# Phát triển hệ thống bắt bám mục tiêu thời gian thực bằng Raspberry Pi

Lê Vũ Nam[1*], Khổng Vũ Liêm[2], Nguyễn Văn Thư[1], Phạm Đình Quý[1]

[1]Viện Vật lý kỹ thuật, Viện Khoa học và Công nghệ quân sự;
[2]Viện Công nghệ thông tin, Viện Khoa học và Công nghệ quân sự.
[*]Email: lvnam.mta@gmail.com
*Nhận bài: 02/6/2023; Hoàn thiện: 26/7/2023; Chấp nhận đăng: 07/8/2023 ; Xuất bản: 25/10/2023.*
**DOI:** https://doi.org/10.54939/1859-1043.j.mst.90.2023.127-133

## TÓM TẮT

*Kỹ thuật bắt bám mục tiêu sử dụng các thuật toán tìm kiếm và dự đoán của máy tính để định vị và dõi theo các mục tiêu một cách tự động mà không cần con người can thiệp. Việc áp dụng kỹ thuật bắt bám mục tiêu vào nhiệm vụ theo dõi, quan sát sẽ giúp nhiệm vụ này trở nên hiệu quả và dễ dàng hơn. Một yêu cầu quan trọng của nhiệm vụ này đó là tốc độ bắt bám phải đủ nhanh để đảm bảo yêu cầu thời gian thực, đồng thời vẫn đảm bảo độ chính xác và ổn định. Ngoài ra, các thiết bị quan sát thường sử dụng phần cứng nhỏ gọn như máy tính nhúng với hiệu năng thấp, các camera lại có độ phân giải cao. Trong bài báo này, một thuật toán bắt bám dựa trên lọc tương quan Kernel (KCF) được sử dụng trên máy tính Raspberry Pi 4B để thực hiện bắt bám tàu thuyền trên biển. Kết quả thử nghiệm cho thấy bắt bám chính xác, ổn định, tốc độ bắt bám đạt tới 20 FPS với độ phân giải của camera là 1280×720 pixel.*

**Từ khoá:** Bắt bám mục tiêu; Raspberry Pi; KCF.

## 1. MỞ ĐẦU

Bắt bám mục tiêu dựa trên hình ảnh luôn là một lĩnh vực nghiên cứu quan trọng của thị giác máy tính và xử lý hình ảnh, thu hút được nhiều sự quan tâm của các học giả. Kỹ thuật này rất hiệu quả trong việc theo dõi, giám sát các mục tiêu di động, vì vậy, có ý nghĩa và tiềm năng lớn trong các nhiệm vụ theo dõi, giám sát. Các nhiệm vụ có sự góp mặt của bắt bám mục tiêu như: Giám sát an ninh tại các khu vực trọng yếu như khu vực quân sự, cơ quan chính phủ, sân bay, nhà ga,...; Hiện đại hóa vũ khí khí tài quân sự, bao gồm tên lửa đạn đạo, ngắm bắn của vũ khí, điều khiển bay, thiết bị trinh sát UAV,...; Hệ thống giao thông thông minh; Các phương tiện giao thông không người lái; Giao tiếp người máy [1-4],...

