ראייה אנושית גישה חישובית

תרגיל 2

מגישים: רז דרזי ורן שחם

3 במאי 2017

שאלה 5

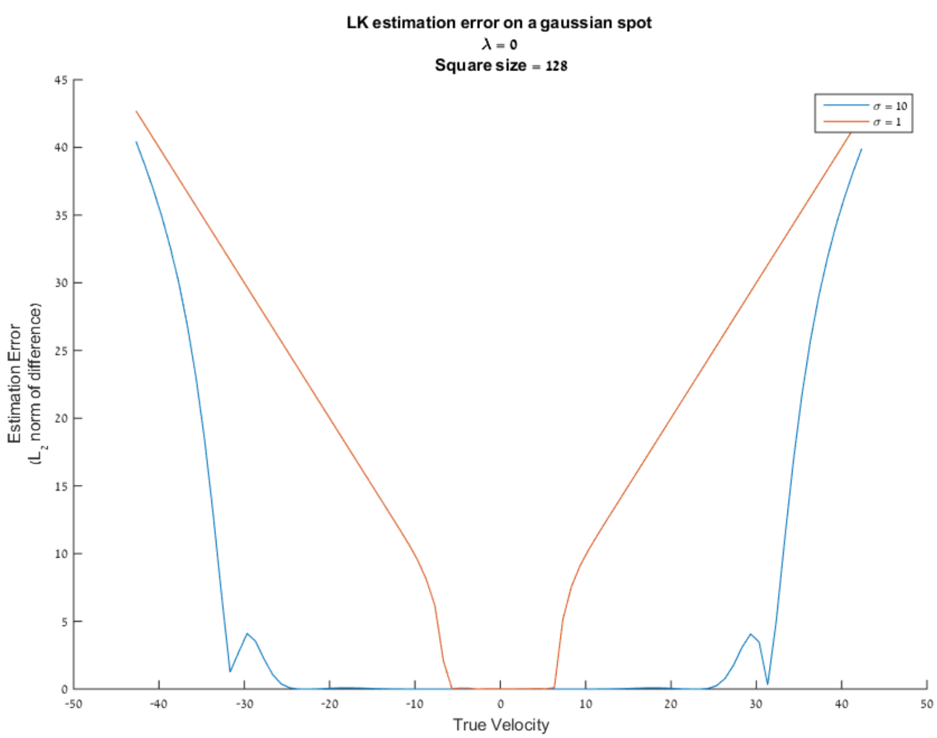
בהרצת mymovie נראה שהעץ זז מהר יותר מהפרחים. כמוכן, לאחר הרצת האלגוריתם על תתי־תמונות של העץ התקבלה מהירות גדולה יותר מאשר זאת שהתקבלה על ידי הרצתו על הפרחים. כמוכן, כצפוי, רכיב המהירות המשמעותי הוא האופקי.

(ממוצעי) פלט האלגוריתם מסוכמים בטבלה הבאה:

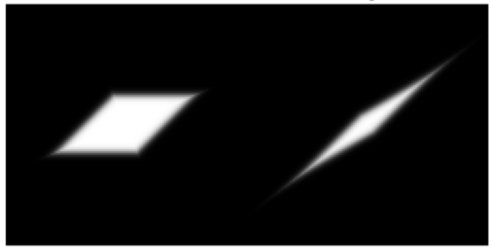
	רכיב אופקי	רכיב אנכי
עץ	-1.8522	-0.2150
פרחים	-1.1541	0.0034

את התוצאות עבור כל תת־תמונה ואת המסכות המגדירות את תתי־התמונה ניתן לראות בנספח הקוד במסמך זה.

שאלה 6



איור 1: שאלה 6



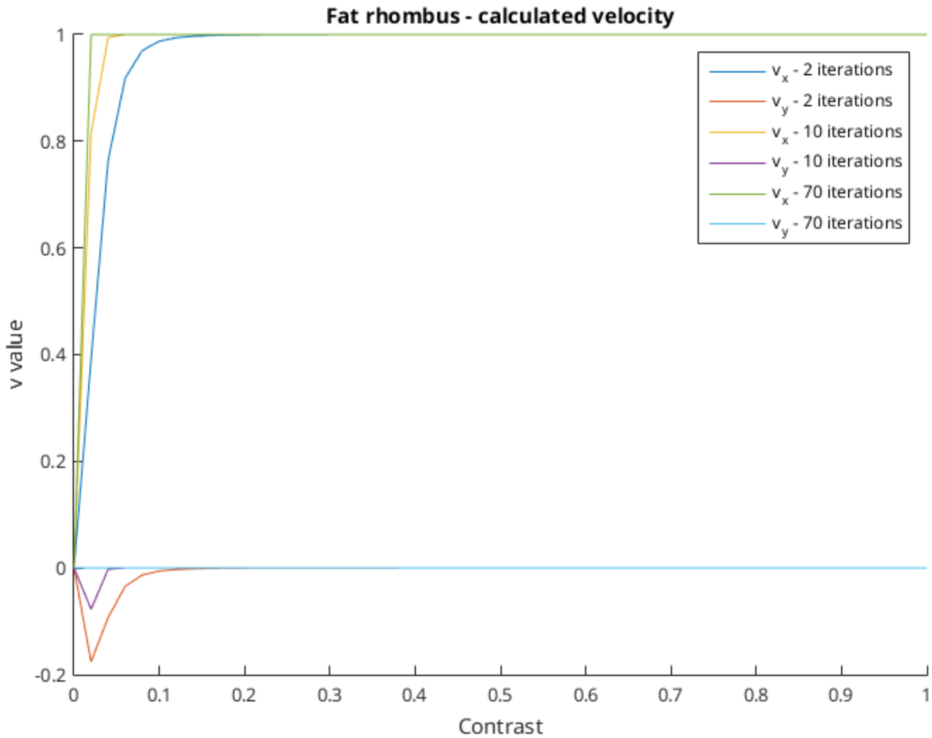
איור 2: המקביליות המתקבלות מ־rhombusMovie

