

1. 实验名称：基于匹配的平移跟踪

1.1. 实验目的

通过计算目标模板与图像的相关曲线，得出跟踪目标的当前位置；深入理解各种相关函数的物理含义与具体效能。

1.2. 实验内容

- 使用 Matlab 中 `imcrop` 函数从基准图或序列图像的首幅图中以合适图像大小，使用 `rgb2gray` 函数将模板转为灰度图像，取出目标区域作为模板，或者根据目标成像特征制作理想模板用目标模板。
- 将序列图像都转为灰度图像，模板在当前帧图像上有秩序地移动，与当前图像待搜索区域进行多种相关性计算，找到相关系数最大值或最小值点，即认定为目标在当前图像上的位置。
实验中使用相关法的数学形式包括：

- 直接相关：

$$C(x, y) = \sum_{(i,j) \in W} f(x+i, y+j)g(i, j)$$

- 均值归一化相关：

$$C(x, y) = \sum_{(i,j) \in W} [f(x+i, y+j) - f_m][g(i, j) - g_m]$$

- 标准化相关：

$$C(x, y) = \frac{\sum_{(i,j) \in W} f(x+i, y+j)g(i, j)}{\sqrt{\sum_{(i,j) \in W} f^2(x+i, y+j) \sum_{(i,j) \in W} g^2(i, j)}}$$

- 方差归一化相关：

$$C(x, y) = \frac{\sum_{(i,j) \in W} [f(x+i, y+j) - f_m][g(i, j) - g_m]}{\sqrt{\sum_{(i,j) \in W} [f(x+i, y+j) - f_m]^2 \sum_{(i,j) \in W} [g(i, j) - g_m]^2}}$$

- 最小二乘相关：

$$C(x, y) = \sum_{(i,j) \in W} [f(x+i, y+j) - g(i, j)]^2$$

- 差绝对值和：

$$C(x, y) = \sum_{(i,j) \in W} |f(x+i, y+j) - g(i, j)|$$

- 使用模板与序列图像逐帧进行匹配，进行目标的识别和跟踪，并使用 `tic` 函数统计各相关方法的平均运算时间进行对比。

1.3. 实验代码

```

%% prepare for start %%
clc;
clear all;
close all;

images_folder = "images_array";
origin_images = {"tiger_1.jpg", "tiger_2.jpg", "tiger_3.jpg", "tiger_4.jpg",
    "tiger_5.jpg", "tiger_6.jpg", "tiger_7.jpg", "tiger_8.jpg", "tiger_9.jpg",
    "tiger_10.jpg", "tiger_11.jpg", "tiger_12.jpg", "tiger_13.jpg", "tiger_14.jpg",
    "tiger_15.jpg", "tiger_16.jpg", "tiger_17.jpg", "tiger_18.jpg", "tiger_19.jpg",
    "tiger_20.jpg" };

first_frame = imread("images_array/tiger_1.jpg");
first_frame = imresize(first_frame, 0.25);
first_template = imcrop(rgb2gray(first_frame));
imwrite(first_template, 'first_template.jpg');

for i = 1:length(origin_images)
    image_name = fullfile(images_folder, origin_images{i});
    target = rgb2gray(imread(image_name));
    gray_name = sprintf("targets_bk_gray/gray_%d.jpg", i);
    imwrite(target, gray_name);
end

%% main function %%
template = imread("first_template_abs.jpg");
template = imresize(template, 0.25);
[template_height, template_width] = size(template);

grayimg_folder = 'targets_abs';
targets = { "gray_1.jpg", "gray_2.jpg", "gray_3.jpg", "gray_4.jpg", "gray_5.jpg",
    "gray_6.jpg", "gray_7.jpg", "gray_8.jpg", "gray_9.jpg", "gray_10.jpg",
    "gray_11.jpg", "gray_12.jpg", "gray_13.jpg", "gray_14.jpg", "gray_15.jpg",
    "gray_16.jpg", "gray_17.jpg", "gray_18.jpg", "gray_19.jpg", "gray_20.jpg" };

for i = 1:length(targets)
    target_name = fullfile(grayimg_folder, targets{i});
    target = imread(target_name);
    target = imresize(target, 0.25);

    c = directCorr(template, target);
    c = covCorr(template, target);
    c = normCorr(template, target);
    c = manual_normxCorr(template, target);
    c = normxcorr2(template, target);
    c = leastSquares(template, target);
    c = absDiff(template, target);

    [ypeak, xpeak] = find(c==max(c(:)));