Có rất nhiều các thuật toán bắt bám với nguyên lý khác nhau và cũng có nhiều cách để phân loại. Nhóm tác giả chỉ xin giới thiệu qua một số thuật toán bắt bám phổ biến, chia thành 3 loại như sau. Một là, thuật toán bắt bám trực tuyến, lợi dụng những thuật toán phân loại và định vị đơn giản để nhận diện mục tiêu dựa trên những đặc trưng cơ bản nhất, do đó, có thể bắt bám mục tiêu một cách nhanh chóng. Những thuật toán loại này thường yêu cầu thấp về tài nguyên của phần cứng nhưng kết quả không mấy ổn định, đồng thời rất nhạy cảm với những vấn đề như che khuất, thay đổi hình dạng mục tiêu, phông nền phức tạp. Tiêu biểu có thể kể đến là thuật toán Mean Shift [5, 6], Thuật toán Frag Track [7, 8], Thuật toán bắt bám dựa trên Linear Subspace [9, 10]; Thuật toán bắt bám dựa trên Sparse Coding [11, 12],... Hai là, thuật toán bắt bám dựa trên lọc tương quan (Correlation Filter, CF), những thuật toán này đưa bài toán bắt bám mục tiêu thành bài toán lọc tương quan đối với khu vực tìm kiếm, vị trí mục tiêu là vị trí để bộ lọc (filter) cho giá trị lớn nhất. Thuật toán lọc tương quan cơ bản nhất do Bolme đề xuất, còn được gọi là thuật toán MOSSE dựa trên tổng phương sai [13]. Tiếp đó có rất nhiều các thuật toán cải tiến như các thuật toán dùng phương pháp thu thập mẫu tuần hoàn CSK và thuật toán lọc tương quan dựa trên không gian Kernel do Henriques đề xuất [14, 15]. Ngoài ra, còn có phương pháp STAPLE của Bertinetto, MKCF của Tang, ACFN của Choi [16-18],... Ba là, thuật toán bắt bám dựa trên học sâu, lợi dụng những đặc tính ưu việt về thiết lập đặc trưng của thuật toán học sâu để giải quyết bài toán bắt bám giúp cải thiện vượt trội về độ chính xác. Tuy nhiên, thông thường

những thuật toán này cần một lượng mẫu lớn và thời gian lâu để huấn luyện. Có rất nhiều các cải tiến của loại thuật toán này, như thuật toán HCFT của Ma, DeepSRDCF của Danelljan [19, 20-23],... Nhìn chung, mỗi thuật toán đều có những ưu nhược điểm nhất định, các thuật toán đơn giản có mức sử dụng tài nguyên thấp thường có kết quả bắt bám thiếu ổn định hoặc khả năng chống che khuất kém, các thuật toán phức tạp sẽ cho hiệu quả bắt bám tốt hơn nhưng thường hao tốn nhiều tài nguyên. Căn cứ vào mục đích sử dụng khác nhau, phải cân bằng giữa các yếu tố trên để lựa chọn thuật toán phù hợp nhất.

Xuất phát từ nhu cầu của một thiết bị quan sát tàu thuyền trên biển có chức năng bắt bám phục vụ theo dõi giám sát mục tiêu và thuận tiện cho thao tác đo cự ly bằng laze tích hợp trên thiết bị. Bài báo xây dựng về lý thuyết, lắp đặt phần cứng và lập trình phần mềm cho một hệ thống bắt bám tàu thuyền sử dụng một máy tính nhúng Raspberry Pi đảm bảo hiệu quả bắt bám mục tiêu tốt và đáp ứng thời gian thực. Việc sử dụng máy tính nhúng Raspberry Pi nhằm hạn chế tối đa về ích thước, khối lượng cũng như mức tiêu thụ điện năng của phần cứng trong thiết bị quan sát tổng thể, đồng thời đảm bảo lợi thế về giá thành và dễ dàng trong triển khai thực tế và khai thác sử dụng.

## 2. THUẬT TOÁN BẮT BÁM DỰA TRÊN LỌC TƯƠNG QUAN KERNEL

Căn cứ vào yêu cầu về tốc độ xử lý cũng như mức độ sử dụng tài nguyên của thuật toán, nhóm tác giả đã tiến hành các thử nghiệm, so sánh và lựa chọn ra thuật toán bắt bám dựa trên lọc tương quan Kernel (KCF) [15]. Đây là thuật toán cho hiệu quả bắt bám tốt, có tốc độ xử lý nhanh đủ đảm bảo yêu cầu thời gian thực với mức sử dụng tài nguyên thấp. Dưới đây là giới thiệu về nguyên lý và quy trình thực hiện của thuật toán này.