אפשר לראות שעבור מהירויות נמוכות ($|v| < 10$) האלגוריתם מחשב בדיוק את המהירות האופקית. עבור מהירויות גבוהות יותר, השגיאה גדלה ככל שהמהירות (בערך מוחלט) גדלה, וכן השגיאה של הכתם הצר ($\sigma = 1$) גדולה משל הכתם הרחב ($\sigma = 10$).

את האפקט הראשון אפשר להסביר באמצעות הפרת ההנחות של אלגוריתם לוקאס־קנדה: האלגוריתם משתמש בפיתוח טיילור סביב הפריים הראשון בתמונה, כאשר ההנחה היא שהפריים השני הוא הזזה קטנה של הראשון ־ ופיתוח טיילור מדוייק פחות ככל שמתרחקים מהנק' סביבה הוא מוגדר.

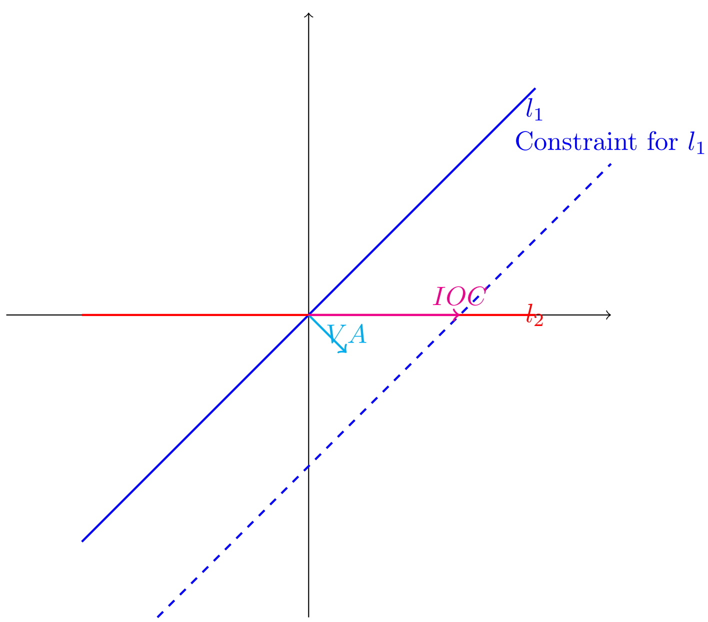
האפקט השני יכול להיות מוסבר על ידי הנחה נוספת באלגוריתם ־ והיא שהתמונה השנייה היא הזזה של הראשונה ־ כלומר שמתקיים $I_2(x,y) = I_1(x+v_x, y+v_y)$. לכן בחישוב v אנחנו סוכמים את הנגזרות על פני כל התמונה. היות והכתם הרחב מהווה חלק משמעותי יותר מהתמונה, ההנחה הזאת מתקיימת בצורה מלאה יותר ולכן שגיאת האלגוריתם קטנה יותר. עבור הכתם הקטן ההנחה מופרת ולכן השגיאה גדלה מהר.

החל ממקום מסויים, השגיאה גדלה באופן לינארי במהירות. אפשר להסביר את זה על ידי כך שהאלגוריתם, החל ממהירות מסויימת בה ההנחות מופרות בצורה משמעותית מספיק, האלגוריתם מחשב איזשהו ערך שגוי קבוע וכך ההפרש בין המהירות האמיתית לערך זה גדל לינארית. אינטואיטיבית, זה קורה בגלל חוסר היכולת לזהות את האובייקט בתמונה השנייה עם האובייקט בראשונה ־ מקרה זה דומה למצב בו עצם נעלם בתמונה הראשונה ועצם מופיע בשנייה ־ ובגלל היותו רחוק ממיקומו בראשונה ניתן להניח שזה עצם אחר, ולא תנועה של עצם אחד.