```

```

[ypeak, xpeak] = find(c==min(c(:)));

% colormap("gray");
% imshow(c, []);

xoffset = xpeak - template_width/2;
yoffset = ypeak - template_height/2;

matched_tiger = insertShape(target, 'Rectangle', [xoffset, yoffset,
template_width, template_height], 'Color', 'red', 'LineWidth', 2);

matched_name = sprintf("directcorr_images/matched_tiger_%d.jpg", i);
matched_name = sprintf("covariance_images/matched_tiger_%d.jpg", i);
matched_name = sprintf("normcorr_images/matched_tiger_%d.jpg", i);
matched_name = sprintf("manual_normxcorr_images/matched_tiger_%d.jpg", i);
matched_name = sprintf("normxcorr2_images/matched_tiger_%d.jpg", i);
matched_name = sprintf("leastquares_images/matched_tiger_%d.jpg", i);
matched_name = sprintf("abs_images/matched_tiger_%d.jpg", i);

imwrite(matched_tiger, matched_name);

template = imcrop( target, [xoffset, yoffset, template_width,
template_height] );

end

%% 直接相关 %%
function directcorrelationmap = directCorr(template, image)
    directcorrelationmap = filter2(template, image);
end

%% 协方差相关 %%
function covariancemap = covCorr(template, image)
    template_mean = mean2(template(:));
    [j, i] = size(template);
    covariancemap = filter2(template, image) - template_mean * filter2(ones(j, i),
image);
end

%% 标准化相关 %%
function normcorrmap = normCorr(template, image)
    amap = filter2(template, image);
    bmap = sqrt( filter2(ones(size(template)), image.^2) * sum(template(:).^2) );
    normcorrmap = amap./bmap;
end

%% 标准化协方差相关 %%
function normxcorrmap = manual_normxCorr(template, image)
    template_mean = mean2(template(:));
    [j, i] = size(template);

```

```

mean_filter = ones(j, i) / (j * i);
mean_image = filter2(mean_filter, image);

covariancemap = filter2(template, image) - template_mean * filter2(ones(j, i),
image);
down_map = sqrt( (filter2(ones([j, i]), image.^2) - 2 * mean_image .* 
filter2(ones([j, i]), image) + i*j* mean_image.^2 ) * ( sum(template(:).^2) -
i*j*template_mean.^2 ));

normxcorrmap = covariancemap ./ down_map;
end

%% 最小二乘相关 %%
function leastsquaresmap = leastSquares(template, image)
    leastsquaresMap = filter2(ones(size(template)), image.^2) - 2 *
filter2(template, image) + (ones(size(image)).*sum(template(:).^2));
end

%% 差绝对值相关 %%
function absmap = absDiff(template, image)
    [t_row, t_col] = size(template);
    [row, col] = size(image);
    absmap = zeros(row, col);

    image_fill = padarray(image, [t_row-1, t_col-1], 0, 'post');

    for x = 1:row
        for y = 1:col
            absmatrix = abs(image_fill(x:x+t_row-1, y:y+t_col-1) - template);
            cxy = sum(absmatrix(:));
            absMap(x,y) = cxy;
        end
    end
end

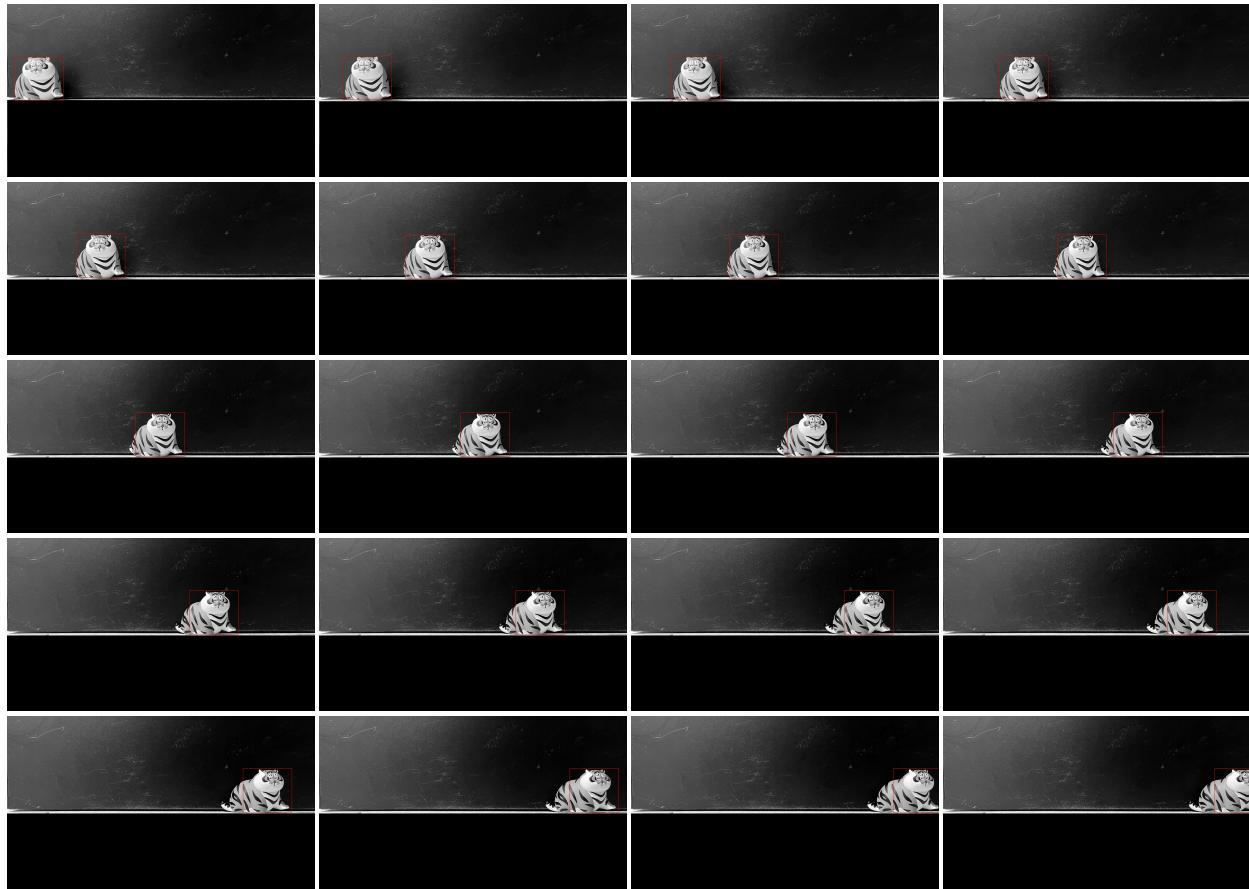
```