### 2.1. Nguyên lý thuật toán KCF

Thuật toán bắt bám KCF dựa trên mẫu lấy từ frame ảnh có chứa mục tiêu ban đầu để khởi tạo ra một bộ lọc và dùng nó trong việc lọc tìm ra vị trí của mục tiêu trong frame mới (Quá trình này tương tự như 1 quá trình huấn luyện trong học sâu) [15]. Cụ thể là tìm ra một hàm số $f(x) = w^T x$ để giá trị phương sai giữa nó và giá trị hồi quy $y_i$ là nhỏ nhất, KCF sử dụng hồi quy Ridge nên ta có:

$$loss = min_w \sum_i (f(x_i) - y_i)^2 + \lambda \|w\|^2 \quad (1)$$

Trong đó, $\|w\|$ biểu diễn mô đun của *w*, tham số chính quy hóa λ được thêm vào để tránh hiện tượng overfitting. Tính đạo hàm với *w* ta được:

$$w = (X^H X + \lambda I)^{-1} X^H y \quad (2)$$

Ma trận $X = [x_i]$ ($x_i$ là mẫu được lấy từ frame chứa mục tiêu ban đầu), *y* là ma trận của các giá trị hồi quy $y_i$. $X^{\mathrm{H}}$ là phép hoán vị Hermitian, $X^{\mathrm{H}}$=$(X^*)^{\mathrm{T}}$ với $X^*$ là số phức liên hợp của *X*. *I* là ma trận định danh. Vì giá trị tối ưu của *w* có thể biểu diễn thành tổ hợp tuyến tính của mẫu $x_i$:

$$w = \sum_{i=1}^{n} \alpha_i x_i \quad (3)$$

Nên hàm bộ lọc có thể biểu diễn thành:

$$f(z) = w^T z = \sum_{i=1}^{n} \alpha_i x_i^T . z \quad (4)$$

Thông qua xử lý tuyến tính hóa và thêm ánh xạ phi tuyến hóa ta được:

$$f(z) = \sum_{i=1}^{n} \alpha_i \phi(x_i)^T . \phi(z) = \alpha^T \phi(X) . \phi(z) \quad (5)$$

$$w = \phi(X)^T \alpha \quad (6)$$

Trong đó, $\alpha^T$=[$\alpha_i$], $\phi(X) = [\phi(x_i)^T]^T$, thay vào phương trình (1) ta được:

$$loss = min_\alpha \|\phi(X)\phi(X)^T \alpha - y\|^2 + \lambda \|\phi(X)^T \alpha\|^2 \quad (7)$$

Tính đạo hàm của α ta được:

$$\alpha = (\phi(X)\phi(X)^T + \lambda I)^{-1} y = (K + \lambda I)^{-1} y \quad (8)$$

Ma trận $K = \phi(X)\phi(X)^T$ là một ma trận tuần hoàn $K = C(k)$. Do đó, có thể đơn giản hóa α:

$$\alpha = (A.diag(F(k)).A^H + A.diag(\lambda\delta).A^H)^{-1} y = C\left(F^{-1}(\frac{1}{F(k) + \lambda\delta})\right) y \quad (9)$$

Ở đây, *diag()* biểu diễn ma trận đường chéo, tiến hành biến đổi fourier rời rạc (DFT) với cả 2 vế ta được:

$$F(\alpha) = \left(\frac{1}{F(k) + \lambda\delta}\right)^* \odot F(y) \quad (10)$$

Thay α vào công thức (5), ta được:

$$f(z) = (K^{xz})^T \alpha \quad (11)$$

Vì $K^{xz} = \phi^T(x)\phi(z)$ cũng là một ma trận tuần hoàn, nên để giải phương trình trên chúng ta tiến hành DFT để biến nó về ma trận đường chéo:

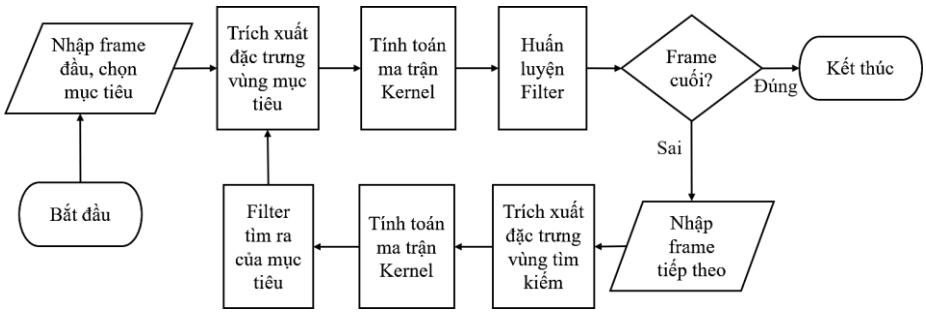
$$F\big(f(z)\big) = F(k^{xz}) \odot F(\alpha) \quad (12)$$

Trong đó:

$$k^{xz} = exp\left(-\frac{1}{\sigma^2}(\|x\|^2 + \|z\|^2 - 2F^{-1}(F(x)^* \odot F(z)'))\right) \quad (13)$$

Theo công thức trên, bằng việc tính toán giá trị của ma trận Kernel $k^{xz}$ thông qua các ma trận giá trị *x* và *z*, công thức của bộ lọc được xác định. Giá trị của $f(z)$ tại các vị trí có thể xem như một thao tác lọc không gian trên các giá trị $k^{xz}$.

**2.2. Quy trình thực hiện thuật toán KCF**




***Hình 1.*** *Lưu đồ thực hiện thuật toán bắt bám KCF.*

Hình 1 thể hiện lưu đồ của thuật toán bắt bám KCF, bắt đầu bằng việc đọc dữ liệu ảnh đầu vào (có thể từ video sẵn có hoặc từ camera). Sau khi nhập frame đầu tiên, người thao tác sẽ lựa chọn mục tiêu cần bắt bám (khoanh vùng hình chữ nhật quanh mục tiêu đó). Thuật toán sẽ trích xuất đặc trưng vùng thông qua sử dụng phép dịch chuyển tuần hoàn để tạo thành nhiều mẫu ảo phục vụ huấn luyện và tính toán ma trận Kernel theo công thức (13) ở mục 2.1. Sau đó, tiếp tục hình thành bộ lọc để phục vụ cho việc tìm kiếm mục tiêu ở frame sau. Ở frame sau đó, tiếp tục trích xuất đặc trưng của vùng tìm kiếm, là một vùng rộng hơn bao hàm vị trí của mục tiêu ở frame trước và tính toán ma trận Kernel tương ứng. Sau đó, thuật toán sử dụng bộ lọc đã tạo ra ở frame trước tiến hành lọc và tìm ra vị trí mới của mục tiêu ở frame hiện tại. Thuật toán tiếp tục

quay lại bước trích xuất đặc trưng vùng mục tiêu và tiến hành lại các bước như ở ở frame đầu và vòng lặp này lặp đi lặp lại với từng frame cho đến khi quá trình bắt bám kết thúc.

## 3. THỬ NGHIỆM VÀ KẾT QUẢ

Trong phần này, bài báo sẽ giới thiệu về hệ thống bắt bám, quá trình lập trình phần mềm để thử nghiệm thuật toán bắt bám và phân tích về kết quả thu được.