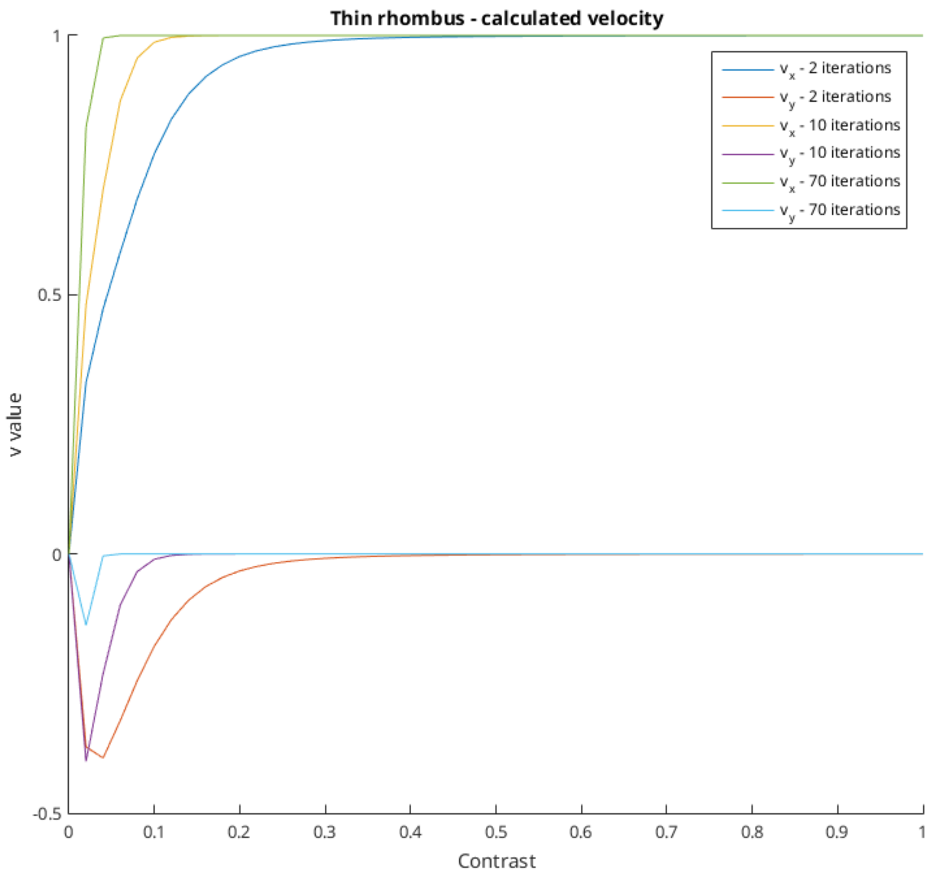


איור 3: גרף המהירות המחושבת כפונקציה של קונטרסט - מקבילית שמנה

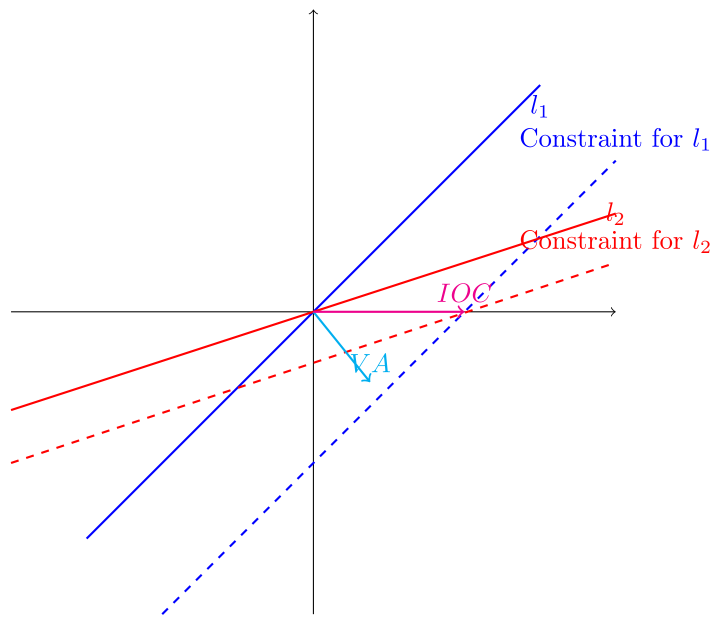
שאלה 7



איור 5: תרשים עבור המקבילית השמנה - קו אחד בזווית $\theta_1 = 45°$ והשני בזווית $\theta_2 = 0°$



איור 4: גרף המהירות המחושבת כפונקציה של קונטרסט - מקבילית רזה



איור 6: תרשים עבור המקבילית הצרה - קו אחד בזווית $\theta_1 = 45°$ והשני בזווית $\theta_2 = 30°$

תחילה נבחין שהמהירות המקורית של כל קו נמצאת על ישר האילוץ שלו. אם כך, בשני המקרים (מקבילית שמנה/רזה), חיתוך האילוצים חייב להימצא בנקודה $(1,0)$ - כי ישר האילוץ של כל קו עובר בנק' זו. לכן, בשני המקרים $IOC = (1,0)$.

כעת, עבור **המקבילית הרחבה**, ברור שהרכיב הניצב לישר l_2 הוא $v_2^{\perp} = 0$, לכן נותר לחשב את הרכיב $v_1^{\perp}$ ל-l_1 (הישר בזווית 45 מע'). קל לראות שהרכיב הניצב לו הוא בכיוון $45°$ כלפי מטה, כלומר בזווית של $315°$. אורכו הוא אורך של ניצב במשולש ישר זוית ושווה שוקיים שאורך היתר בו הוא 1, לכן אורכו $\frac{1}{\sqrt{2}}$. אם כך:

$$v_1^{\perp} = \frac{1}{\sqrt{2}} \begin{pmatrix} \cos 315° \\ \sin 315° \end{pmatrix} = \frac{1}{2} \begin{pmatrix} 1 \\ -1 \end{pmatrix}$$

לכן:

$$VA = \frac{1}{2}\left(v_1^{\perp} + v_2^{\perp}\right) = \frac{1}{2} v_1^{\perp} = \frac{1}{4} \begin{pmatrix} 1 \\ -1 \end{pmatrix}$$

