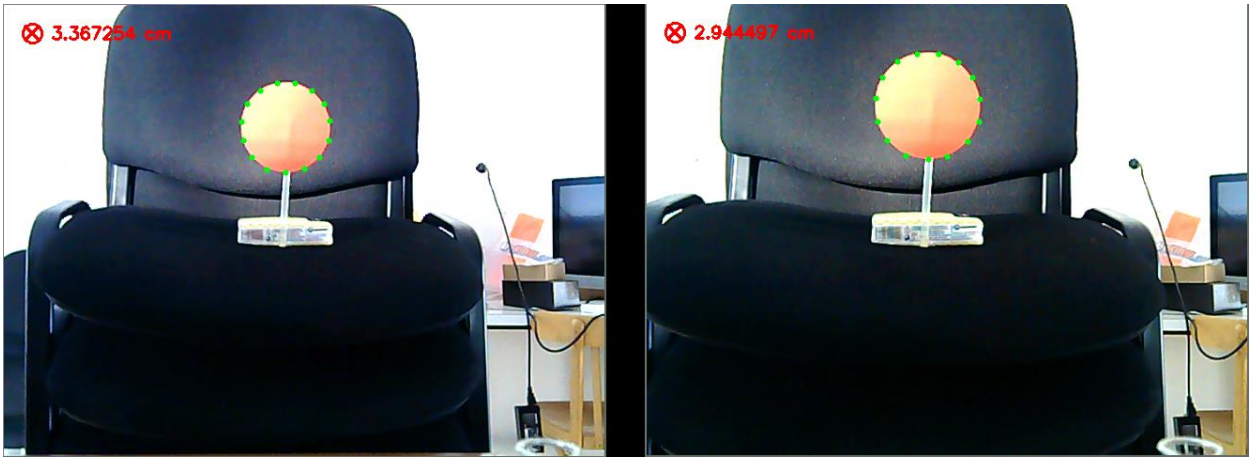
Obsahuje dve funkcie na vizualizáciu posunutia kamery:

- displayVectorY – zobrazí vektor v smere osi Y určujúci smer pohybu a zobrazí hodnotu posunutia v centimetroch – vstupom je smerník na cv maticu, kde sa má vektor zobraziť, hodnota posunutia a bod, na ktorom sa má vektor zobraziť.

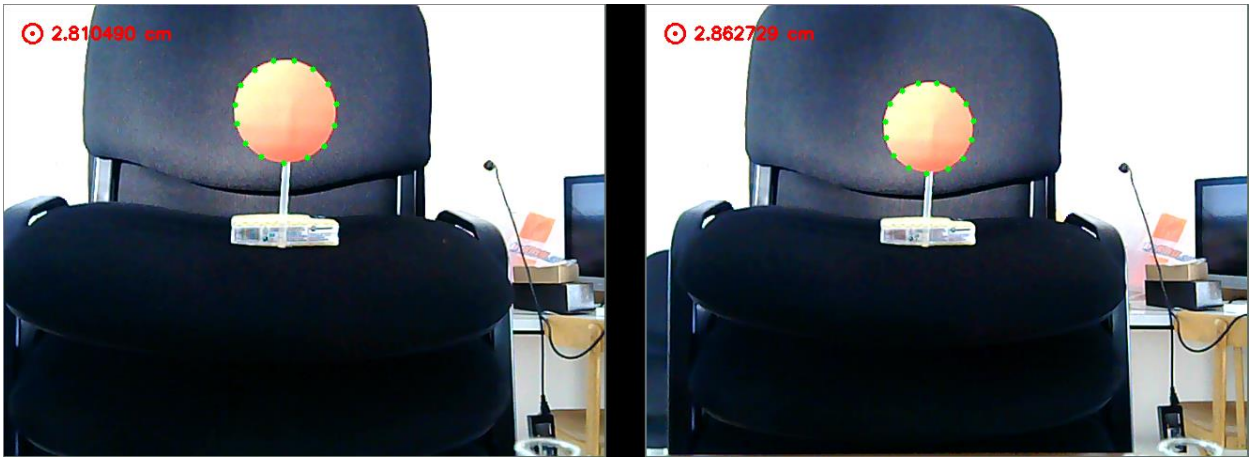
- displayVectorXZ – zobrazí vektor v smere osi X a Z určujúci smer a veľkosť pohybu – vstupom je smerník na cv maticu, kde sa má vektor zobraziť, hodnota posunutia, uhol posunutia a bod, na ktorom sa má vektor zobraziť. Takisto zobrazí hodnotu posunutia a uhol posunutia.