### 3.1. Thiết lập hệ thống và nội dung thử nghiệm

Hệ thống này bao gồm: 01 máy tính nhúng Raspberry Pi, 01 camera ngày, 01 màn hình hiển thị, bàn phím, chuột,... Sử dụng máy tính Raspberry Pi 4B với CPU 4 nhân 1.5 GHz 8GB RAM có hỗ trợ nhiều loại cổng giao tiếp tốc độ cao. Camera màu của Owon với cổng kết nối USB với độ phân giải 1280×720 pixel và tốc độ quét của camera là 30 FPS. Sử dụng một màn hình 15.6 inch độ phân giải 1920×1080 pixel để hiển thị hình ảnh. Sau khi kết nối camera, màn hình, bàn phím, chuột với máy tính thông qua các cổng USB và micro HDMI, tổng thể hệ thống như trong hình 2. Sau khi lắp đặt phần cứng nhóm tác giả cài đặt hệ điều hành Raspberry Pi OS 32x, cài phần mềm Python, OpenCV và các thư viện liên quan để phục vụ việc lập trình. Nội dung thử nghiệm bao gồm: Thử nghiệm độ chính xác, ổn định bắt bám và tốc độ của thuật toán KCF đối với các mục tiêu tàu thuyền; Thử nghiệm so sánh tính năng thuật toán KCF với một số thuật toán khác.


***Hình 2.*** *Ảnh tổng thể của hệ thống bắt bám dựa trên Raspberry Pi.*

### 3.2. Kết quả thử nghiệm và đánh giá

*3.2.1. Thử nghiệm độ ổn định bắt bám của thuật toán KCF đối với các mục tiêu tàu thuyền trong video sẵn có và ngoài thực tế*

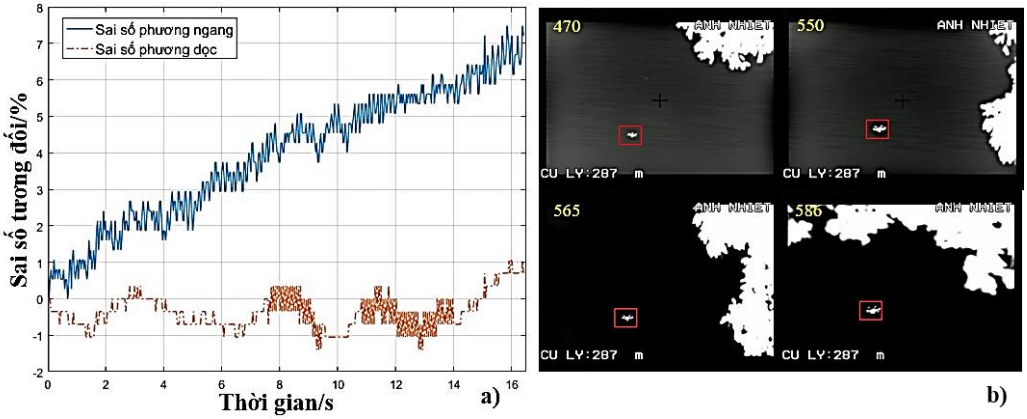
Để phục vụ cho việc thử nghiệm chúng tôi thu thập một số video sẵn có về tàu thuyền cũng như sử dụng camera để thu thập ảnh ngoài thực tế. Lưu ý, kết quả bắt bám của thuật toán với các mục tiêu trên video và qua camera là tương đương nhau, do đó, xử lý trên video sẵn có vẫn đánh giá được thuật toán một cách chính xác, chúng chỉ khác nhau ở tốc độ hiển thị kết quả.

Nhóm tác giả sử dụng thuật toán KCF tiến hành bắt bám với mục tiêu trong các video và thu được một số kết quả. Hình 3 giới thiệu một số kết quả bắt bám có tính đại diện, khung chữ nhật màu đỏ là kết quả bắt bám (hình 3a) và hình 3b) là các video sẵn có, hình 3c) là hình ảnh ngoài thực tế). Trong đó, hình 3 a) frame 243 và hình 3 b) frame 26 đều thể hiện khả năng bắt bám hiệu quả của thuật toán khi gặp phông nền phức tạp. Hình 3a) frame 450 và hình 3c) frame 214, 218 lần lượt thể hiện khả năng bắt bám của thuật toán khi gặp che khuất một phần và khả năng khôi phục bắt bám sau khi bị che khuất hoàn toàn. Hình 3b) frame 132, 199 đều thể hiện khả năng bắt bám tốt khi mục tiêu có hình dạng và kích thước thay đổi.