עבור **המקבילית הצרה**, נותר לחשב את הרכיב הניצב לישר l_2 בשיפוע $30°$. שיפועו הוא $60°$ כלפי מטה, כלומר $300°$. המשולש הנוצר על ידי הרכיב הניצב, הרכיב המקביל ווקטור המהירות הוא בעל הזוויות $30, 60, 90$ מע', לכן גודל היתר הקטנה הוא גודל הרכיב הניצב, והוא חצי מאורכה של היתר - לכן אורך הרכיב הניצב הוא $\frac{1}{2}$. לכן:

$$v_2^{\perp} = \frac{1}{2} \begin{pmatrix} \cos 300° \\ \sin 300° \end{pmatrix} = \frac{1}{4} \begin{pmatrix} 1 \\ -\sqrt{3} \end{pmatrix}$$

ונקבל:

$$VA = \frac{1}{2}\left(v_1^{\perp} + v_2^{\perp}\right) = \frac{1}{2}\left(\frac{1}{2}\begin{pmatrix} 1 \\ -1 \end{pmatrix} + \frac{1}{4}\begin{pmatrix} 1 \\ -\sqrt{3} \end{pmatrix}\right) = \frac{1}{8}\begin{pmatrix} 3 \\ -2-\sqrt{3} \end{pmatrix}$$

נשים לב שגם עבור המעויין השמן וגם עבור הרזה, בקונטרסט גבוה האלגוריתם מחשב את המהירות האמיתית - $(1,0)$. כמוכן, בקונטרסט 0 האלגוריתם מחשב מהירות 0 - תוצאה הגיונית בהתחשב בכך שהתמונה חלקה ואין תנועה. הבדל בין סוגי המקבילית מתקבל כאשר

הקונטרסט נמוך - במקרה זה האלגוריתם מחשב עבור המקבילית הצרה מהירות תנועה שקרובה יותר ל-VA מאשר ל-IOC. עבור המקבילית הרחבה באותו קונטרסט התוצאה קרובה יותר ל-IOC. תוצאה זו קרובה מאוד לחוויה מהתבוננות ב-rhombus demo - כאשר המקבילית רחבה יחסית קל לזהות שכיוון התנועה הוא אופקי גם בקונטרסט נמוך; כאשר המקבילית צרה אנחנו (והאלגוריתם) נוטים לייחס לה תנועה בכיוון ה-VA - וזו תוצאה מעניינת לגבי האלגוריתם (או לגבי מערכת תפישת הראייה האנושית).

שאלה 8

נגדיר מ"מ $x\sim N\left(\mu_p,\sigma_p^2\right)$ ונתונה מדידה רועשת y כך ש-$y|x\sim N\left(x,\sigma_y^2\right)$. נראה:

$$\hat{x}=\arg\max_x p\left(x|y\right)=\frac{\frac{1}{\sigma_p^2}\mu_p+\frac{1}{\sigma_y^2}y}{\frac{1}{\sigma_p^2}+\frac{1}{\sigma_y^2}}$$

מנוסחת בייס מתקיים:

$$p\left(x|y\right)=\frac{p\left(y|x\right)p\left(x\right)}{p\left(y\right)}$$

לכן:

$$\begin{aligned}
\hat{x}&=\arg\max_x p\left(x|y\right)=\arg\max_x\frac{p\left(y|x\right)p\left(x\right)}{p\left(y\right)}\overset{1}{=}\arg\max_x p\left(y|x\right)p\left(x\right)\\
&\overset{2}{=}\arg\max_x\frac{1}{\sqrt{2\pi}\sigma_y}\exp\left(\frac{-\left(y-x\right)^2}{2\sigma_y^2}\right)\cdot\frac{1}{\sqrt{2\pi}\sigma_p}\exp\left(\frac{-\left(x-\mu_p\right)^2}{2\sigma_p^2}\right)\\
&\overset{3}{=}\arg\max_x\exp\left(-\frac{\left(y-x\right)^2}{2\sigma_y^2}-\frac{\left(x-\mu_p\right)^2}{2\sigma_p^2}\right)\overset{4}{=}\arg\max_x\log\exp\left(-\frac{\left(y-x\right)^2}{2\sigma_y^2}-\frac{\left(x-\mu_p\right)^2}{2\sigma_p^2}\right)\\
&=\arg\max_x-\frac{\left(y-x\right)^2}{2\sigma_y^2}-\frac{\left(x-\mu_p\right)^2}{2\sigma_p^2}=:\arg\max_x f\left(x\right)
\end{aligned}$$

כאשר:

1. ההסתברות $p\left(y\right)$ אינה תלויה ב-x (וכמובן שהיא אי-שלילית), לכן ערך זה אינו משפיע על מקס' הפונקציה
2. לפי הגדרת $y|x$ ו-x (מ"מ נורמליים)
3. הורדת הפקטור הכפלי (שאינו משפיע על ה-x המביא את הפונ' למקסימום) ושימוש בחוקי אקספוננט
4. מונוטוניות של $\log$ - x הממקסם את הפונ' ימקסם גם את $\log$הפונ'

אם כך נמצא x הממקסם את הפונקצייה f שהתקבלה לעיל. נגזור ונשווה ל-0:

$$\begin{aligned}
0=f'\left(x\right)&=2\frac{\left(y-x\right)}{2\sigma_y^2}-2\frac{\left(x-\mu_p\right)}{2\sigma_p^2}=\frac{y}{\sigma_y^2}+\frac{\mu_p}{\sigma_p^2}-x\left(\frac{1}{\sigma_y^2}+\frac{1}{\sigma_p^2}\right)\\
\Longleftrightarrow x&=\frac{\frac{1}{\sigma_p^2}\mu_p+\frac{1}{\sigma_y^2}y}{\frac{1}{\sigma_p^2}+\frac{1}{\sigma_y^2}}
\end{aligned}$$

נוודא שזו אכן נק' מקסימום על ידי גזירה נוספת:

$$f''\left(x\right)=-\left(\frac{1}{\sigma_y^2}+\frac{1}{\sigma_p^2}\right)<0$$

כי σ_y^2,σ_p^2 קבועים חיוביים - ולכן f מתמקסמת ב-$\hat{x}$ כנ"ל כנדרש.