1.4. 实验结果

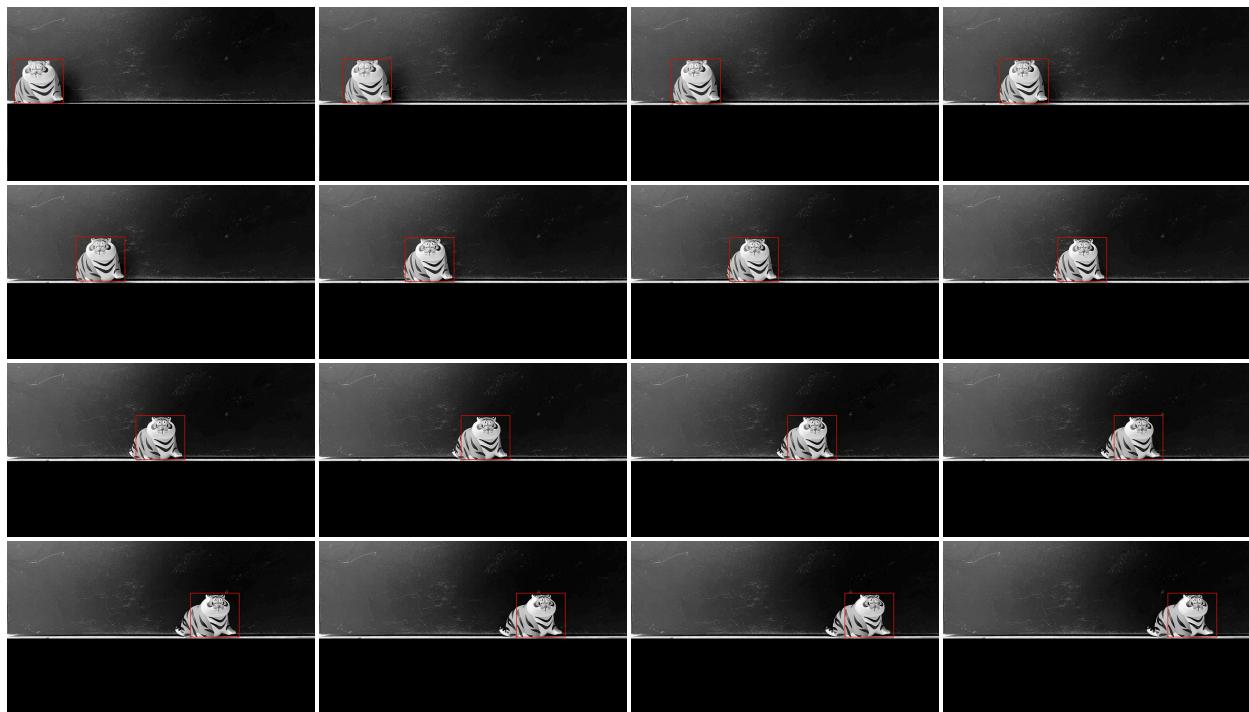
算法使用时间如表所示，准确率见图（基本都达到了极高的正确率）。

算法	直接	均值归一化	标准化	方差归一化（自定义）	方差归一化（内置）	最小二乘	差绝对值和
平均时间/s	0.0783	0.0959	0.0964	0.0955	0.0902	0.0910	0.0717