Qua thử nghiệm cho thấy thuật toán KCF có kết quả bắt bám tốt với khả năng giải quyết một số hiện tượng thường gặp khi bắt bám mục tiêu là phông nền thay đổi phức tạp, mục tiêu bị che khuất và hình dạng, kích thước mục tiêu thay đổi.


***Hình 3.*** *Kết quả bắt bám đối với một số mục tiêu tàu thuyền trên biển.*

*3.2.2. Tính toán sai số bắt bám và tốc độ của thuật toán KCF, so sánh với một số thuật toán phổ biến khác*




***Hình 4.*** *a) Sai số tương đối của thuật toán KCF theo thời gian;*
*b) Thử nghiệm thuật toán bắt bám với mục tiêu quay bởi camera hồng ngoại.*

Để đánh giá một cách định tính về độ chính xác của thuật toán, nhóm tác giả sử dụng sai số tương đối giữa vị trí bắt bám và vị trí thực của mục tiêu, theo hai phương dọc và ngang để đánh giá. Hình 4a) là đồ thị của sai số tương đối theo thời gian, trục tung biểu thị sai số tương đối tính theo%, trục hoành là thời gian tính bằng giây. Kết quả cho thấy sai số chủ yếu là phương ngang, cũng là phương di chuyển của mục tiêu. Trong 16 giây bắt bám sai số lũy kế là tương đối nhỏ chỉ khoảng 7%, lúc này khung bắt bám vẫn bao quát được mục tiêu. Tuy nhiên, sai số này có xu hướng tích lũy dần theo thời gian do đó nếu thời gian theo dõi quá lâu có thể dẫn đến bị trượt mục tiêu, lúc này cần phải tiến hành bắt bám lại. Về tốc độ của thuật toán, qua tính toán thu được tốc độ video đầu ra đạt tới 20 FPS, đảm bảo yêu cầu thời gian thực [24]. Như vậy, qua các thử nghiệm trên cho thấy thuật toán KCF đủ đảm bảo các yêu cầu về độ ổn định cũng như tốc độ để thực hiện nhiệm vụ bắt bám mục tiêu trên biển đã đề ra.

Bên cạnh đó, để so sánh thuật toán KCF với một số thuật toán phổ biến khác ở cùng một điều kiện thử nghiệm dựa trên các tiêu chí gắn với ứng dụng có thể kể đến như: Độ ổn định, tốc độ xử lý, khả năng khôi phục tìm kiếm sau khi bị che khuất hoàn toàn, mức sử dụng tài nguyên phần cứng, và tổng hợp như trong bảng 1. Qua so sánh có thể thấy, thuật toán KCF vừa có độ ổn định tốt, khả năng chống che khuất 1 phần, khả năng khôi phục tìm kiếm, tốc độ xử lý tương đối cao và mức sử dụng tài nguyên gần như thấp nhất (24%, chỉ xếp sau thuật toán MOSSE), là thuật toán phù hợp nhất cho yêu cầu bắt bám mục tiêu trên biển đã đặt ra ở phần mở đầu của bài báo này.

***Bảng 1.*** *So sánh thuật toán KCF với một số thuật toán khác.*

| Thuật toán | Độ ổn định | Tốc độ/FPS | Khả năng khôi phục | S.dụng tài nguyên/% |
|---|---|---|---|---|
| **KCF** | **Ổn định, chống che khuất 1 phần** | **19-20** | **Có** | **24** |
| Boosting | Tương đối ổn định | 5-13 | Không | 32 |
| CSRT | Rất ổn định, chống che khuất trong thời gian dài | 5-6 | Có | 41 |
| TLD | Tương đối ổn định | 1 | Có | 40 |
| MOSSE | Không ổn định, lỗi khi gặp che khuất hoặc phông nền thay đổi lớn | 17-18 | Có | 19 |
| MIL | Không ổn định | 4 | Không | 50 |
| Optical flow | Không ổn định, dễ trượt mục tiêu | 10-11 | Không | 28 |
| Medianflow | Không ổn định | 17-18 | Không | 44 |