קוד

```matlab
function [Ix, Iy, It] = ImageDerivatives(I1, I2)
%IMAGEDERIVATIVES Calculates the derivative of the given images
%   Parameters
%   ----------
%   I1 - one frame of an image
%   I2 - another frame (size identical to I1)
%   Returns
%   -------
%   Ix - the derivative of the frame on the x axis
%   Iy - the derivative of the frame on the y axis
%   It - the derivative of the frame over time

    % kernels and constants
    Ky = 0.25 * [-1, -1; 1,  1];
    Kx = -Ky';
    Kt = 0.25 * ones(2, 2);
    CONV_PARAM = 'same';
    % actual work
    Ix = conv2(I1, Kx, CONV_PARAM) ...
        + conv2(I2, Kx, CONV_PARAM);
    Iy = conv2(I1, Ky, CONV_PARAM) ...
        + conv2(I2, Ky, CONV_PARAM);
    It = conv2(I2, Kt, CONV_PARAM) ...
        - conv2(I1, Kt, CONV_PARAM);

end
```

```
function  v = LK_alg(I1, I2, lambda, mask, ...
                      v_initial, num_iterations)
%LK_ALG Runs the Lucas Kanade iterative algorithm for calc. optical flow
%   Parameters
%   ----------
%   I1 - the first frame of an image
%   I2 - the second frame of an image (same size as I1)
%   lambda - the noise variance to prior variance ratio (scalar)
%   mask - area of the image to sum upon (same size as I1)
%   v_initial - initial guess for the velocity (2d vector)
%   num_iterations - ... y'know
%   Returns
%   -------
%   v - the computed velocity (2d vector)

    At = zeros(size(I1,1), size(I1,2), 4);
    Bt = zeros(size(I1,1), size(I1,2), 2);
    v = v_initial;
    for i = 1:num_iterations
        [I2w, warpMask] = warp(I2, v);
        newMask = mask .* warpMask;
        [Ix, Iy, It] = ImageDerivatives(I1, I2w);
        Ix = Ix .* newMask;
        Iy = Iy .* newMask;
        It = It .* newMask;
        At(:, :, 1) = Ix.^2;
        At(:, :, 2) = Ix .* Iy;
        At(:, :, 3) = At(:, :, 2);
        At(:, :, 4) = Iy.^2;
        Bt(:, :, 1) = Ix .* It;
        Bt(:, :, 2) = Iy .* It;
        A = reshape(sum(sum(At, 1), 2), 2, 2) + eye(2).*lambda;
        B = -reshape(sum(sum(Bt, 1), 2), 2, 1);
        v = v + A \ B;
    end

end
```

```matlab
function blurredI = blur_downsample(I)
%BLUR_DOWNSAMPLE Reduces image size by a factor of 2
%   Parameters
%   ----------
%   I - an image
%   Returns
%   -------
%   blurredI - a blurred and downsampled image (half the size of I)

    kernel = load('GaussKernel.mat');
    kernel = kernel.GaussKernel;
    I = conv2(I, kernel, 'same');
    blurredI = I(1:2:end, 1:2:end);

end
```

Published with MATLAB® R2014b

```
function v = Full_LK(I1, I2, lambda, mask, num_iterations)
%FULL_LK The full version of the algorithm
%   for parameters reference see LK_alg.m

    % get the initial guess
    v = [0; 0];
    I1b = blur_downsample(I1);
    I2b = blur_downsample(I2);
    v = LK_alg(I1b, I2b, lambda, mask(1:2:end, 1:2:end), v, 1);
    % run the algorithm with the initial guess
    v = LK_alg(I1, I2, lambda, mask, v.*2, num_iterations);

end
```