- 直接相关

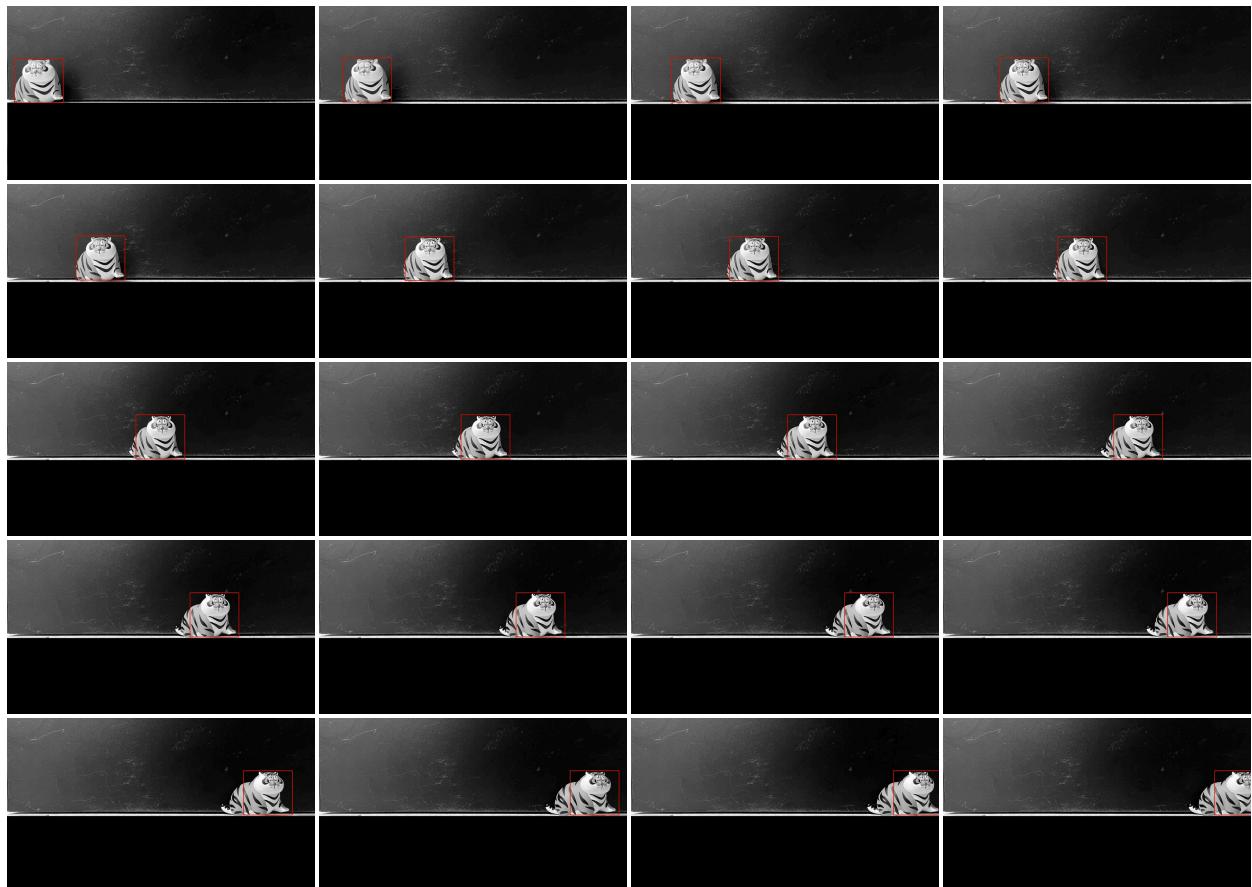


- 均值归一化相关

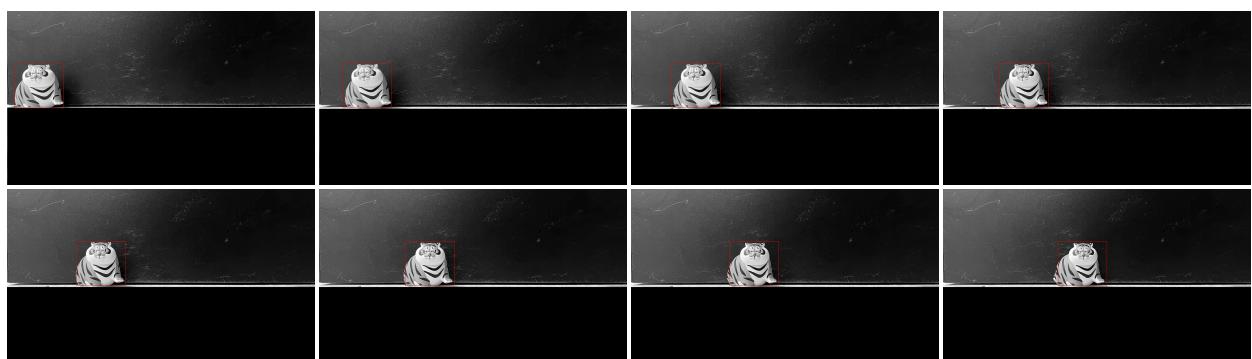


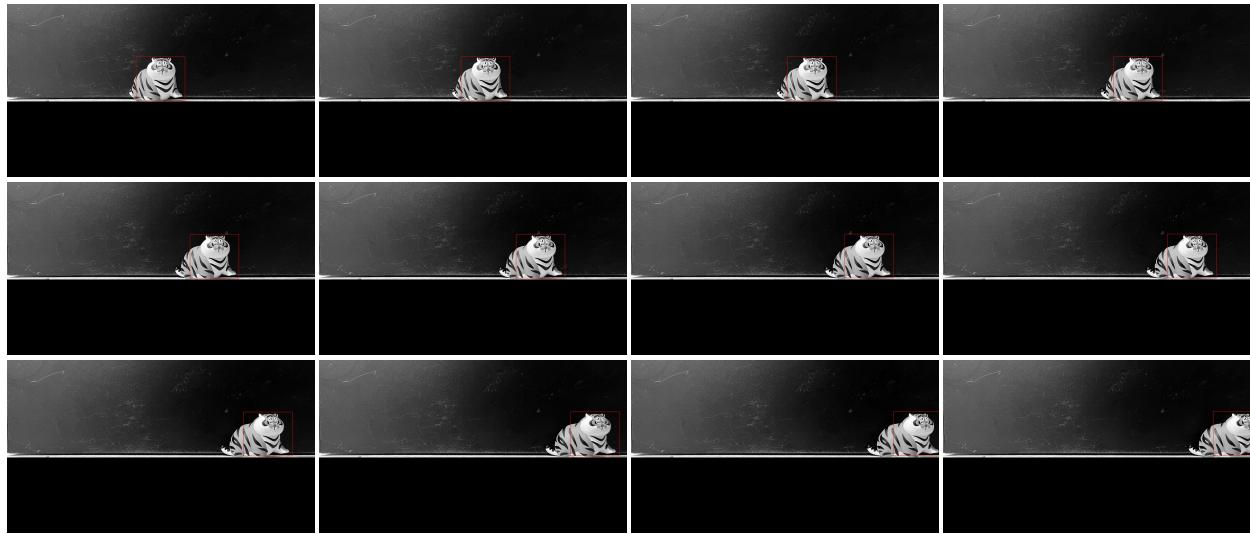


- 标准化

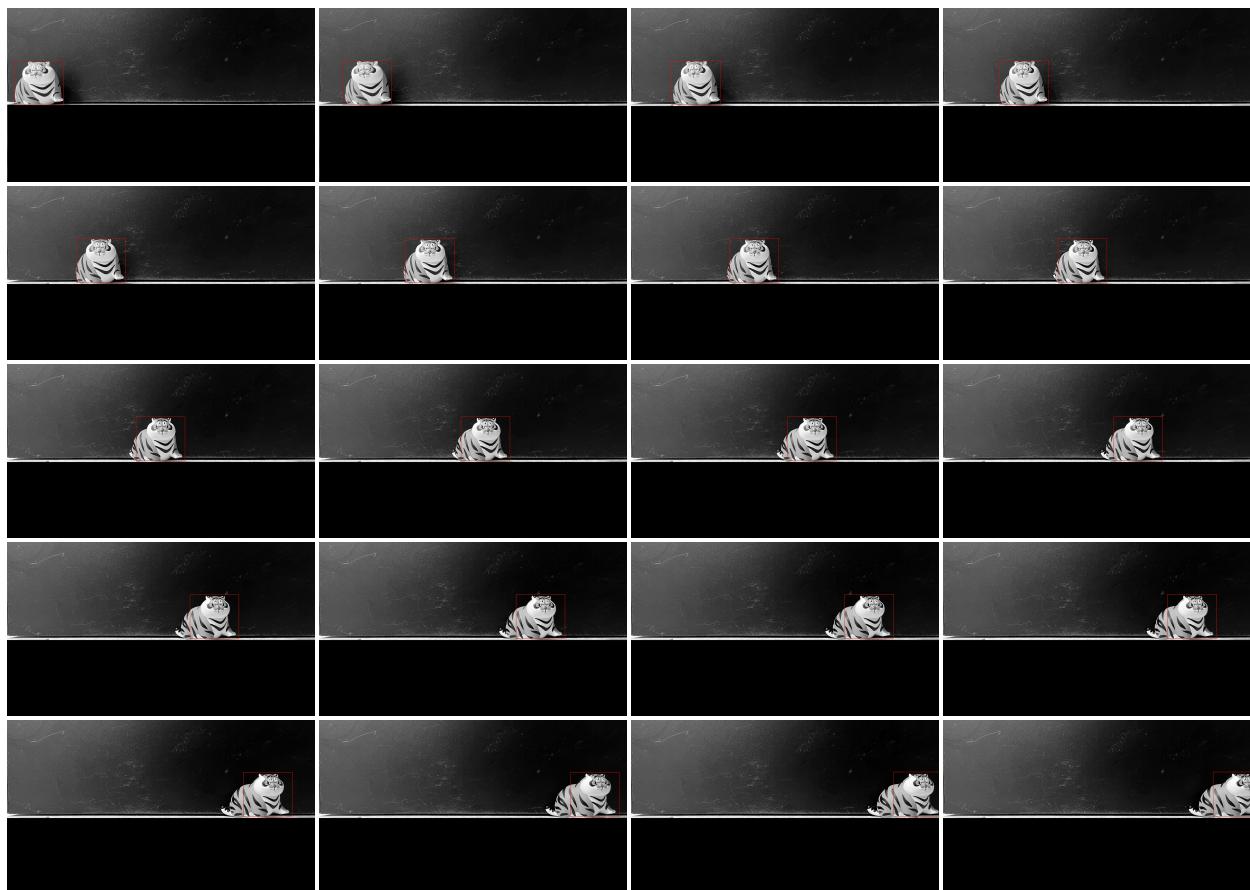


- 方差归一化（自定义）

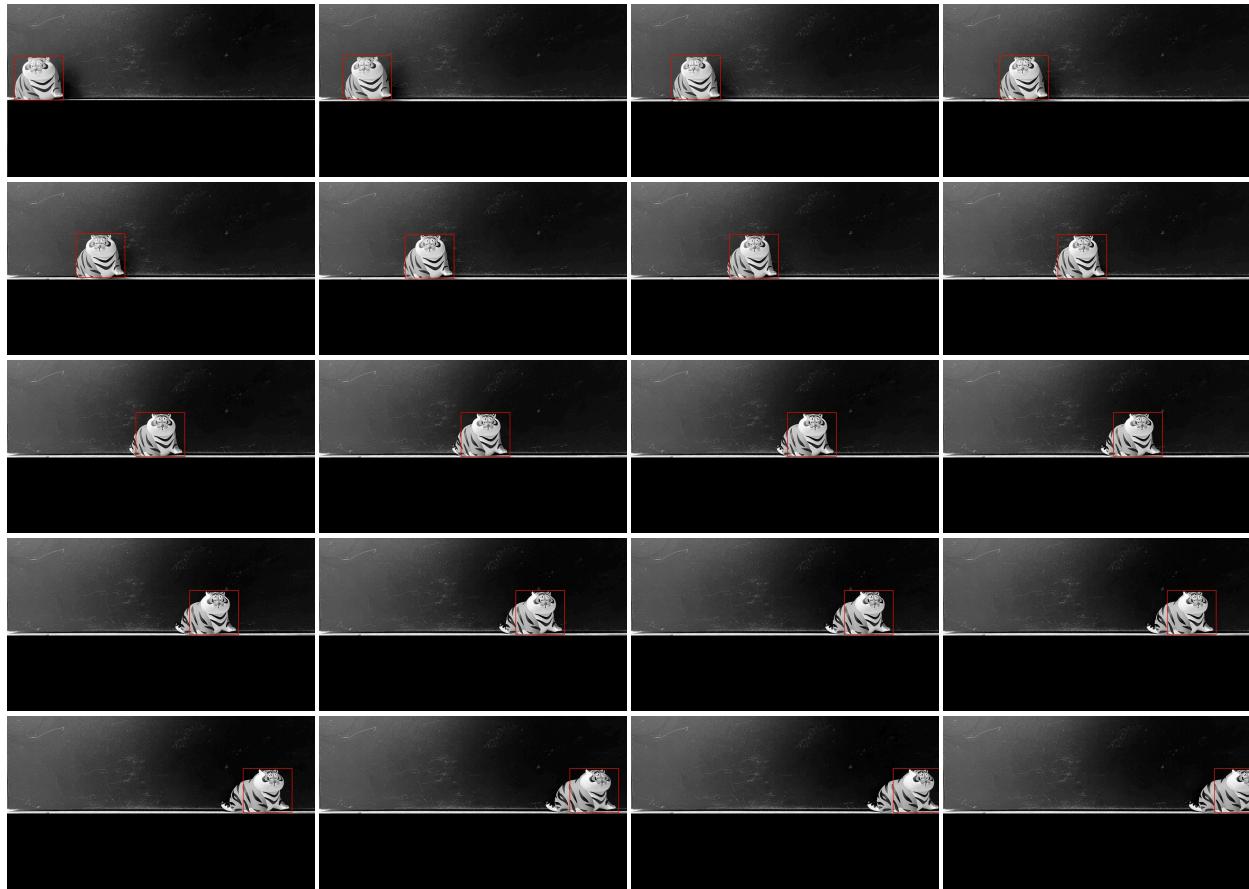




- 方差归一化（内置）



- 最小二乘



- 差绝对值和





1.5. 实验分析