Ngoài ra, nhóm tác giả cũng đã thử nghiệm thuật toán trên với mục tiêu là một drone đang bay được quay bởi camera hồng ngoại, như trong hình 4b). Kết quả cho thấy mặc dù mục tiêu kích thước nhỏ, hình dạng thay đổi và hình ảnh là ảnh đen trắng nhưng thuật toán vẫn đạt được kết quả bắt bám tốt. Việc áp dụng thuật toán bắt bám đối với ảnh ngày và ảnh hồng ngoại có ý nghĩa quan trọng trong việc ứng dụng nó trên các hệ thống quan sát đa kênh ngày đêm phục vụ quan sát cảnh giới các mục tiêu trên biển.

## 4. KẾT LUẬN

Trong bài báo này nhóm tác giả đã giới thiệu về thuật toán bắt bám dựa trên lọc tương quan Kernel, phân tích các ưu nhược điểm và đưa ra cơ sở lý luận để áp dụng cho hệ thống bắt bám tàu thuyền thời gian thực với phần cứng là máy tính nhúng Raspberry Pi. Qua thử nghiệm trên các video sẵn có và video thực tế từ camera quan sát các mục tiêu tàu thuyền trên biển đã làm sáng tỏ độ chính xác, ổn định và tốc độ của thuật toán KCF. Kết quả cho thấy thuật toán đều đảm bảo tốt các yêu cầu, giải quyết được các hiện tượng thường gặp trong bắt bám, đồng thời đáp ứng yêu cầu thời gian thực. Hệ thống bắt bám thời gian thực với Raspberry Pi mà chúng tôi xây dựng hoàn toàn đủ điều kiện để lắp đặt và sử dụng trong các thiết bị quan sát cảnh giới trên biển.

Thuật toán KCF vẫn chưa phải là tối ưu nhất về tốc độ bắt bám, do vậy, hướng nghiên cứu tiếp theo là cải thiện về tốc độ thuật toán để có FPS cao hơn để áp dụng được với các camera có độ phân giải cao hơn cũng như khả năng bắt bám nhiều mục tiêu đồng thời.

***Lời cảm ơn:*** *Nhóm tác giả chân thành cảm ơn sự tài trợ về kinh phí từ ngân sách của Sở Khoa học tỉnh Bà Rịa-Vũng Tàu.*

## TÀI LIỆU THAM KHẢO

[1]. T. Collins *et al., "Introduction to the Special Section on Video Surveillance"*, IEEE Transactions on Pattern Analysis and Machine Intelligence, **Vol. 22**, No. 8, 745-746, (2000).

[2]. J. Zhao, M. Wang, *"A Study on Missile Plume Tracking and Localizing by Means of Forward Looking Infrared (FLIR)"*, Journal of Solid Rocket Technology, **Vol. 23**, No. 4, 64-68, (2000).

[3]. V. Kastrinaki *et al., "A Survey of Video Processing Techniques for Traffic Applications"*, Image and Vision Computing, **Vol. 21**, No. 4, 359-381, (2003).