Published with MATLAB® R2014b

Contents

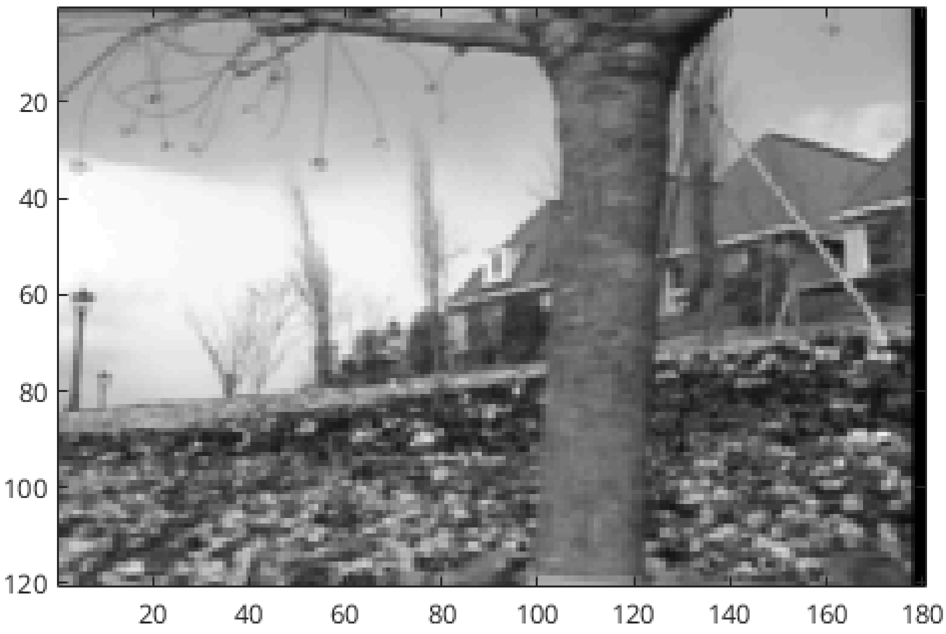
question 5

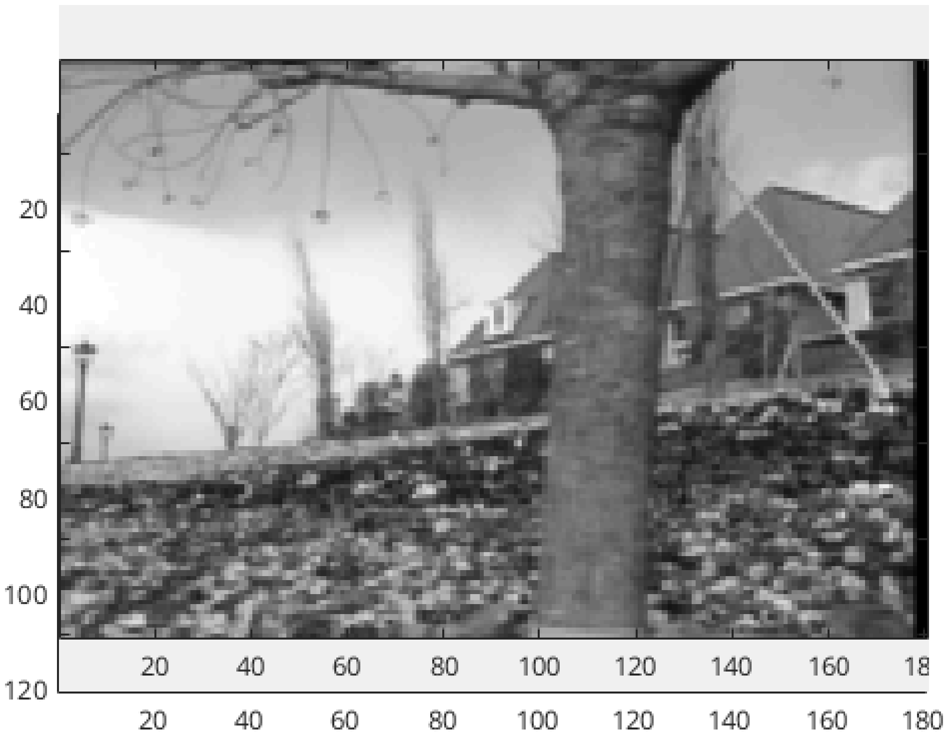
read images and view them

```
I1 = im2double(imread('flower-i1.tif'));
I2 = im2double(imread('flower-i2.tif'));
mymovie(I1, I2);

% define the algorithm's parameters
treeMasks = zeros(size(I1,1), size(I1,2), 3);
flowersMasks = zeros(size(I1,1), size(I1,2), 3);
treeMasks(1:40, 90:130, 1) = 1;
treeMasks(41:80, 90:130, 2) = 1;
treeMasks(81:end, 90:130, 3) = 1;
flowersMasks(90:end, 1:40, 1) = 1;
flowersMasks(85:end, 41:80, 2) = 1;
flowersMasks(80:end, 140:end, 3) = 1;
lambda = 0;
num_iterations = 100;

v_tree = zeros(2, 1, 3);
v_flowers = zeros(2, 1, 3);

% for each tree/flowers subimage, run the LK algorithm
for i = 1:3
    v_tree(:,:, i) = Full_LK(I1, I2, lambda, treeMasks(:,:,i), ...
                             num_iterations);
    v_flowers(:,:, i) = Full_LK(I1, I2, lambda, ...
                                flowersMasks(:,:,i), num_iterations);
end
v_tree
v_flowers
mean_v_tree = mean(v_tree, 3)
mean_v_flowers = mean(v_flowers, 3)
```

```
v_tree(:,:,1) =

   -2.7673
   -0.4681


v_tree(:,:,2) =

   -0.9555
   -0.1756


v_tree(:,:,3) =

   -1.8337
   -0.0013


v_flowers(:,:,1) =

   -1.1373
    0.0208
```

```
v_flowers(:,:,2) =

   -1.0938
   -0.0014


v_flowers(:,:,3) =

   -1.2313
   -0.0091


mean_v_tree =

   -1.8522
   -0.2150


mean_v_flowers =

   -1.1541
    0.0034
```