通用处理单元（CPU）并不专注于矩阵计算，实验过程中发现普通计算机 CPU 对于大图像的遍历循环处理效率不高。实验的仿真平台 Matlab 后端对于矩阵运算做了大量优化，因此编码时利用矩阵运算代替循环，可以大大提高计算速度与精度。

仿真中使用了多种方法提高运行效率与准确率：

- 挑选拍摄场景，使得目标特征尽量明显，拍摄背景尽量纯净，先后拍摄试验了多组场景与目标特征：



Figure 1: 机箱背景、键盘遮盖

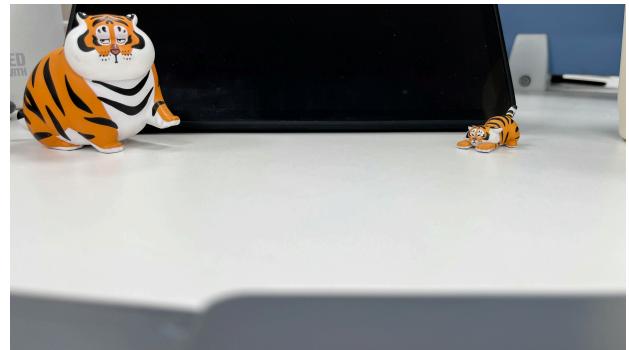


Figure 2: 桌面 iPad 部分黑色背景

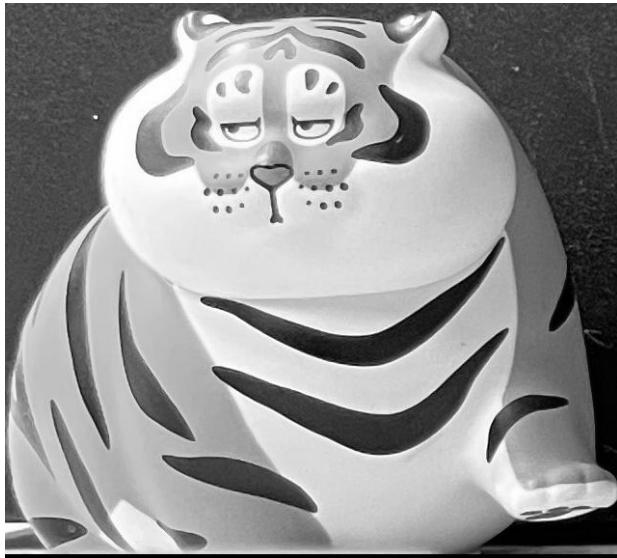


Figure 3: 猛虎全身模板



Figure 4: 猛虎眼神模板