[4]. F. Bonin-Font *et al., "Visual Navigation for Mobile Robots: A Survey"*, Journal of Intelligent and Robotic Systems, **Vol. 53**, No. 3, 263-296, (2008).
[5]. D. Comaniciu *et al., "Real-Time Tracking of Non-rigid Objects Using Mean Shift"*, Proc of the IEEE CCV and PR, Washington, (2000).
[6]. Y. Cheng, *"Mean Shift, Mode Seeking, and Clustering"*, IEEE Transactions on Pattern Analysis and Machine Intelligence, **Vol. 17**, No. 8, 790-799, (1995).
[7]. A. Adam *et al., "Robust Fragments-Based Tracking Using the Integral Histogram"*, Proc of the IEEE CSCCV and PR, Washington, 798-805, (2006).
[8]. F. Wang *et al., "Robust and Efficient Fragments Based Tracking Using Mean Shift"*, AEU-International Journal of Electronics and Communications, **Vol. 64**, No. 7, 614-623, (2010).
[9]. M. Turk, A. Pentland, *"Eigenfaces for Recognition"*, Journal of Cognitive Neuroscience, **Vol. 3**, No. 1, 71-86, (1991).
[10]. S. Yan *et al., "Graph Embedding and Extensions: A General Framework for Dimensionality Reduction"*, IEEE Transactions on PA and MI, **Vol. 29**, No.1, 40-51, (2007).
[11]. D. Donoho, *"Compressed Sensing"*, IEEE Transactions on Information Theory, **Vol. 52**, No.4, 1289-1306, (2006).
[12]. J. Wright, *et al, "Sparse Representation for Computer Vision and Pattern Recognition"*, IEEE, **Vol. 98**, No. 6, 1031-1044, (2010).
[13]. D. Bolme, *et al., "Visual Object Tracking Using Adaptive Correlation Filter"*, Proc of the IEEE CCV and PR, Washington, 2544-2550, (2010).
[14]. J. Henriques *et al., "Exploiting the Circulant Structure of Tracking-by-Detection with Kernels"*, Proc of the 12th ECCV, Berlin, 702-715, (2012).
[15]. J. Henriques *et al., "High-Speed Tracking with Kernelized Correlation Filter"*, IEEE Transactions on Pattern Analysis and Machine Intelligence, **Vol. 37**, No. 3, 583-596, (2014).
[16]. L. Bertinetto, *et al., "Staple: Complementary Learners for Real-Time Tracking"*, Proc of the IEEE ICCV&PR, Washington, 1401-1409, (2016).
[17]. M. Tang, J. Feng, *"Multi-kernel Correlation Filter for Visual Tracking"*, Proc of the IEEE ICCV, Washington, 3038-3046, (2016).
[18]. J. Choi *et al., "Attentional Correlation Filter Network for Adaptive Visual Tracking"*, Proc of the IEEE ICCV&PR, Washington, 4807-4816, (2017).
[19]. C. Ma, *et al., "Hierarchical Convolutional Features for Visual Tracking"*, Proc of the IEEE ICCV, Washington, 3074-3082, (2015).
[20]. M. Danelljan *et al., "Learning Spatially Regularized Correlation Filters for Visual Tracking"*, Proc of the IEEE ICCV, Washington, 4310-4318, (2015).
[21]. D. Meimetis, *et al., "Real-time multiple object tracking using deep learning methods"*, Neural Computing and Applications, **Vol. 35**, 89-118, (2023).
[22]. Z. Soleimanitaleb, M. A. Keyvanrad, *"Single Object Tracking: A Survey of Methods, Datasets, and Evaluation Metrics"*, Computer Vision and Pattern Recognition, (2022).
[23]. H. Huadi *et al., "Survey of Target Tracking Based on Improved Block Algorithm of Correlation Filtering"*, Software Guide, **Vol. 22**, No. 3, 245-252, (2023).
[24]. H. Grabner, *et al., "Real-Time Tracking via On-line Boosting"*, Proc of the British Machine Vision Conference, Edinburgh, (2006).

## ABSTRACT

### Development of a real-time object-tracking system using Raspberry Pi

*Object tracking uses computerized algorithms to locate and track targets automatically without human intervention. Applying object-tracking technology to the observation mission will make it more effective and easier. An important requirement was that the tracker must be fast enough to meet the real-time requirements while still ensuring accuracy and stability. In addition, observation equipment usually uses compact hardware (such as embedded computers) and high-resolution cameras. In this paper, a Kernel Correlation Filter (KCF) based tracking algorithm is used with a Raspberry Pi 4B to track objects at sea. The experiment results show that the tracker works well and stably, and the tracking speed reaches 20 FPS with 1280×720 pixels of camera resolution.*

**Keywords:** Object tracking; KCF; Raspberry Pi.