Výsledok odometrie

Posun kamery smerom k objektu v osi Y



Posun kamery smerom od objektu v osi Y

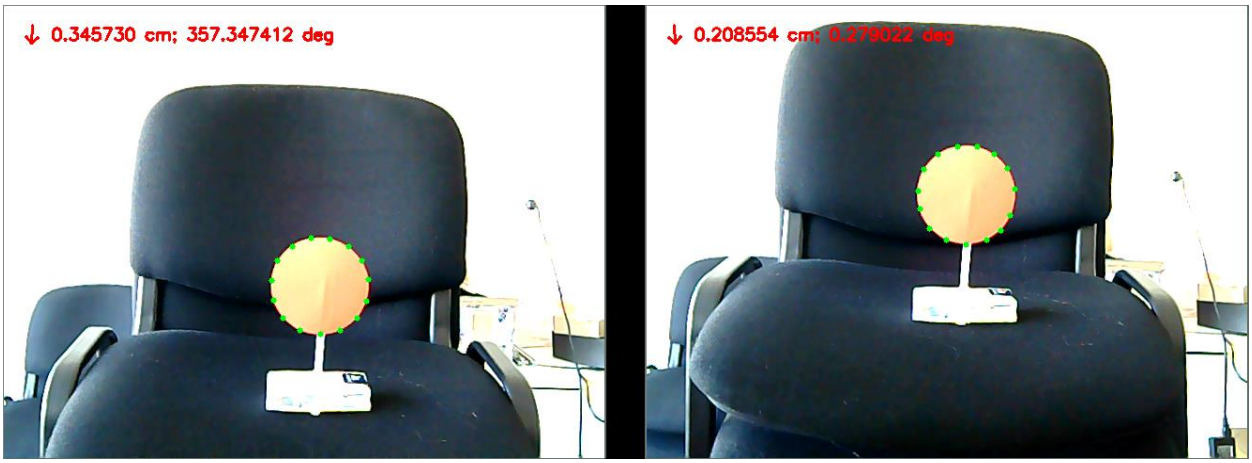


Posun kamery vpravo v osi X

Posun kamery vľavo v osi X

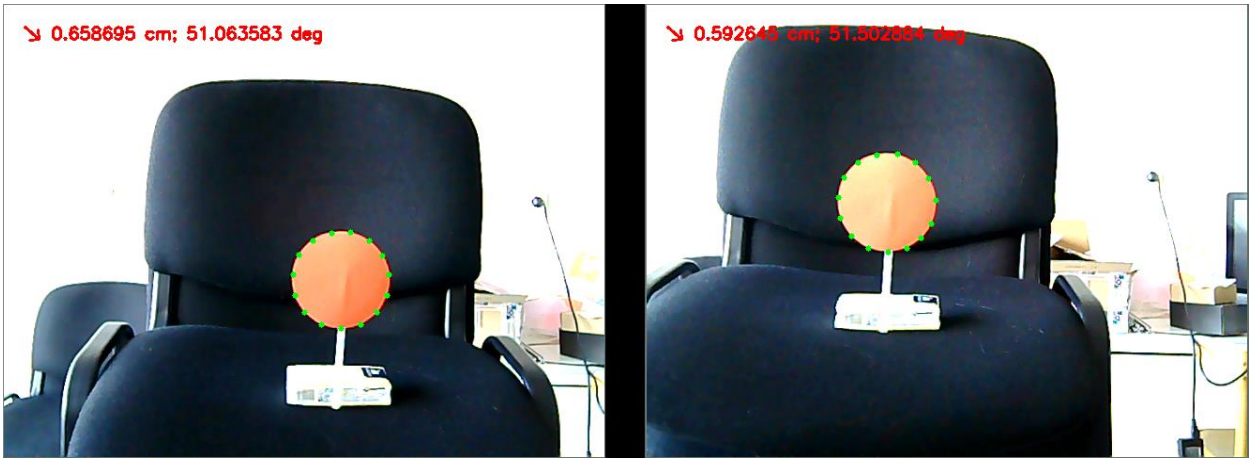
Posun kamery hore v smere Z

Posun kamery dole v smere Z



Posun kamery smerom hore vľavo v osi X a súčasne Z

Posun kamery smerom dole vpravo v osi X a súčasne Z



Zdroje

http://docs.opencv.org/2.4/doc/tutorials/introduction/windows_install/windows_install.html

http://docs.opencv.org/2.4/doc/tutorials/introduction/windows_visual_studio_Opencv/windows_visual_studio_Opencv.html

https://github.com/opencv/opencv/blob/master/samples/cpp/lkdemo.cpp

http://docs.opencv.org/2.4/modules/imgproc/doc/feature_detection.html?highlight=houghcircles

http://docs.opencv.org/2.4/modules/core/doc/drawing_functions.html

http://docs.opencv.org/2.4/modules/imgproc/doc/miscellaneous_transformations.html