实验发现如果图像背景尽量纯净，模板匹配准确率可以大幅提升，甚至原本无法进行正确匹配的相关方法也可以达到不错的效果。最后选定为 Figure 1 和 Figure 3，并将 Figure 1 中的下方灰色键盘也通过 GIMP 图片处理软件填充为纯黑色以在使用最大值的相关法匹配中达到更好的匹

配效果（最大值尽量与背景黑色灰度值 0 相差较大），相反地，使用最小值的相关法背景为白色最好（最小值与背景白色灰度值 255 相差较大）。

需要注意的是其实目标并非作完全平移运动。限于手机相机镜头，图像边缘实际有一定扭曲畸变，并且镜头与目标距离一定并且较为接近，无法观测目标在一个平面的运动，而是在小孔成像的模型中有一定的旋转和放大。

- 缩小由手机拍摄获取的高清序列图片，使用 `imresize` 函数将图像从原 4032×2268 分辨率缩小至 2016×1134 (0.5 倍) 或 1008×567 (0.25 倍)，同时模板也对应缩小为同样比例，对于本次定性实验，大大降低了算力要求，具体由每帧十几秒的匹配速度提升到一秒一帧。
- 使用二维滤波器函数 `filter2` 代替双循环加快图片处理，具体来说，已知 `filter2(g, f)` 的数学公式为

$$C(x, y) = \sum_{(i,j) \in W} f(x+i, y+j)g(i, j)$$

其中， V 为原图像， W 为模板， $C(x, y)$ 为卷积结果。 f 和 g 分别是原图像和模板的像素值。即 `filter2` 函数的输出是一个与目标图像像素一一对应的相关值矩阵。将各相关系数配合 Matlab 中的矩阵对内部每个元素进行处理的 `.` 运算以使用 `filter2`：