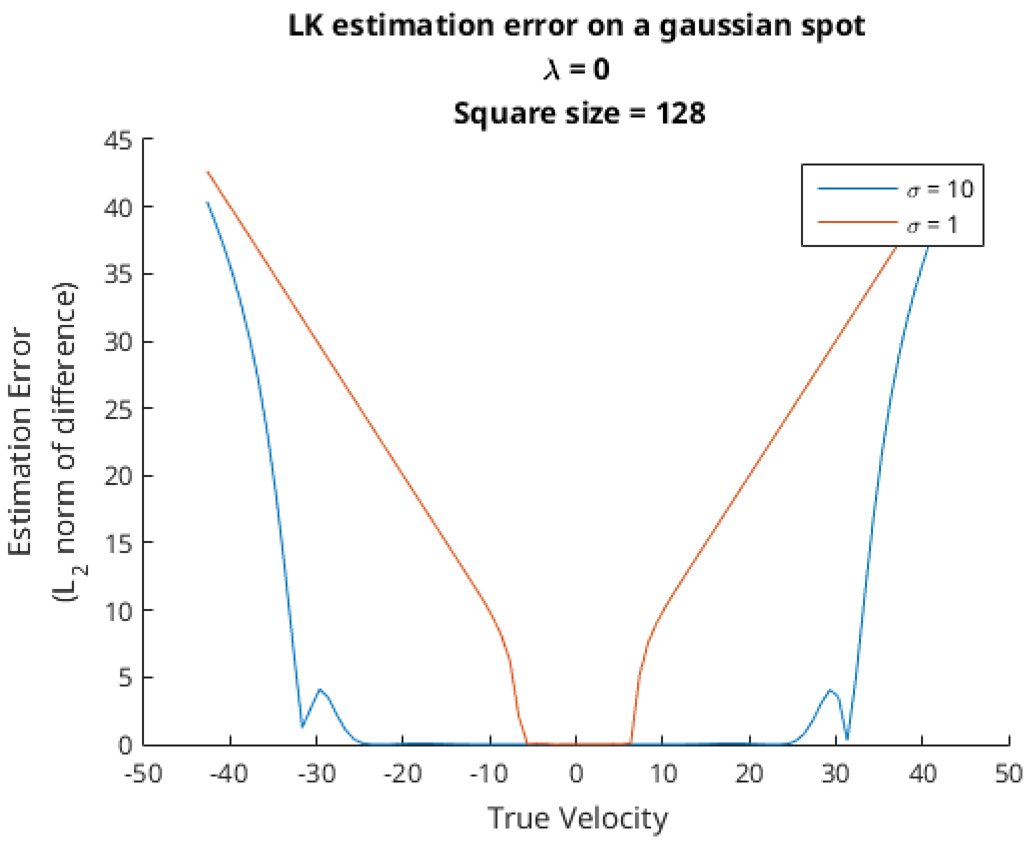




question 6

```
REAL_VELOCITY = 1;
L2_DIFF = 2;

squareSize = 128;
sigmas = [10, 1];
%lambda = 0.001;
lambda = 0;
mask = ones(squareSize);
num_iterations = 1;
maxVelocity = squareSize / 3;
velocities = (-maxVelocity):maxVelocity;
results = zeros(length(velocities), 2);
```

```
figure;
hold on;
for sigma = sigmas
    firstFrame = GausSpot(squareSize, sigma, [0, 0]);
    for i = 1:numel(velocities)
        real_v = velocities(i);
        secondFrame = GausSpot(squareSize, sigma, [real_v, 0]);
        estimated_v = Full_LK(firstFrame, secondFrame, lambda, ...
                              mask, num_iterations);
        results(i, REAL_VELOCITY) = real_v;
        % the L2 norm of a scalar is the abs value
        results(i, L2_DIFF) = abs(estimated_v(1) - real_v);
    end
    plot(results(:, REAL_VELOCITY), results(:, L2_DIFF));
end
hold off;
title({'LK estimation error on a gaussian spot', ...
       ['\lambda = ', num2str(lambda)], ...
       [' Square size = ', num2str(squareSize)]});
xlabel('True Velocity');
ylabel({'Estimation Error', '(L_2 norm of difference)'});
legend(['\sigma = ', num2str(sigmas(1))], ...
       ['\sigma = ', num2str(sigmas(2))]);
```

