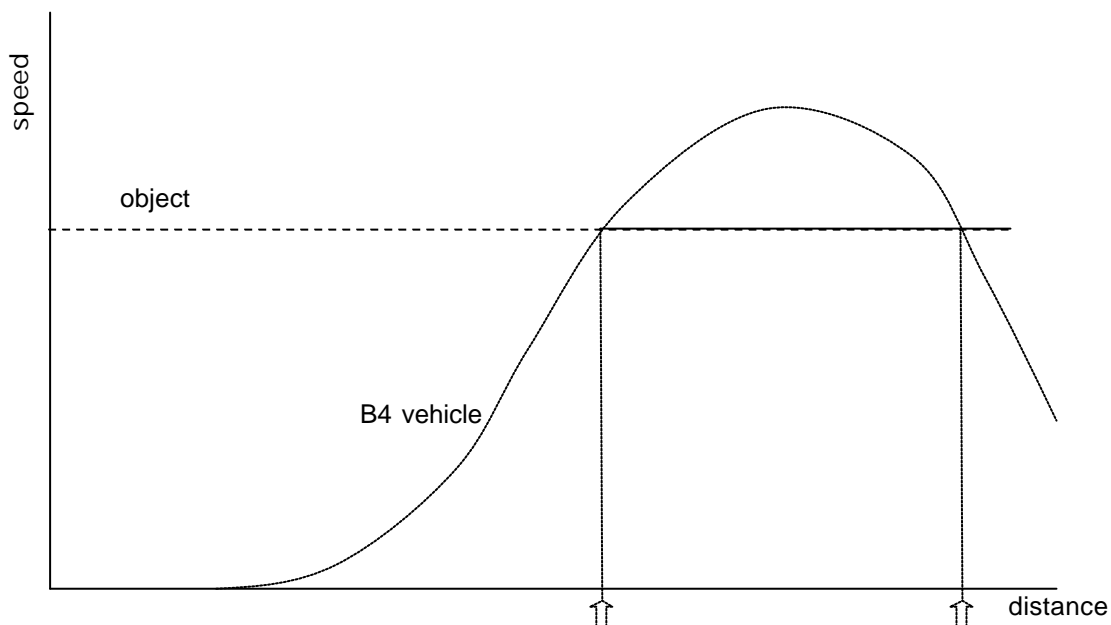


*theoretical tasks***1. architecture of a Braitenberg type 4 vehicle**

In a Braitenberg type 4 vehicle the motor's dependency on the sensors is nonlinear. If we look at a type 4 vehicle that follows a slow-moving object, it would probably look like this:

The vehicle follows the object with a lower speed than the object itself. It looks like the object is running away. Then, when the vehicle reaches a certain distance to the object, it will speed up and catch up with the object. As it gets nearer to the object it again slows down. The cycle is going to start over again.

Here we're talking about a constant distance between the vehicle and the object. So, we actually need to place the vehicle in the exact position where its motor speed is the same as the speed of the moving object.

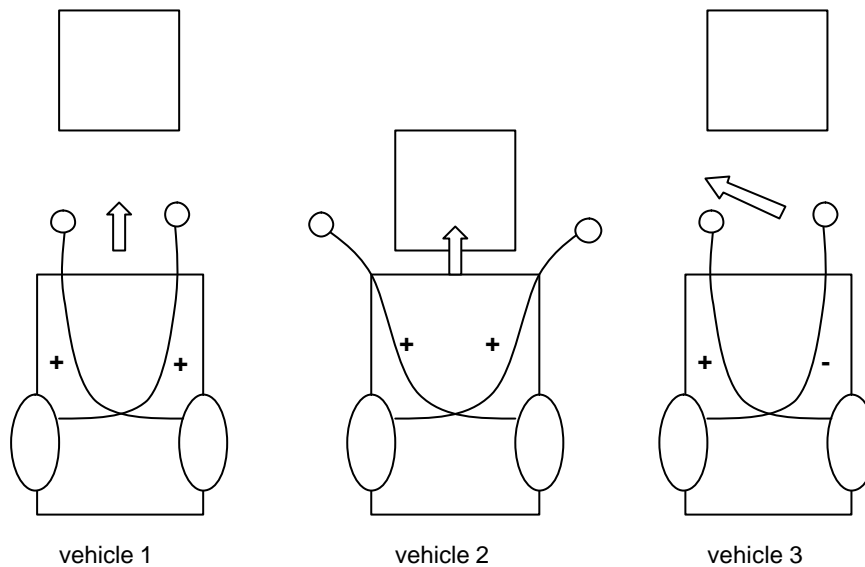
**2. Segmentation of behavior**

If we take a look at a Braitenberg type 3c vehicle, obstacle avoiding and light following vehicle, it is pretty hard to segment its behavior. 3c is a multisensorial vehicle. The agent's behavior is an emergent behavior of all its internal variables. We – as observers – cannot really tell what the agent exactly is involved in, whether it is "turning toward the light" or "turning away from the obstacle". If we didn't know anything about the internal mechanism of such a vehicle, we would probably have a hard time in designing such an agent. Actually the internal mechanism of such a vehicle is pretty simple. All we need is two proximity and two light sensors, on the left and on the right front side of the vehicle. The emergent behavior looks much more "intelligent" as it actually is, so it's kind of dangerous to just segment an agent's behavior on observation alone.