- 直接相关：

$$C(x, y) = \sum_{(i,j) \in V} \sum_{(i,j) \in W} f(x+i, y+j)g(i, j)$$

- 均值归一化相关：

$$\begin{aligned} C(x, y) &= \sum_{(i,j) \in W} [f(x+i, y+j) - f_m][g(i, j) - g_m] \\ &= \sum_{(i,j) \in W} f(x+i, y+j)g(i, j) - f(x+i, y+j)g_m - \sum_{(i,j) \in W} f_m[g(i, j) - g_m] \\ &= \sum_{(i,j) \in W} f(x+i, y+j)g(i, j) - \sum_{(i,j) \in W} f(x+i, y+j)I(i, j)g_m \end{aligned}$$

其中， I 为全一矩阵。

- 标准化相关：

$\sum_{(i,j) \in W} g^2(i, j)$ 是一个与 x, y 无关的常数，可以提取以与前面的 $\sum_{(i,j) \in W} f^2(x+i, y+j)$ 分离

$$C(x, y) = \frac{\sum_{(i,j) \in W} f(x+i, y+j)g(i, j)}{\sqrt{\sum_{(i,j) \in W} f^2(x+i, y+j)I(i, j) \sum_{(i,j) \in W} g^2(i, j)}}$$

- 方差归一化相关：

$\sum_{(i,j) \in W} [g(i, j) - g_m]^2$ 是一个与 x, y 无关的常数，可以提取以与前面的 $\sum_{(i,j) \in W} [f(x+i, y+j) - f_m]^2$ 分离

$$\begin{aligned}
C(x, y) &= \frac{\sum_{(i,j) \in W} [f(x+i, y+j) - f_m][g(i, j) - g_m]}{\sqrt{\sum_{(i,j) \in W} [f(x+i, y+j) - f_m]^2 \sum_{(i,j) \in W} [g(i, j) - g_m]^2}} \\
&= \sum_{(i,j) \in W} f(x+i, y+j)g(i, j) - f(x+i, y+j)I(i, j)g_m \\
&\quad * \frac{1}{\sqrt{\sum_{(i,j) \in W} f^2(x+i, y+j) - 2f_m f(x+i, y+j) + f_m^2 \sum_{(i,j) \in W} [g(i, j) - g_m]^2}} \\
&= \sum_{(i,j) \in W} f(x+i, y+j)g(i, j) - f(x+i, y+j)I(i, j)g_m \\
&\quad * \frac{1}{\sqrt{\sum_{(i,j) \in W} f^2(x+i, y+j)I(i, j) - 2f_m \sum_{(i,j) \in W} f(x+i, y+j)I(i, j) + \sum_{(i,j) \in W} f_m^2}} \\
&\quad * \frac{1}{\sqrt{\sum_{(i,j) \in W} g^2(i, j) - i * j * g_m^2}}
\end{aligned}$$

其中， I 为全一矩阵； f_m 同样使用 `filter2` 可以避免使用双循环，构造均值滤波器 `mean_filter $\frac{1}{ij}I(i, j)$` ，则 f_m 可以表示为 `filter2(mean_filter, f)`

- 最小二乘相关：

$$\begin{aligned}
C(x, y) &= \sum_{(i,j) \in W} [f(x+i, y+j) - g(i, j)]^2 \\
&= \sum_{(i,j) \in W} f^2(x+i, y+j) - 2f(x+i, y+j)g(i, j) + g^2(i, j) \\
&= \sum_{(i,j) \in W} f^2(x+i, y+j)I(i, j) - 2 \sum_{(i,j) \in W} f(x+i, y+j)g(i, j) + \sum_{(i,j) \in W} g^2(i, j)
\end{aligned}$$

其中， I 为全一矩阵。

- 差绝对值和相关

差绝对值和较为特殊，没有使用 `filter2`，相对地，双循环需要在图像矩阵周围填充 0，本实验采用模板左上角对齐，因此在目标图片右边和下方填充 0。此时相关系数的最值就是匹配开始的坐标而不需要减去模板大小的偏移。

总结：直接、均值归一化、标准化、方差归一化寻找相关系数最大的坐标，最小二乘、差绝对值和相关寻找相关系数最小的坐标。经过实验，发现方差归一化相关法的应用能力最强，即使对于 Figure 2 场景以及 Figure 4 模板也能获得很好的匹配结果，匹配的准确性其次是最小二乘法较为优秀，匹配时间最短的是差绝对值法，因为其计算量最小，然而对于目标和模板要求很高，不太容易应用。综合来说，实际中最小二乘法很好地权衡了准确性与匹配效率，可以作为应用的首选。