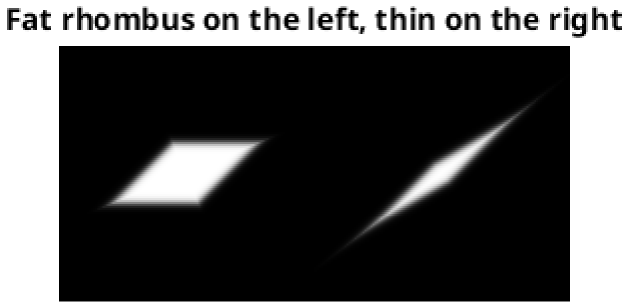


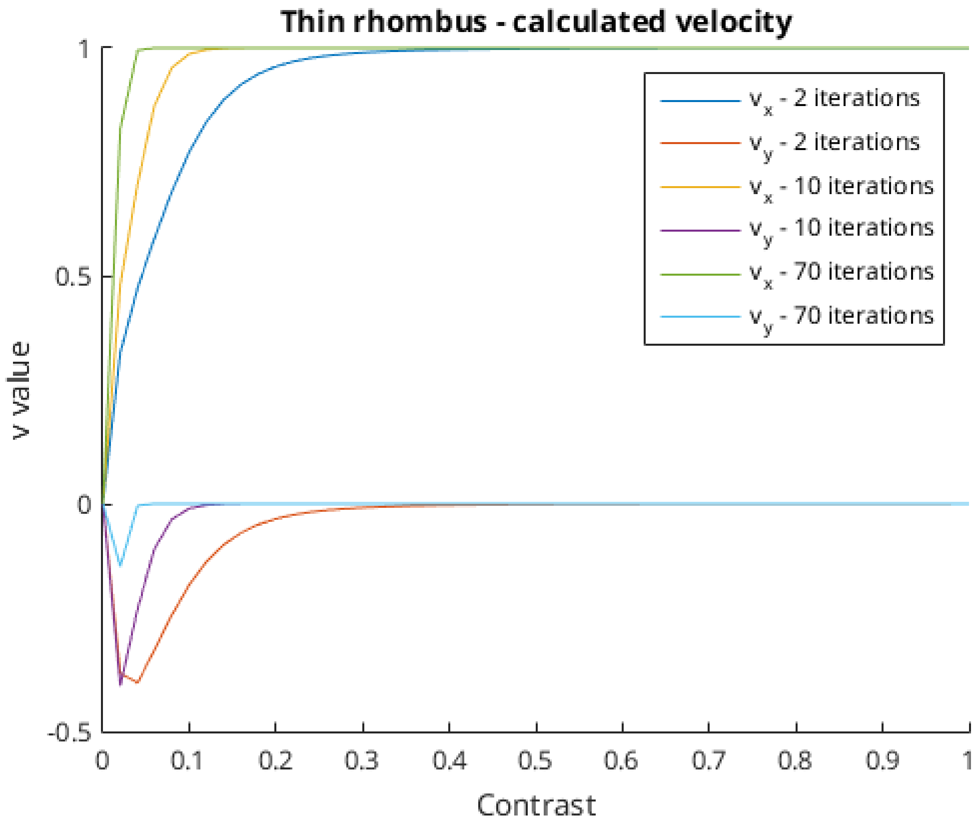
question 7

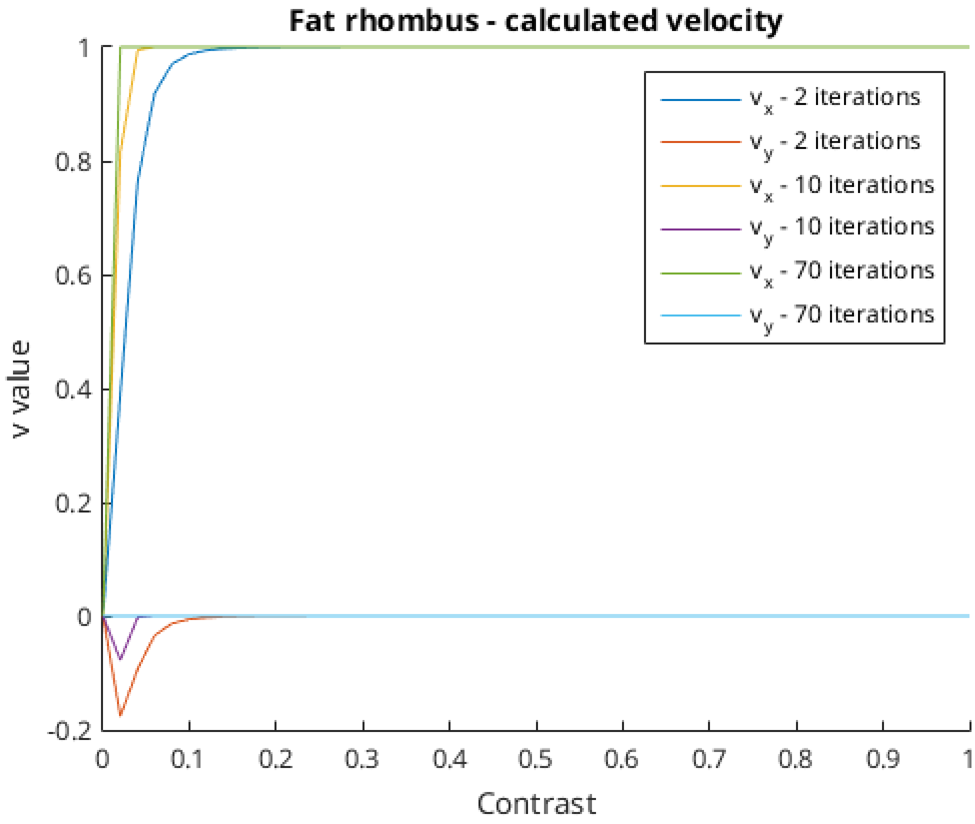
constants

```
fatStr = {'Thin', 'Fat'};
THIN = 0;
FAT = 1;
lambda = 0.01;
contrasts = 1:-0.01:0;
iters = [2, 10, 70];
legends = cell(2, numel(iters));

% plotting
fatRhom = rhombusMovie(1, 1);
thinRhom = rhombusMovie(0, 1);
figure; imshow([fatRhom, thinRhom]);
```

```
title('Fat rhombus on the left, thin on the right');

for fatFlag = [THIN, FAT]
    figure;
    for i = 1:numel(iters)
        iter = iters(i);
        hold all;
        plotRhombus(fatFlag, lambda, iter);
        legends{1, i} = sprintf('v_x - %d iterations', iter);
        legends{2, i} = sprintf('v_y - %d iterations', iter);
    end
    legend(legends{:})
    title([fatStr{fatFlag+1}, ' rhombus - calculated velocity']);
    hold off;
end
```







Published with MA TLAB® R2015a