If we look at a Gigabot's behavior as an observer, it looks like a "tidy-up" behavior to us. Actually the internal mechanism is much simpler. It's just a simple obstacle avoidance mechanism realized by two simple sensors.

Here, again, we focus a F-O-R problem. We cannot actually look at the problem from the agent's view. We're always the observer.

### 3. Deriving different behaviors – sensor positioning



The sensor positioning is terribly important for an agent's behavior. It got a huge influence. Vehicle 1 would avoid obstacles that are right in front of it. If it drives straight into an obstacle it would speed up at the end and bump into it. As we don't want to build a self-destructive vehicle, we might program the sensors, that at a certain amount of sensor intension the right motor will go much faster than the left. Like this the vehicle would still turn left before bumping into the obstacle.

If we place the sensors as it is shown in vehicle 2, we're discovering an emergent behavior as we saw in tests with Gigabots. For an observer it looked like the Gigabots were cleaning up the floor as they were accumulating the blocks. Actually this just happened because the Gigabot couldn't see what is in front of it and just pushed the block. This is a very good example to see how important the sensor positioning actually is.

### 4. Motor controller commands vs. actual wheel speed

We cannot just design a motor speed scale from 1 to 10 to control the wheels. The actual wheel speed is always relative to the robots environment. It is a huge difference whether the robot runs on an ice surface or a sandy desert floor. On an ice surface the wheels would spin and the robot may not even get forward at all. If we would have a plain surface with no resistance between surface and wheels at all, then we could relate the motor controller commands to the actual wheel speed. But physically, something like this doesn't exist. We still got air resistance. Right, we could neglect this, but I just want to make clear that there is no "perfect world". Real world is always a complicated environment and we have to think of every single detail that could affect the robot.

What about, if the surface inclines? That would make the whole thing again much more complicated. Then we would also have to take note of the robot's direction. Facing the hill or driving downwards definitely is not the same, considering the actual wheel speed.

*practical tasks***2. motors.khepera****a) Get the robot to move straight:**

line 62, uncomment `state = TURN_STATE;`

```

    if(counter > STRAIGHT_DELAY)
    {
        // state = TURN_STATE;
        counter = 0;
    }

```

Like this the bot will always stay in `STRAIGHT_STATE` and will never turn.

**b) Try to make the robot move along a square trajectory:**

On top, we define `TURN_DELAY` 24 milliseconds (I'm going to call it 'milliseconds' in this text, even though this value is not defined). The resulting figure shows a (almost perfect) pentagon. As 24 milliseconds correspond to  $72^\circ$ , we choose 30 milliseconds to get a  $90^\circ$  angle:

```

// Default forward speed
#define FORWARD_SPEED 5
#define STRAIGHT_DELAY 100 // the time the robot moves straight
#define TURN_DELAY 30 // the time the robot turns
#define STRAIGHT_STATE 0
#define TURN_STATE 1

```

The problem is that we cannot specify a value for the angle the robot should turn. As we know the robot got no sensors for direction changes, that's why it is impossible to specify an angle. First of all it is hard to match the correct amount of milliseconds for a special angle. Second, again, it all depends on the environment as mentioned in theoretical task 4.

It seems to be impossible to get the robot move along a correct square trajectory.

**4. braiten2.khepera**

The two sensors in the front of Khepera got the same stimulation curve. If Khepera drives straight against a wall, it would speed up and bump straight into it. To avoid this, we could make one sensor less sensitive than the other.

→ Here we got still a Braitenberg type 2 vehicle

Adjusting sensor stimulation values

→ still a Braitenberg type 2 vehicle

If Khepera drives straight against a wall we could give her the power to decide whether to turn left or right. Either we build some kind of random generator in Khepera's "brain" or we give her the power of a neuronal net

→ Braitenberg type 5 (neuronal nets)

With a Braitenberg type 2 vehicle it is hard and seems to be impossible, to handle all kind of obstacles and traps of the "real world"!

## 5. lightseeker.khepera

This is a Braitenberg type 3 vehicle because it uses two different kinds of sensor systems, the light sensors and the proximity sensors.